НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО» ***Факультет інформатики та обчислювальної техніки***

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Веб-сервіс адміністрування IT-зустрічей*

Виконав: студент IV курсу, групи *ІП-63Калініченко Владислав Сергійович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. в. кафедри АСОІУ Халус О.А.* _____
посада, науковий ступінь, вчене звання, прізвище, і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада, науковий ступінь, вчене звання, прізвище, і ім’я, по батькові (підпис)

Рецензент:

проф. каф. ТК, д.т.н., проф. Стенін О.А.
посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові (підпис)

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

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Калініченко Владислава Сергійовича
(прізвище, ім’я, по батькові)

1. Тема проєкту « Веб-сервіс адміністрування ІТ-зустрічей »

керівник проєкту *Халус О.А., ст. викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог, математичне забезпечення
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, аналіз безпеки даних
3) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, процесів тестування, опис контрольного прикладу
4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, робота з програмним забезпеченням

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема структурна діяльності

4) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*19.03.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*26.03.2020*	
3.	*Постановка та формалізація задачі*	*26.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*02.04.2020*	
5.	*Алгоритмізація задачі*	*02.04.2020*	
6.	*Моделювання програмного забезпечення*	*09.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*16.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*23.04.2020*	
9.	*Розробка програмного забезпечення*	*30.04.2020*	
10.	*Налагодження програми*	*07.05.2020*	
11.	*Виконання графічних документів*	*14.05.2020*	
12.	*Оформлення пояснювальної записки*	*21.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*	*03.05.2020*	
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____Владислав КАЛІНІЧЕНКО
(підпис)

Керівник _____ Олена ХАЛУС
(підпис)

№з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6313.045440.02.81	Пояснювальна записка	58	
3	А4	КПІ.ІП-6313.045440.03.91	Технічне завдання	12	
4	А4	КПІ.ІП-6313. 045440.04.51	Програма та методика тестування	6	
5	А4	КПІ.ІП-6313.045440.05.34	Керівництво користувача	7	
6	А4	КПІ.ІП-6313.045440.06.13	Опис програми	18	
7	А3	КПІ.ІП-6313.045440.07.99.СС	Схема структурна варіантів використання	1	
8	А3	КПІ.ІП-6313.045440.07.99.СБД	Схема бази даних	1	
9	А3	КПІ.ІП-6313.045440.07.99.СС	Схема структурна діяльності	1	
10	А3	КПІ.ІП-6313.045440.07.99.КЕ	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6313.045440.01.90		
Розробн.		Калініченко В.С.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Халус О.А.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 30 таблиць, 16 рисунків та 10 джерела.

Об’єкт дослідження: сервіси адміністрування ІТ-зустрічей.

Мета дипломного проєкту: полегшення роботи власників великих клубів, які мають свої філії в різних куточках планети, шляхом делегування основних організаційних процесів довіреним особам.

У першому розділі проведено аналіз відомих технічних рішень і розроблено вимоги до програмного забезпечення. Побудовано схему структурну варіантів використання.

У другому розділі було розроблено архітектуру сервісу як комп’ютерного додатку. Побудовано структурну схему класів.

У третьому розділі проведено тестування сервісу аналізу ринку іноземних валют за розробленим планом тестування. Описано процес тестування.

У четвертому розділі описано розгортання та впровадження сервісу, створено керівництво користувача.

У додатках наведено: текст програмного коду.

КЛЮЧОВІ СЛОВА: ВЕБ-СЕРВІС, АДМІНІСТРУВАННЯ, ІТ-ЗУСТРІЧІ

ABSTRACT

The explanatory note of the diploma project consists of four sections, 30 tables, 16 illustrations and 10 sources.

The object of study: web services for administering of IT meetups

The aim of the diploma project: facilitating the work of owners of large clubs that have their branches in different parts of the world, by delegating the main organizational processes.

In the first section the analysis of known technical solutions was provided and software requirements were developed. The use-case diagram was constructed.

In the second section the architecture of the service as desktop application was described and developed. A structural diagram of classes is constructed.

In the third section, the application was tested according to the developed test plan. The process of testing was described.

The fourth section describes the deployment and implementation of the application. The user guide was provided.

Appendices contain the description of the program.

KEYWORDS: WEB SERVICE, ADMINISTERING, IT MEETUPS

Пояснювальна записка
до дипломного проєкту

на тему: Веб-сервіс адміністрування ІТ-зустрічей

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Бібліотека – збірник програм, що виконують деякі типові задачі та застосовуються для полегшення розробки програмного забезпечення.

JSON – це текстовий формат який створений для обміну даними. Він базується на тексті, може бути прочитаним людиною

Дерево (деревовидна структура даних) – спосіб представлення ієрархічної структури даних у графічній формі.Неможливість молодих програмістів набути реального досвіду спричиняє негативну тенденцію для всього ринку. Тому метою проєкту є покращення коммунікації між розробниками різних рівнів.

HTTP — протокол передачі даних, який використовується в комп'ютерних мережах. Скорочена назва від Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів.

XML (eXtensible Markup Language) — розширювана мова розмітки. Рекомендований Консорціумом Всесвітньої павутини (W3C). Специфікація XML описує XML-документи і частково описує поведінку XML-процесорів (програм, які читають XML-документи і забезпечують доступ до їх вмісту). XML розроблявся як мова з простим формальним синтаксисом, зручний для створення і обробки документів програмами і одночасно зручний для читання і створення документів людиною, з підкресленням націленості на використання в Інтернеті.

CRUD — (англ. create read update delete) 4 базові функції управління даними «створення, зчитування, зміна і видалення».

SAX (англ. «Simple API for XML») — спосіб послідовного читання / запису XML-файлів.

BPMN (англ. Business Process Model and Notation, нотація і модель бізнес-процесів) — система умовних позначень і їх опису в XML для моделювання бізнес-процесів.

Heroku — хмарна PaaS-платформа, что підтрімує ряд мов програмування. Зараз список підтрімуваніх мов включає в себе Java, Node.js, Scala, Clojure, Python и PHP. На серверах Heroku використовуються операційні системи Debian або Ubuntu.

GraphQL — це мова запитів та маніпуляції даними з відкритим кодом для API і середовище виконання для обслуговування запитів з наявних даних.

API (програмний інтерфейс програми, інтерфейс прикладного програмування) (англ. Application programming interface) — опис варіантів взаємодії між різними додатками.

ВСТУП

Зараз ринок праці розділився на дві частини: з одного боку досвідчені або дуже талановитий програмісти, які можуть легко знайти роботу, з іншого молоді фахівці в яких не достатньо практичних навичок, яких зараз вимагають майже в кожній компанії. Неможливість молодих программістів набути реального досвіду спричиняє негативну тенденцію для всього ринку. Тому метою проєкту є покращення коммунікації між розробниками різних рівнів.

Актуальність теми: Наразі існує багато сервісів для організації тематичних зустрічей, але метою більшості з них є заробіток шляхом впровадження платних послуг та продажу персональних даних. Та зазвичай такого типу сервіси не надають повного відчуття володіння своєю спільнотою, що є досить великим недоліком. Отже завядки цій розробці власник спільноти зможе у повній мірі насолодитися керуванням та відслідковування усіх процесів.

В наш час дуже важливо вміти швидко масштабуватись. Але зазвичай аналогічні сервіси не спрямовують свій погяд на цей аспект. Цей проєкт надає можливість власникам всесвітніх IT-клубів делегувати більшу частину своєї роботи довіреним особам, що дозволяє без великих проблем масштабувати свою спільноту, тим самим підвищуючи відсоток зв’язків між розробниками та роботодавцями.

Завдання розробки: авторизація користувачів, реєстрація користувачів, перегляду зустрічей, бронювання місць на зустрічі, пошук зустрічей, видалення зустрічей, оновлення інформації про зустрічі.

Практичне значення одержаних результатів: розроблено програмне забезпечення у вигляді веб-сервісу, що надає можливість адміністрування IT-зустрічей. Особливість програмного забезпечення полягає в можливості створення амбасадорів, для пришвидшення процесів масштабування спільноти.

Публікації: Калініченко В.С. ОПТИМІЗАЦІЯ РОБОТИ З DOM ШЛЯХОМ ІНТЕГРАЦІЇ JAVASCRIPT-ОБ’ЄКТА(VDOM) // IV Всеукраїнська

науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління – ІСТУ-2020».

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Веб-сервіс це програмна система, що ідентифікується рядком URI, чиї публічні інтерфейси та прив'язки визначені і описані за допомогою XML. Опис такої програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису за допомогою повідомлень, заснованих на XML, і переданих за допомогою Інтернет-протоколів. Веб-сервіси забезпечують наступні переваги системі: взаємодія програмних компонентів можливо незалежно від платформи, завдяки використанню протоколу HTTP можлива взаємодія компонентів шляхом запитів.

З боку логічної організації: сервісом можна назвати ресурс, який реалізує бізнес-функцію, що володіє наступними властивостями:

– є повторно використованою;

– визначається одним або декількома технологічно-незалежними інтерфейсами;

– слабкий зав’язок з іншими подібними ресурсами і може бути викликаний за допомогою комунікаційних протоколів, що забезпечують взаємодію ресурсів між собою.

Використання веб-сервісів дозволяє послабити зв'язаність компонентів веб-додатку. Для надання користувачеві функцій, які реалізуються веб-сервісом, потрібна організація взаємодії трьох компонентів: джерела запиту до веб-сервісу, постачальника веб-сервісу і брокера веб-сервісів, що виступає посередником між джерелом і постачальником веб-сервісу.

В наш час все більше уваги приділяється інформаційним технологіям. Тому ІТ-галузь вже не перший рік очолює рейтинг сфер, що мають попит у світі. Одним із стримуючих розвиток факторів є брак талановитих та кваліфікованих кадрів.

Однак для того щоб працювати у великій компанії та отримувати гідну зарплату, в наш час не вистачить лише вміння писати програми. Зараз при наявності різних методорогій розробки, таких як Scrum або Kanban вміння комунікувати з командою іноді має більш важливий статус.

Soft skills - це ті навички, які допомагають спеціалісту виконувати свою робу більш ефективно. Саме вони дозволяють чітко формулювати свої думки, обробляти та систематизувати інформацію, а також швидко адаптуватися в умовах непередбачуваних ситуаций, які досить часто трапляються на сучасних проєктах. В університетах намагаються надати базові знання з цього напрямку, але дійсний досвід можливо отримати тільки в бойових умовах.

Англійська мова, не меньш важливий критерій відбору фахівців, оскільки більшість інформації необхідної для розробки програм існує, або виключно англійською, або першоджерела, які оновлюються значно швидже, є англомовними. Також все частіше ІТ-компаніям потрібні люди які можуть брати на себе більше відповідальності аніж просто написання программи за планом. Як показує практика вміння розмовляти з англомовним кліентом та можливість донести до нього свої думки сприяє покращенню відносин між компанією та замовником.

Таким чином було вирішено, що наявність сервісу, який зможе покращити одрзу всі аспекти необхідні у сфері ІТ, значно покращить становище фахівців не тільки в нашій країні але й в інших куточках світу.

1.2 Змістовний опис і аналіз предметної області

Основною задачею, в даній роботі, було створення сервісу адміністрування офлайн ІТ-зустрічей, де всі учасники могли б ділитися досвідом та заводити нові знайомства. Хоча зараз існують такі соціальні мережі, як LinkedIn, що дозволяють фахівцям потрапляти на співбесіди та знаходити своє місце в різноманітних компаніях, залишається проблема порозуміння між новим колегою та командою, яка працює разом деякий час та

має свої звички. Саме через це продуктивність роботи команди може різко знижуватись через те що новий член команди, дуже часто, соромиться висловлювати свої думки відкрито. Це сприяє поширеню помилок та погіршеню становища нового члена команди.

Виходячи з усіх проблем, покращення комунікації шляхом створення сервісу, який дозволить організовувати офлайн зуструрічи в неформальній обстановці є досить актуальною темою. Це дозволить стерти рамки між найманими фахівцями та роботодавцями. Такий підхід значно прискорює процес адаптації, що в свою чергу позитивно відображається на темпах розробки продуктів компанії.

Сервіс не обмежує організаторів лише однією країною, а навпаки сприяє поширенню створеної спільноти по всьому світу. Завдяки цьому вирішується ще одна сучасна проблема. Англійська мова - найпоширеніша мова у світі, також загальноприйнята як еталон усіх мов програмування. Далеко не всі фахівці мають можливість працювати за межами своєї країни через не достатній рівень розмовних навичок. Але якщо ви є частиною всесвітньої спільноти, можливості щодо покращення ситуації значно зростають. Представимо ситуацію коли ви вирішили полетіти в іншу країну де є ваша спільнота, у цьому разі виші шанси знайти роботу в іншій країні значно зростають, та навіть якщо ви просто намагаєтесь вивчити англійску, з великою ймовірністю познайомитесь з тим хто вам допоможе.

Користувачі сервісу мають можливість відфільтрувати зустрічи за місцем проведення, за тематикою, побачити інформацію про присутніх та обрати місце за столом. Одною з відмінностей є те, що ці зустрічі не відбуватимуться для великих мас, вони проводитимуться лише для групп не більше ніж 20 чоловік. Таких підхід дозволяє бути впевненим що пристуні з великою ймовірністю познайомляться один з одним не лише на рівні імен. Такий підхід був експерементально доведений в 2005 році та широко застосовується по сьогодні.

Діаграму діяльності сервісу зображено на рисунку 1.1.

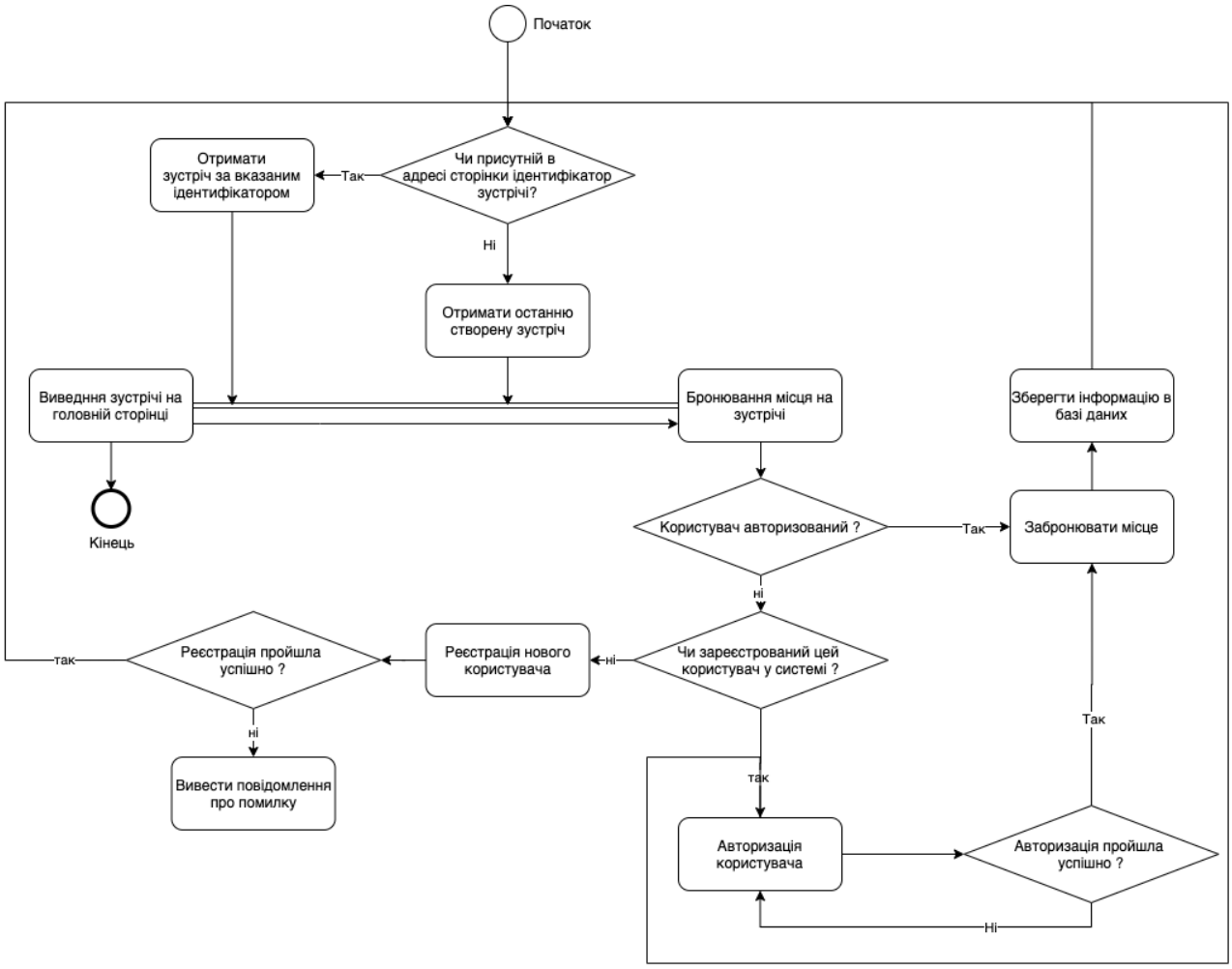


Рисунок 1.1 – Схема структурна діяльності

1.3 Аналіз успішних IT-проєктів

1.3.1 Аналіз відомих технічних рішень

Найпоширінішою реалізацією сервісів для адмініміністрування офлайн зустрічей є звичайний CRUD з використання реляційних баз даних та системою авторизації. Такий підхід є загальноприйнятим рішенням у багатьох системах. Більше відмінностей можна знайти у реалізації візуальної частини. На сьогодні існує дуже багато бібліотек та фреймворків, які дозволяють полегшити створення складних сервісів, а також пришвидшити їх роботу. Значна частина бібліотек зосереджена на оптимізації роботи з DOM шляхом інтеграції javascript-об’єкта, а саме Virtual DOM.

Віртуальний DOM – концепція програмування, в якій ідеальне або «віртуальне» представлення інтерфейсу користувача зберігається в пам’яті і синхронизується зі «справжнім» DOM. Цей процес називається узгодженням. Як результат така техніка дозволяє нам поліпшити продуктивність на клієнтській стороні, уникаючи прямої роботи з DOM шляхом роботи з легким javascript-об’єктом, що імітує DOM-дерево.

З точки зору об'єктно-орієнтованого програмування, DOM визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленям документів у вигляді дерева. Все це призначено для того, аби надати можливість комп'ютерній програмі доступу та динамічної модифікації структури, змісту та оформлення документа. Через те, що структура документа зображена у вигляді дерева, повний зміст документа аналізується та зберігається в пам'яті комп'ютера. Тому, DOM підходить для використання в програмах, які вимагають багаторазовий доступ до елементів документа в довільному порядку. В разі, якщо треба лише послідовний або одноразовий доступ до елементів документа, рекомендується, для прискорення переробки та зменшення обсягів необхідної пам'яті комп'ютера, застосовувати послідовну модель роботи зі структурованими документами (SAX).

Замість того, щоб взаємодіяти з DOM безпосередньо, робота відбувається з його полегшеною копією. Це надає можливість вносити зміни в копію, виходячи з потреб користувача, а після цього застосовувати зміни до реального DOM.

При цьому відбувається порівняння DOM-дерева з його віртуальної копією, визначається різниця і запускається перерисовка того, що було змінено.

Такий підхід працює швидше, бо не включає в себе всі тяжкі частини реального DOM.

Але тільки якщо ми робимо це правильно. Є дві проблеми: коли саме робити повторну перерисовку DOM і як це зробити ефективно.

Є два варіанти дізнатися, що дані змінилися:

– «dirty checking» (брудна перевірка) полягає в тому, щоб опитувати дані через регулярні проміжки часу і рекурсивно перевіряти всі значення в структурі даних;

– «observable» (спостережуваний) полягає в спостереженні за зміною стану. Якщо нічого не змінилося, ми нічого не робимо. Якщо змінилося, ми точно знаємо, що потрібно оновити.

Шляхи досягнення швидкодії роботи з віртуальним DOM-деревом:

– використання ефективних алгоритмів порівняння;

– угруповання операцій читання / запису при роботі з DOM;

– ефективне оновлення тільки під-дерев.

Реалізація може виявитися досить складною, але є деякі бібліотеки, які допомагають реалізувати цей підхід.

У цьому проєкті для оптимізації роботи візуальної частини використовується бібліотека React.js. При зміні двох дерев, React спочатку порівнює два кореневих елемента. Всякий раз, коли кореневі елементи мають різні типи, React буде руйнувати старе дерево і будувати нове дерево з нуля. Перехід від посилання (<a>) до зображення (<img>), або від кнопки (<button>) до контейнеру ( ) - будь-який з них призведе до повного відновлення. При зриві дерева старі вузли DOM знищуються. Отримують екземпляри компонентів «componentWillUnmount». При створенні нового дерева нові DOM-вузли вставляються в DOM. Отримують екземпляри компонентів «componentWillMount» і потім «componentDidMount». Будь-який стан, пов'язаний зі старим деревом, втрачається. Будь-які компоненти нижче кореня також будуть демонтовані і знищені.

При порівнянні двох елементів React DOM того ж типу React розглядає атрибути обох, зберігає один і той же базовий вузол DOM і оновлює тільки змінені атрибути. Інтерфейс додатку був розроблений з урахування компонентного підходу, вигляд головної сторінки можна побачити на рисунку 1.2.

Рисунок 1.2 - Головна сторінка

Для візуалізації роботи програми використовувалися шаблони, написані на HTML5, застосовувалися таблиці стилів CSS. Розглянемо кожен з елементів більш докладно. HTML (Hyper Text Markup Language) - мова для опису веб-сторінок. Під мовою розмітки розуміється набір тегів, кожен з яких описує вміст документа. Мова HTML інтерпретується браузерами і отриманий в результаті інтерпретації форматований текст відображається в зрозумілому для людини вигляді. CSS (від англ. Cascading Style Sheets - каскадні таблиці стилів) - це набір критеріїв для форматування, який застосовується до елементів веб-сторінки а саме для управління їх видом і станом. Таблиці стилів дозволяють змінити зовнішній вигляд і розташування всіх сторінок веб-сайту шляхом редагування лише одного файлу.

Авторизація користувача відбувається за допомогою технології JSON Web Token. Після першого логіна, клієнту повертається згенерований сервером JWT. При кожному наступному запиті, клієнт повинен передавати JWT встановленим API способом (наприклад, через заголовок або як параметр запиту). Сервер декодує header і payload і перевіряє заздалегідь зарезервовані поля. Якщо все в правильно, за вказаною в header алгоритмом складається підпис. Якщо отриманий підпис збігається з переданим, користувача авторизують. . Вигляд форми авторизації можна побачити на рисунку 1.3.

Рисунок 1.3 - Форма авторизиції

1.3.2 Аналіз відомих програмних продуктів

При аналізі Web-застосувань для адміністрування офлайн зустрічей були обрані такі сайти для дослідження: Meetup.com, Diobox.

Meetup.com – один з найпопулярніших сервісів подібного формату. Відомий у всьому світі завдяки своїй простоті, але через не можливість безкоштовного використання втрачає велику кількість користувачів. Також одним з мінусів є те що він створений з метою охопити усі ніші. Але через цю

універсальність не можливо створити свою спільноту в яка б суттєво відрізнялась від усіх інших на цьому сайті. Meetup - це онлайн-сервіс, який використовується для створення груп, які проводять місцеві події. Станом на 2017 рік сервіс налічує близько 35 мільйонів користувачів. Кожен користувач може бути членом групи або RSVP для будь-якого іншого події. Користувачі використовують веб-сайт для пошуку друзів або для професійних заходів. Користувачі, що зустрічаються, не мають "підписників" або інших зв'язків один між одним, як на інших веб-сайтах для заходів.

Користувачі Meetup самоорганізуються в групи. Станом на 2017 рік є близько 225 000 груп у 180 країнах. Кожна група має різні теми, розміри та правила. Групи поділяються на більш ніж 30 категорій.

Групами у Meetup керують приблизно 140 000 організаторів. Будь-який користувач Meetup може бути організатором. Організатори створюють групи, організовують заходи та розробляють план подій. Вони також сплачують плату за керівництво групою, сподіваючись розділити вартість іншими учасниками. Meetup має жорстку політику щодо організації зустрічей навколо комерційних заходів, слів ненависті чи груп, які не зустрічаються особисто. Приклад роботи роботи цього сервісу ви можете побачити на рисунку 1.4.

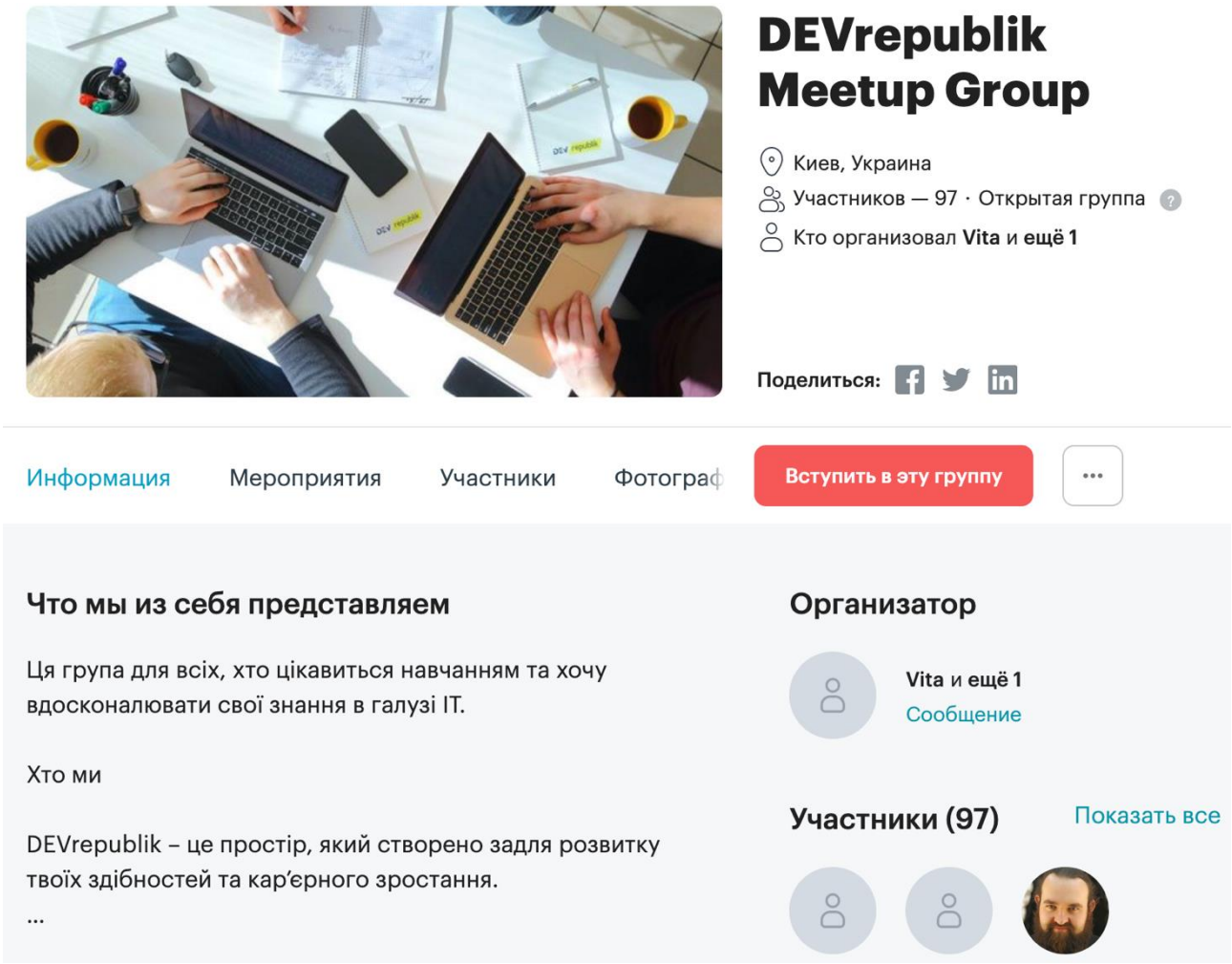


Рисунок 1.4 - Meetup.com

Diobox – це сервіс адміністрування офлайн зустріч, який має дуже широкий спектр можливостей, аде нажаль також є платним та не дає можливості зробити свою спільноту брендовою. Цей сервіс має багато функцій, наприклад створення свого сайту, та попри всі переваги ви не маєте можливості використовувати унікальний дизайн. А його інструменти організації зустрічей унеможливлює делегування та створення амбасадорських програм.

Diobox надає майже все необхідне для ефективного управління вашими зустрічами. Можливість імпортувати наявний гостевий лист та зберегти всю власну контактну інформацію є його найбільшою перевагою. Після того як ви завантажили усіх гостей, треба класифікувати їх на основі їх підтвердження.

Для подальшого сегментування можна створити додаткові списки та позначити їх своїми гостями.

Також сервіс пропонує використовувати вбудований модуль CRM Diobox для відстеження уподобань гостей, планів подорожей та інших заходів для персоналізації їх взаємодії з подіями. Приклад роботи цього сервісу ви можете побачити на рисунку 1.5.

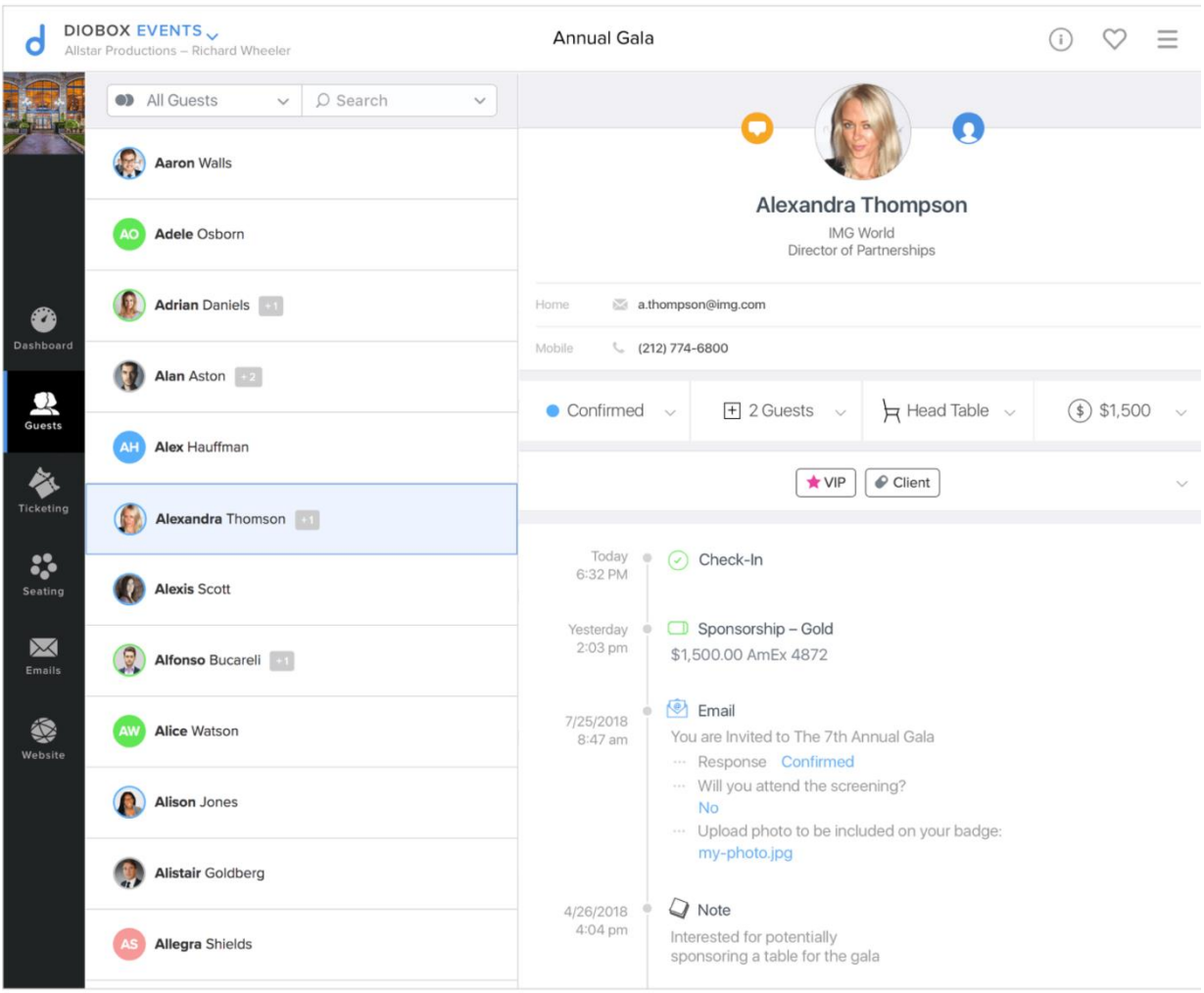


Рисунок 1.5 - Diobox

Таким чином можна сказати що ці сервіси розроблені на широку аудиторію що є досить великим плюсом, та по при всі переваги, для створення спільноти світового масштабу досить вагомими є унікальність та бренд.

1.4 Аналіз вимог до програмного забезпечення

Програмне забезпечення повинно надавати можливість створення ролей у системі. Адміністратор може створити амбасадора та керувати усією іншою інформацією. Амбасадор в свою чергу має можливість створювати та видаляти лише ті зустрічи які створив він сам. Таким чином усі процеси будуть під контролем власника сервісу. Авторизація обох повинна відбуватися в адмін панелі. Яка в свою чергу надаватиме наступні можливості:

- реєстрація у системі;
- авторизація у системі;
- перегляд усіх наявних у системі зустрічей;
- створення нових зустрічей;
- видалення зустрічей;
- оновлення існуючи зустрічей;
- вихід із системи.

Підводячи підсумок програмне забезпечення повинне надавати наступні варіанти використання:

Таблиця 1.1 - Варіант використання UC-1

Унікальний ідентифікатор	UC-1
Назва	Реєстрація у системі
Опис	Неавторизований користувач проходить реєстрацію в додатку.
Учасники	Неавторизований користувач
Передумови	- неавторизований у системі користувач; - у системі є хоча б одна зустріч; - на обраній зустрічі є хоча б одне вільне місце;

Продовження таблиці 1.1

Передумови	- користувач заповнив форму реєстрації та натиснув кнопку «зареєструватись».
Постумови	Користувач зареєстрований у системі
Сценарій	- неавторизований користувач намагається забронювати місце на зустрічі; - у формі авторизації натискає на кнопку «реєстрація»; - заповнює всі поля форми валідними даними; - натискає на кнопку «зареєструватись».

Таблиця 1.2 - Варіант використання UC-2

Унікальний ідентифікатор	UC-2
Назва	Авторизація у системі
Опис	Неавторизований користувач проходить авторизацію в додатку.
Учасники	Неавторизований користувач
Передумови	- зареєстрований у системі користувач; - у системі присутня хоча б одна зустріч; - на обраній зустрічі є хоча б одне вільне місце; - користувач заповнив форму авторизації валідними даними.
Постумови	Користувач авторизований у системі

Продовження таблиці 1.2

Сценарій	- неавторизований користувач намагається забронювати місце; - у формі авторизації заповнює всі поля валідними даними; натискає кнопку «авторизуватись».

Таблиця 1.3 - Варіант використання UC-3

Унікальний ідентифікатор	UC-3
Назва	Пошук зустрічі
Опис	Користувач шукає зустріч за обраними критеріями.
Учасники	Користувач
Передумови	- у системі присутня хоча б одна зустріч; - користувач вказав критерії пошуку; - користувач вводить валідні для шуканої зустрічі дані.
Постумови	Користувачу виведено список зустрічей які догоджають критеріям відбору
Сценарій	- користувач обирає критерії пошуку зустрічі; - сервіс автоматично відправляє запити, на оновлення списку відповідних зустрічей, при оновленні критерію; - користувачу виводиться інформація про знайдені зустрічі.

Таблиця 1.4 - Варіант використання UC-4

Унікальний ідентифікатор	UC-4
Назва	Бронювання місця
Опис	Користувач бронює місце на обранії зустрічі
Учасники	Користувач
Передумови	- у системі присутня хоча б одна зустріч; - користувач авторизований у системі; - на обраній зустрічі є вільне місце.
Постумови	Користувач забронював місце, та за столом з'явилось його зображення
Сценарій	- користувач натискає на місце за столом; - його бронювання зберігається у системі; на обраному місці з'являється його зображення.

Таблиця 1.5 – Варіант використання UC-5

Унікальний ідентифікатор	UC-5
Назва	Створення зустрічі
Опис	Адміністратор, або амбасадор створює нову зустріч
Учасники	Адміністратор, Амбасадор
Передумови	- авторизований користувач – адміністратор або амбасадор; - усі необхідні поля заповнені валідними даними. - у системі немає ідентичної зустрічі.
Постумови	У системі з'явилась нова зустріч

Продовження таблиці 1.5

Сценарій	- адміністратор або амбасадор заповнює усі поля форми створення зустрічі валідними даними; - натискає на кнопку «створити»; на екрані з’являється повідомлення про успішно створену зустріч.

Таблиця 1.6 – Варіант використання UC-6

Унікальний ідентифікатор	UC-6
Назва	Видалення зустрічі
Опис	Адміністратор, або амбасадор видаляє зустріч
Учасники	Адміністратор, Амбасадор
Передумови	- авторизований користувач – адміністратор або амбасадор; - уси необхідні поля заповнені валідними даними; - у системі немає ідентичної зустрічі.
Постумови	У системі більше не існує видаленої зустрічі
Сценарій	- адміністратор або амбасадор натискає на кнопку «видалити» в полі обраної зустрічі; - система виводить повідомлення про успішне видалення зустрічі; - у таблиці всіх зустрічей більше не відображається видалена зустріч.

Таблиця 1.7 – Варіант використання UC-7

Унікальний ідентифікатор	UC-7
Назва	Оновлення інформації про зустріч
Опис	Адміністратор, або амбасадор оновлює інформацію про зустріч
Учасники	Адміністратор, Амбасадор
Передумови	- авторизований користувач – адміністратор або амбасадор; - у системі присутня хоча б одна зустріч; оновлена інформація не дублює основну інформацію в присутніх у системі зустрічах.
Постумови	У системі змінилась інформація про щойно оновлену зустріч;
Сценарій	- адміністратор або амбасадор натискає на кнопку «змінити» в полі обраної зустрічі; - змінюється необхідна інформація та натискається кнопка «зберегти»; - у системі оновлюється інформація про щойно змінену зустріч.

Таблиця 1.8 – Варіант використання UC-8

Унікальний ідентифікатор	UC-8
Назва	Створення амбасадору
Опис	Адміністратор створює нового користувача «Амбасадор»
Учасники	Адміністратор
Передумови	- авторизований користувач -

Продовження таблиці 1.8

Передумови	- адміністратор; форма створення амбасадору заповнена валідними даними.
Постумови	У системі зареєстрований амбасадор
Сценарій	- адміністратор заповнює поля форми валідними даними; - натискає кнопку «створити»; - на екрані з'являється повідомлення про успішне створення амбасадора; - у системі з'яляється зареєстрований амбасадор.

Таблиця 1.9 – Варіант використання UC-9

Унікальний ідентифікатор	UC-9
Назва	Видалення амбасадору
Опис	Адміністратор видаляє амабасадора
Учасники	Адміністратор
Передумови	- авторизований користувач - адміністратор; - у системі присутні хоча б один зареєстрований амбасадор.
Постумови	У системі більше не існує видаленого амбасадору
Сценарій	- адміністратор натискає кнопку «видалити» у полі обраного амбасадору; - на екрані з'являється повідомлення про те що амбасадор був видалений; - у системі більше не видаленого

Продовження таблиці 1.9

Сценарій	амбасадору.

При відкриванні головної сторінки повинна виконуватись наступна умова: якщо у вказаній адресі є унікальний ідентифікатор зустрічі, то відкрити сторінку цієї зустрічі, інакше відкрити сторінку з останньою створеною зустріччу у системі.

На головній сторінці звичайний користувач повинен мати можливість зареєструватись у системі та забронювати обране місце. Також у його арсеналі повинні бути інструменти швидкого пошуку зустрічей за вказаним місцем проведення, часом та тематикою. У розділі «минулі зустрічи» користувач повинен отримати стислий витяг про зустрічі які вже відбулися. При переході за посиланням «дізнатися більше» користувач повинен потрапити на сторінку з детальним описом обраної зустрічі, де зможе переглянути розклад усіх подій, меню та фотографії місця проведення.

Для зображення опцій була використана діаграма варіантів використання. Побачити діаграму варіантів використання можна на рисунку 1.6.

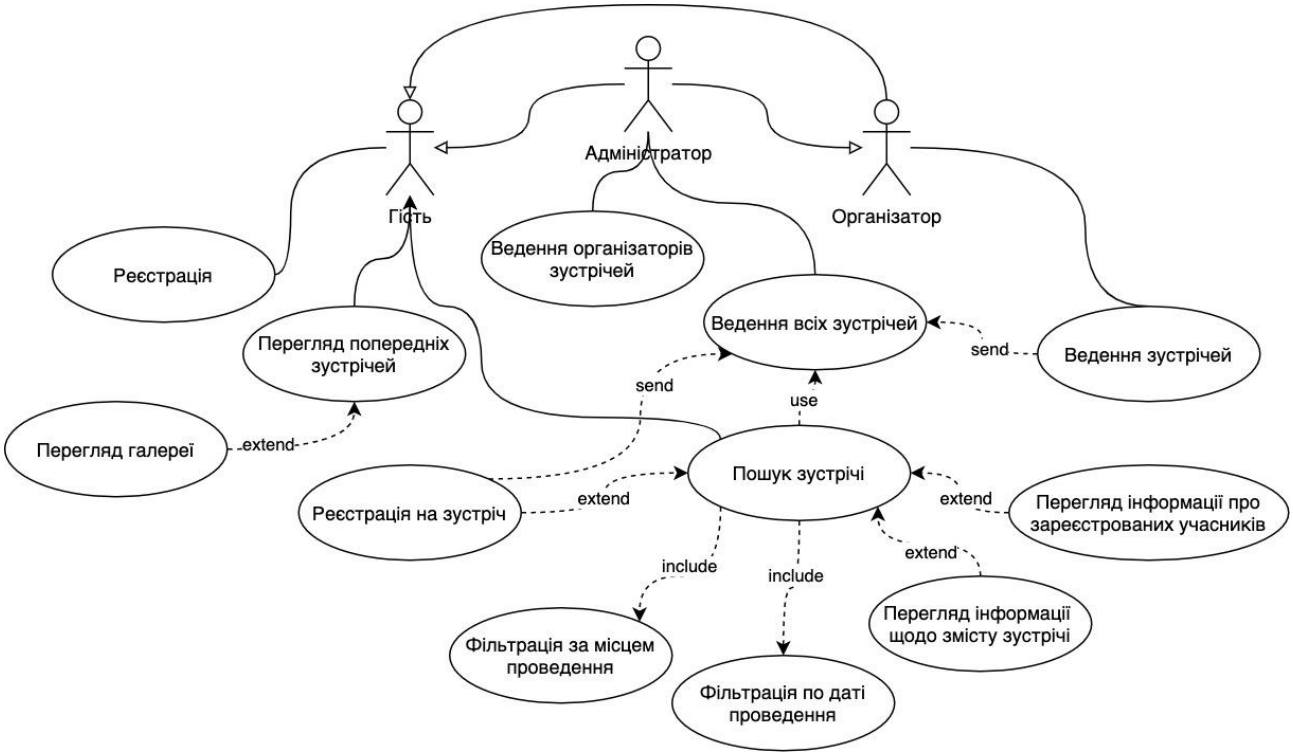


Рисунок 1.6 – Схема структурна варіантів використання

1.4.1 Розроблення функціональних вимог

Згідно схеми варіантів використання було сформовано ряд функціональних вимог які було детально описано та зображено у наступних таблицях:

Таблиця 1.10 – Опис функціональної вимоги RQ-1

Унікальний ідентифікатор	RQ-1
Назва	Реєстрація у системі
Опис	Система повинна надавати незареєстрованому користувачеві можливість зареєструватись у системі заповнивши поля «логін», «пароль» та «пошту» та натиснувши кнопку «Зареєструватись».

Таблиця 1.11 – Опис функціональної вимоги RQ-2

Унікальний ідентифікатор	RQ-2

Продовження таблиці 1.11

Назва	Авторизація у системі
Опис	Система повинна надавати неавторизованому користувачеві можливість авторизуватись у системі заповнивши поля «логін» та «пароль» та натиснувши кнопку «Авторизуватися».

Таблиця 1.12 – Опис функціональної вимоги RQ-3

Унікальний ідентифікатор	RQ-3
Назва	Пошук зустрічей
Опис	Система повинна надавати користувачеві можливість знаходити зустрічі за введеними критеріями, а саме за «місцем проведення», та «датою».

Таблиця 1.13 – Опис функціональної вимоги RQ-4

Унікальний ідентифікатор	RQ-4
Назва	Бронювання місця
Опис	Система повинна надавати користувачеві можливість забронювати місце на обраній зустрічі натиснувши на вільне місце за столом.

Таблиця 1.14 – Опис функціональної вимоги RQ-5

Унікальний ідентифікатор	RQ-5
Назва	Створення зустрічі
Опис	Система повинна надавати можливість створювати нові зустрічі заповнивши наступні поля: «назва», «мета зустрічі», «дату проведення», «місце проведення», «амбасадори», «спікери», «план подій»

Таблиця 1.15 – Опис функціональної вимоги RQ-6

Унікальний ідентифікатор	RQ-6
Назва	Видалення зустрічі
Опис	Система повинна надавати адміністратору можливість видалити будьяку наявну у системі зустріч та надати амбасадору можливість видалити тільки ту зустріч, яку він створив власноруч.

Таблиця 1.16 – Опис функціональної вимоги RQ-7

Унікальний ідентифікатор	RQ-7
Назва	Оновлення інформації про зустріч
Опис	Система повинна надавати адміністратору можливість оновити інформацію про будьяку наявну у системі зустріч та надати амбасадору можливість оновити інформацію тільки про ту зустріч, яку він створив власноруч.

Таблиця 1.17 – Опис функціональної вимоги RQ-8

Унікальний ідентифікатор	RQ-8
Назва	Створення амбасадорів
Опис	Система повинна надавати адміністратору можливість створити амбасадора заповнивши поля «логін», «пароль» та «пошта».

Таблиця 1.18 – Опис функціональної вимоги RQ-9

Унікальний ідентифікатор	RQ-9
Назва	Видалення амбасадорів
Опис	Система повинна надавати адміністратору можливість видаляти амбасадора натиснувши на

Продовження таблиці 1.18

Опис	нього у списку наявних амбасадорів та натиснувши кнопку «видалити».

Згідно функціональних вимог було побудовано матрицю трасування, з якою можна ознайомитися на рисунку 1.7.

Варіанти використання / Вимога	RQ-1	RQ-2	RQ-3	RQ-4	RQ-5	RQ-6	RQ-7	RQ-8	RQ-9
UC-1	■								
UC-2		■							
UC-3			■						
UC-4				■					
UC-5					■				
UC-6						■			
UC-7							■		
UC-8								■	
UC-9									■

Рисунок 1.7 - Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Даний сервіс обробляє надану інформацію та зберігає її у сховищі даних на сервері.

Розроблювана система є веб-сервісом, що надає можливість адміністрування офлайн зустрічей у сфері IT. Система побудована на основі клієнт-серверної архітектури.

Клієнтська частина потребує наявності одного з перечислених веб-браузерів: Google Chrome, Internet Explorer, Safari, Opera Browser.

Сервіс повинен надавати інтуїтивно зрозумілий інтерфейс користувача, швидку навігацію, елементи керування повинні бути однозначними і зрозумілими. Також застосування має забезпечувати високу продуктивність.

1.4.3 Постановка завдання

Виходячи з вище зазначених функціональних та нефункціональних вимог можна встанови комплекс завдань, які повинен вирішувати розроблюваний сервіс.

Метою створення данного сервісу є:

– спрощення процесів адміністрування ІТ-зустрічей;
– покращення комунікації між спеціалістами різних рівней;
– полегшення процесів масштабування великих спільнот.

1.5 Висновки по розділу

У цьому розділі була виконана змістовна постановка задачі, освітлені основні проблеми та існуючі методи реалізації. Розглянуто успішні ІТ-проєкти, що виконують схожий набір задач та проведено порівняння створеного сервісу. А також були описані функціональні та нефункціональні вимоги. Згідно чого був розроблений комплекс завдань модулю.

2 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Сьогодні для створення програмного забезпечення є досить важливим попереднє моделювання його архітектури і бізнес процесів. Для детального проєктування бізнес процесів було використано BPMN нотацію. Модель містить основні етапи, які проходить користувач під час використання сервісу, а саме: реєстрацію, авторизацію, перегляд інформації про зустрічі, пошук зустрічей за обраними критеріями та бронювання місць.

Клієнт вказує свої логін та пароль після чого натискає кнопку авторизації. Настпним кроком дані відсилаються на сервер. Сервер в свою чергу звертається до бази даних з запитом на отримання даних користувача за логіном, далі відбувається обчислення хеш-коду від введеного користувачем паролю та порівння його з хеш-кодом паролю, що зберігається в базі даних. Якщо хеш-коди співпадають користувач успішно проходить авторизацію. В іншому випадку формується повідомлення про помилку, яке відображається в інтерфейсі користувача. Схема бізнес-процесу авторизації наведена на рисунку 2.1.

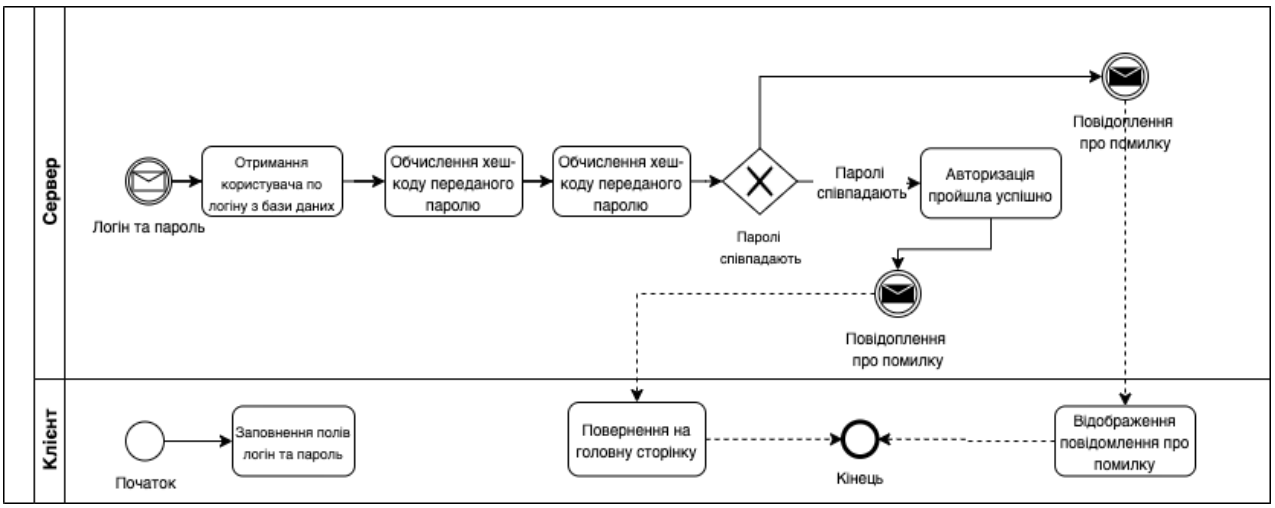


Рисунок 2.1 - Схема бізнес процесу авторизації

Користувач натискає на обране ним місце після чого на сервер відсилається запит на бронювання місця. Сервер перевіряє токен користувача, у випадку, якщо користувач авторизований в базу даних додається запис про реєстрацію користувача на обрану ним зустріч, інакше сервер повертає повідомлення, що користувач не авторизований. В інтерфейсі користувача з'являється форма авторизації, де він повинен вказати свої пароль та логін, після чого у випадку успішної авторизації процедура бронювання повторюється. Схема бізнес-процесу бронювання місця зображена на рисунку 2.2.

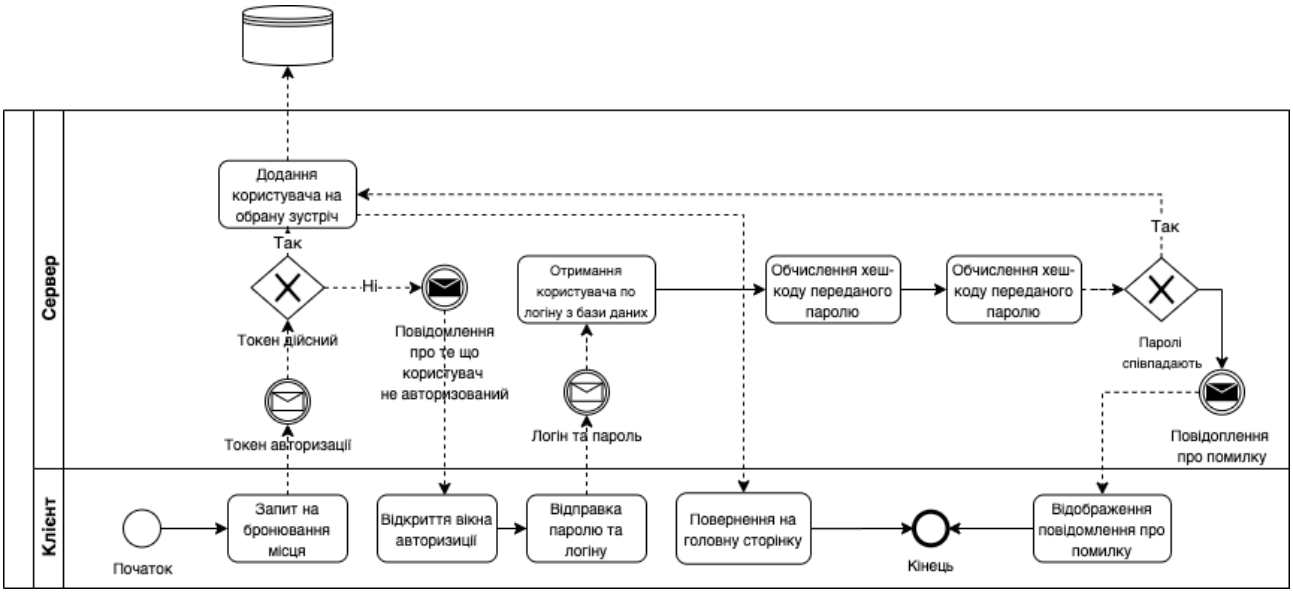


Рисунок 2.2 - Схема бізнес-процесу бронювання місця

Адміністратор на сторінці створення нової зустрічі заповнює всі необхідні поля та натискає кнопку «створити». Повідомлення з усією інформацією відсилається на серевер де перевіряється на унікальність. У випадку, якщо дані зутрічі унікальні то вона додається до бази даних та користувач отримує повідомлення про успішне створення нової зустрічі. У іншому випадку, якщо в базі даних така зустріч вже існує, користувач отримує повідомлення про те що така зустріч вже присутнся у системі. Схема бізнес-процесу додавання нової зустрічі зображена на рисунку 2.3.

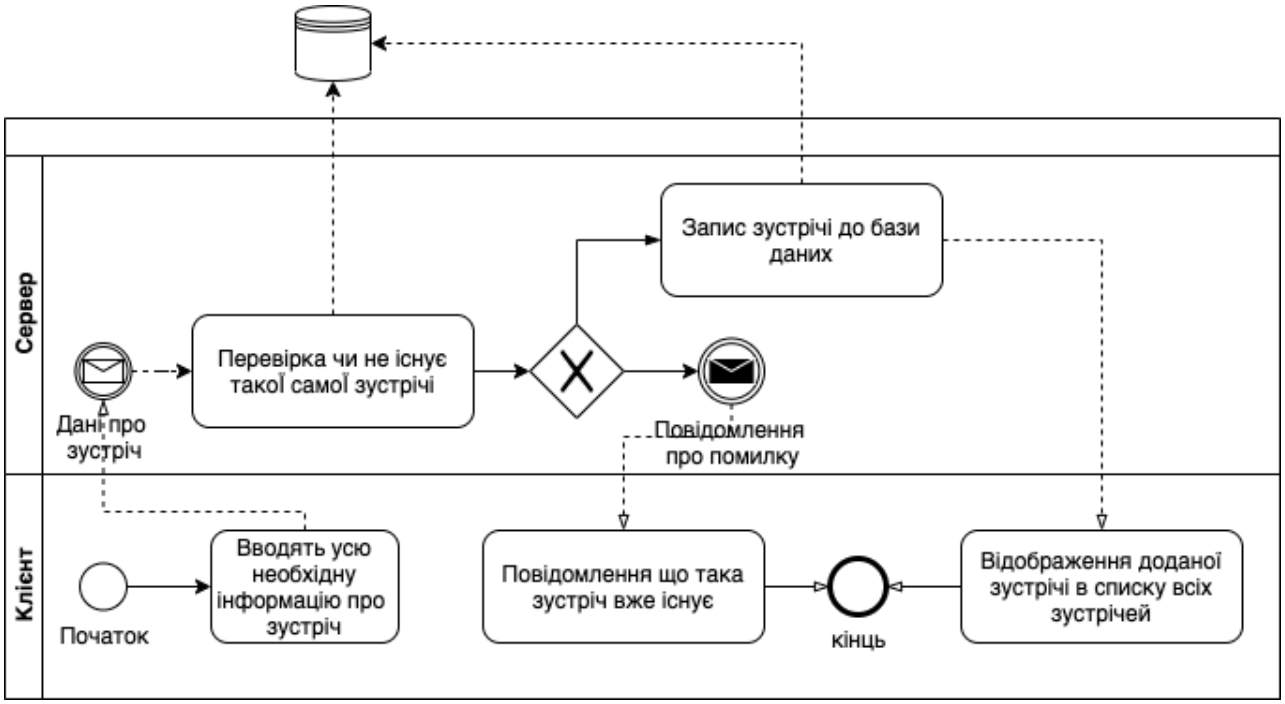


Рисунок 2.3 - Схема бізнес процесу додавання нової зустрічі

2.2 Архітектура програмного забезпечення

Для розробки клієнтської та серверної частини було використано мову JavaScript завдяки її властивостям:

– пряме підключення скриптів до HTML коду;

– можливість запуску програм в браузері і на сервері;

– широкий вибір корисних функціональних параметрів.

React.js – бібліотека, яка використовується для оптимізації роботи з DOM. Дозволяє створювати великі веб-застосунки, які використовують дані, котрі змінюються без перезавантаження сторінки. Її переваги:

– швидка у роботі;

– проста в інтеграції;

– легко масштабована.

Apollo - це бібліотека – клієнт для взаємодії з GraphQL. Завдяки їй стає можливим використання гнучких запитів до API.

У ролі сховища данних використовується MongoDB – не реляційна база даних. Зв’язок з якою відбувається за допомогою Express. Усі зв’язки зображенні у рисунку 2.4.

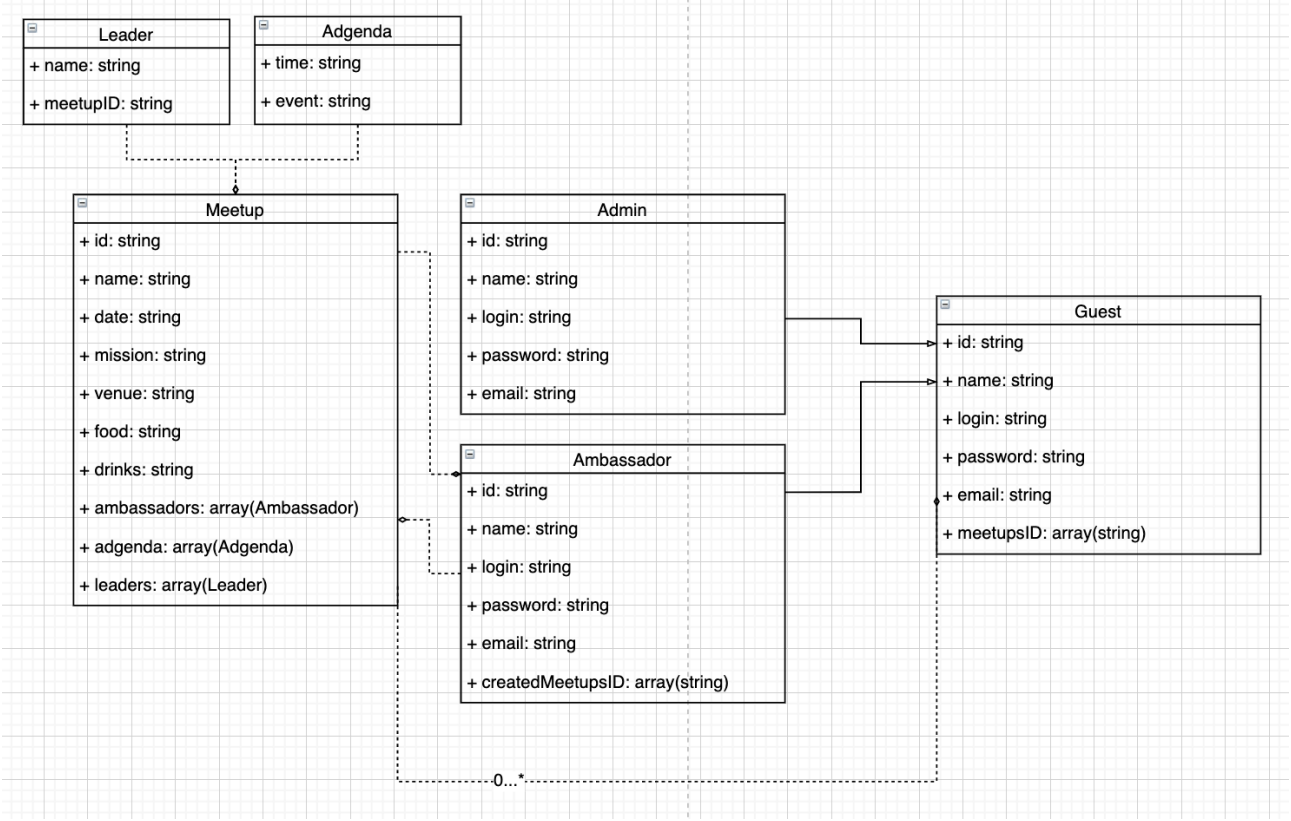


Рисунок 2.4 - Схема бази даних

Таблиця 2.1 - Опис об’єкту Admin

Поле	Тип	Опис
name	string	Ім’я адміністратору
login	string	Логін адміністратору
password	string	Пароль адміністратору
email	string	Пошта адміністратору

Таблиця 2.2 - Опис об’єкту Ambassador

Поле	Тип	Опис
name	string	Ім’я амбасадору
login	string	Логін амбасадору
password	string	Пароль амбасадору
email	string	Пошта амбасадору
createdMeetupID	array<String>	Масив ідентифікаторів створених зустрічей

Таблиця 2.3 - Опис об’єкту Guest

Поле	Тип	Опис
name	string	Ім’я адміністратору
login	string	Логін адміністратору
password	string	Пароль адміністратору
email	string	Пошта адміністратору
meetupsID	array<String>	Ідентифікатори зустрічей на які зареєстрований користувач

Таблиця 2.4 - Опис об’єкту Leader

Поле	Тип	Опис
name	string	Ім’я адміністратору
meetupID	string	Ідентифікатори зустрічі

Таблиця 2.5 - Опис об’єкту Adgenda

Поле	Тип	Опис
time	string	Час події
event	string	Назва події

Таблиця 2.6 - Опис об’єкту Meetup

Поле	Тип	Опис
name	string	Назва зустрічі
date	string	Дата проведення зустрічі
mission	string	Мета зустрічі
venue	string	Місце проведення
food	string	Страви, які буде присутня на зустрічі
drinks	string	Назви напоїв, які будуть присутні на зустрічі
ambassadors	array<Ambassador>	Масив амбасадорів зустрічі
adgenda	array<Adgenda>	Масив подій, які будуть на зустрічі
leaders	array<Leader>	Масив спікерів зустрічі

Redux – бібліотека яка допомагає керувати станом додатку на стороні клієнта. Вона значно полегшує роботу з даними, які зазвичай необхідно

просувати з одного компонента в інший, іноді через усе дерево компонентів. Redux легко справляється зі складними взаємодіями станів, які іноді важко передати за допомогою стану компонента React. По суті, це система передачі даних, яка зустрічається і в об'єктно-орієнтованому програмуванні, але вона не вбудована безпосередньо в javascript, а реалізована у вигляді бібліотеки. Подібно ООП, Redux переводить контроль від компонента до одержувача - інтерфейс не керує станом безпосередньо, а передає йому повідомлення для обробки. Схема архітектури Redux наведена на рисунку 2.5.

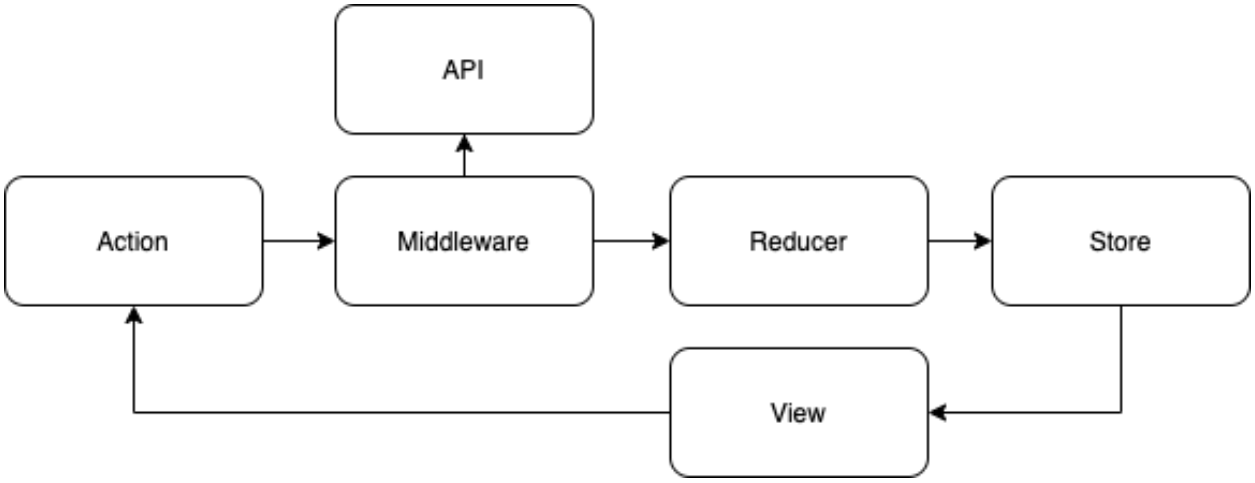


Рисунок 2.5 - Схема архітектури Redux

Архітектура клієнтського додатку була розроблена з урахувнням зручності подальшої підтримки сервісу. Тому для розробки було обрану архітектуру в якій кожен компонент містить у собі важливі для нього файли та інші компоненти які є його не від’ємною частиною. Зображення такого підходу можна побачити на рисунку 2.6.

```
my-app
├── build
├── node_modules
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
├── src
│   ├── components
│   │   ├── app
│   │   │   ├── app.css
│   │   │   ├── app.js
│   │   │   └── app.test.js
│   │   └── index.js
│   ├── images
│   │   └── logo.svg
│   ├── packages
│   │   └── ...
│   ├── utils
│   │   ├── ...
│   │   └── index.js
│   ├── index.css
│   ├── index.js
│   └── service-worker.js
├── .gitignore
├── package.json
└── README.md
```

Рисунок 2.6 - Приклад архітектури проєкту

В ході розробки архітектури веб-сервісу була побудована діаграма класів. Представлення діаграми класів можна побачити на рисунку 2.7.

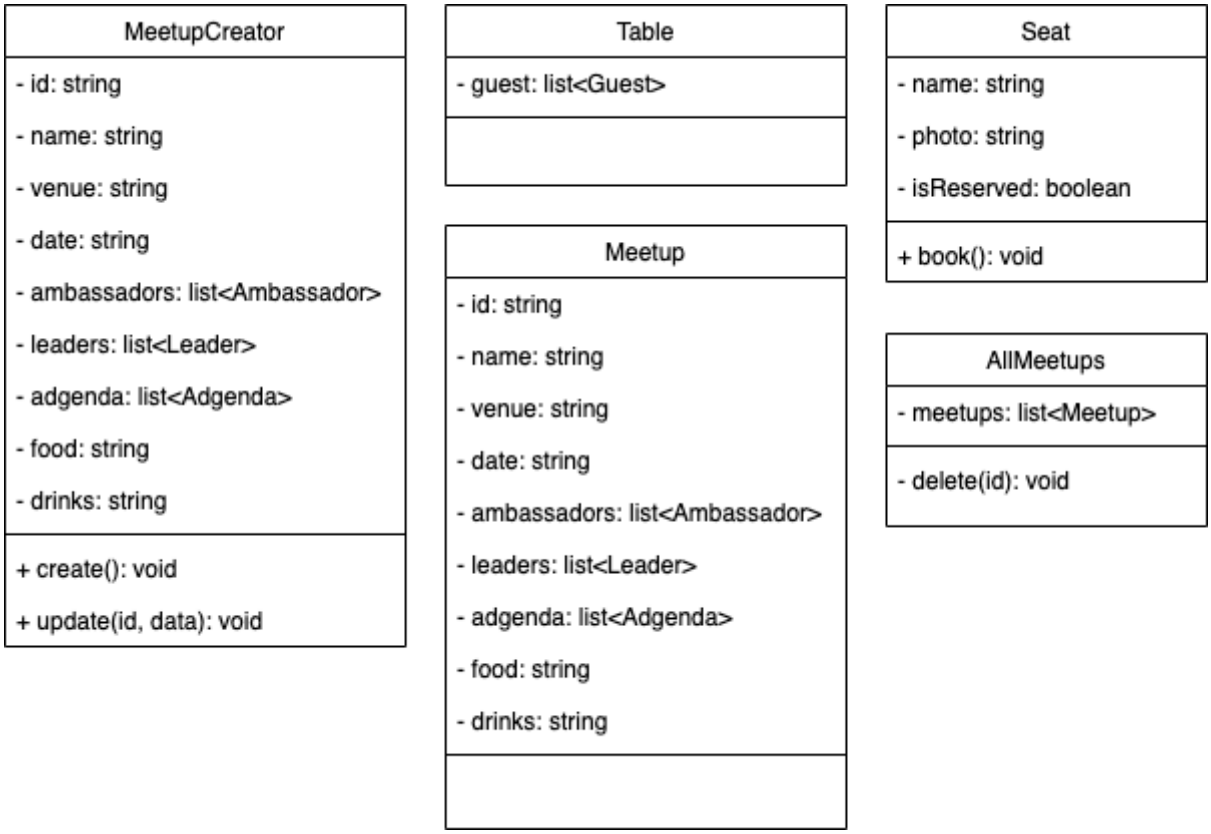


Рисунок 2.7 – Схема структурна класів

Детальний опис класів прогрманого продукту та їх функцій можна побачити на таблицях 2.7 та 2.8.

Таблиця 2.7 - Опис класів

Клас	Опис
Meetup	Клас який мітстить всю головну інформацію про зустріч
Table	Проміжний клас для відображення вільних та вже забронюваних місць
Seat	Клас для відображення місця на зустрічі та бронювання.
MeetupCreator	Клас для створення нових зустрічей та оновлення вже існуючих
AllMeetups	Клас для відображення усіх існуючих в системі зустрічей та видалення їх видалення.

Таблиця 2.8 - Опис методів класів

Клас	Метод	Опис
MeetupCreator	create	Метод для створення нових зустрічей. Вхідні параметри відсутні.
MeetupCreator	update	Метод для оновлення. Вхідним параметром є ідентифікатор оновлюваної зустрічі.
AllMeetups	delete	Метод для видалення зустрічі. Вхідним параметром є ідентифікатор зустрічі.
Seat	book	Метод для бронювання місця на зустрічі. Вхідними параметрами є ідентифікатор зустрічі та користувача.

Для побудови інтерфесу веб-сервісу було використано деревовидну структуру компонентів яку можна побачити на риснку 2.8.

Рисунок 2.8 - Дерево компонентів

2.3 Аналіз безпеки даних

Для забезпечення безпечного обміну пакетами сервіс використовує JWT. Завдяки такому підходу користувачеві не потрібно передавати інформацію у відкритому вигляді, що дозволяє значно підвищити ступінь захищеності. Технологія JSON Web Token широко використовується у багатьох застосунках, які потребують постійного обміну інформацією між клієнтом та сервером.

Перша частина JWT — рядок, у якому закодований звичайний JavaScript об'єкт, містить у собі токен, та назву використаного алгоритму хешування.

Друга частина зберігає основне тіло токена. Довжина корисного навантаження прямопропорційна кількості збереженої інформації.

Кінцева частина JWT — підпис, який генерується на основі заголовка та корисного навантаження. Саме він використовується для перевірки аунтифікованих користувачів.

HTML5 надає можливість перевірки більшості призначених для користувача даних без використання скриптів. Це робиться за допомогою атрибутів перевірки елементів форми, які дозволяють вказувати правила заповнення форми, наприклад, чи потрібне значення, мінімальна і максимальна довжина даних, адреса електронної пошти, чи повинно це бути число, адреса або щось ще, і шаблон, якому це має відповідати. У випадку коли заповнені дані відповідають обраним критеріям, дані вважаються валідними.

Коли елемент валідний:

– елемент відповідає CSS псевдо-класу «valid», що дозволяє застосувати конкретний стиль до цього елементу;

– якщо користувач намагається відправити дані, браузер це зробить, якщо форма пуста, зупинить відправку.

Якщо елемент невалідний:

– елемент відповідає CSS псевдо-класу «invalid» це дозволяє застосувати конкретний стиль до невалідного елементу;

– якщо користувач намагається відправити дані, браузер затримає форму і видасть повідомлення про помилку.

В додатку також застосовується функція перевірки за патерном, яка очікує Regular Expression в якості значення. Регулярний вираз - це шаблон який застосовується для співставлення комбінацій символів в текстових рядках, тому він ідеально підходить для запобігання відправки не валідних даних (а також для деяких інших цілей в JavaScript).

2.4 Висновки по розділу

У цьому розділі було розглянуто основні технології та архітектурні рішення, що застосовуються в розробленій системі. З’ясувавши всі аспекти,

маємо чітке уявлення того, як саме спроєктований додаток та чому були обрані саме такі технології. Було наведео діаграму класів, дерево компонентів та схеми архітектур деяких рішень. Проаналізовано безпечність сервісу та за допомогою BPMN-діаграм зображено бізнес-процеси додатку.

3 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Якість програмного забезпечення — полягає у ступені його відповідності до вимог замовника. Для того щоб визначити цей ступінь необхідно провести тестування. Базовими характеристиками для оцінки якості є: надійність, швидкодія, безпека, зручність використання та масштабованість.

Формально, надійність - це властивість додатку зберігати у встановлених межах значення всіх параметрів, що характеризують здатність виконувати необхідні функції в заданих режимах і умовах застосування. Тобто вимоги до надійності додатки визначаються умовами функціонування програми (параметри сервера, максимальна кількість користувачів додатка) і допустимими показниками якості роботи системи в цих умовах (час обробки запиту користувача до системі, кількість відмов системи). Таким чином, надійне веб-додаток має забезпечувати доступ до всіх функцій для користувача при будь-яких умовах

Швидкодія програми визначається як середній час обробки запитів користувача до системи. Максимально допустимим часом відгуку для веб-додатків вважається 5 секунд.

Безпека веб-додатку включає в себе: розмежування прав доступу до функцій і даних кожного компонента веб-додатку. Контроль рівня доступу користувачів здійснюється шляхом авторизації та верифікації користувачів.

Масштабованість - це здатність системи утримувати свою продуктивність при підвищеному навантаженні і додаванні ресурсів. для користувача масштабованого веб-додатку повинен залишатися непомітним момент, коли зросте навантаження (до додатку намагатимуться отримати доступ одночасно ще кілька користувачів), а також при зміні конфігурації додатка (на рівні бізнес-логіки буде додано компонент обробки даних).

Системи, які будуть протестовані включають в себе браузерний додаток та API, що написане в середовищі NodeJS. Додаток має бути протестований в останній версії Google Chrome (користується 48,06% користувачів у світі), Internet Explorer (19,6%), Mozilla Firefox останньої версії (16,74%), Safari (10,63%). API достатньо протестувати лише у середовищі в якому воно буде працювати після запуску проєкту, що є Ubuntu 18.04 Bionic Beaver LTS, 64 bit.

3.2 Опис процесів тестування

Об’єктами тестування є:

- сервер обробки даних;
- інтерфейс користувача.

Функції які повинні бути протестовані з точки зору користувача:

- реєстрація користувача;
- авторизація користувача;
- бронювання місця на зустрічі;
- пошук зустрічей у системі.

Функції які повинні бути протестовані з точки зору адміністратора:

- створення зустрічі;
- видалення будь’якої зустрічі;
- оновлення будь’якої зустрічі;
- створення амбасадору;
- видалення амбасадору.

Функції які повинні бути протестовані з точки зору амбасадора:

- створення зустрічі;
- видалення самостійно створеної зустрічі;
- оновлення самостійно створеної зустрічі.

Оскільки нам відома внутрішня будова додатку тестування проходитиме у режимі білої скриньки. Для отримання більш цілісної картини будуть проведені як позитивні, так і негативни тести.

На цьому етапі будуть проводитись наступні функціональні типи тестів:

– тестування реєстрації користувача у системі;
– тестування авторизації користувача у системі;
– димне тестування додавання нових зустрічей;
– димне тестування видалення зустрічі з бази даних;
– димне тестування створення амбасадору;
– димне тестування видалення амбасадору;
– димне тестування бронювання місця.
– авторизація користувача.

Для цього додатку не будуть проводитись нефункціональні типи тестів, оскільки вони були проведені на етапі формування дизайну застосування.

3.3 Опис контрольного прикладу

В ході тестування було перевірено всю функціональність підсистеми. Послідовно були перевірені всі варіанти використання, отримані зображені у наступних таблицях:

Таблиця 3.1 – Перевірка реєстрації користувача

Мета тесту	Перевірка реєстрації користувача
Передумови	Користувача з такою поштою та логіном не зареєстровано у системі
Схема виконання	Заповнити усі поля форми валіднними даними, а саме вказати пошту, логін та пароль після чого натиснути кнопку «зареєструватись»
Очікуваний результат	Користувача було зареєстровано у системі
Дійсний результат	Користувача було зареєстровано у системі

Таблиця 3.2 – Перевірка авторизації користувача

Мета тесту	Перевірка авторизації користувача

Продовження таблиці 3.2

Передумови	Користувач з такою поштою та логіном зареєстрований у системі
Схема виконання	Заповнити усі поля форми валіндними даними, а саме вказати логін та пароль після чого натиснути кнопку «авторизуватись»
Очікуваний результат	Користувач був авторизований у системі
Дійсний результат	Користувач був авторизований у системі

Таблиця 3.3 – Перевірка бронювання місця

Мета тесту	Перевірка бронювання місця користувачем
Передумови	1. Авторизований користувач 2. Наявність у системі хоча б однієї зустрічі 3. Наявність вільного місця
Схема виконання	Користувач натискає на вільне місце за столом після чого система додає його до списку гостей цієї зустрічі
Очікуваний результат	Користувача додано до списку гостей та на обраному місці з’вилось його зображення
Дійсний результат	Користувача додано до списку гостей та на обраному місці з’вилось його зображення

Таблиця 3.4 – Перевірка пошуку зустрічей

Мета тесту	Перевірка пошуку зустрічей
Передумови	В системі присутня зустріч яка проходить у місті Київ та відбудеться 10.10.2020
Схема виконання	Користувач змінює критерії пошуку за місцем на Київ та обирає дату проведення 10.10.2020 після чого система автоматично виводить

Продовження таблиці 3.4

Схема виконання	знайдені зустрічі.
Очікуваний результат	У списку знайдених зустрічей присутня зустріч яка відбувається у місті Київ та відбудеться 10.10.2020
Дійсний результат	У списку знайдених зустрічей присутня зустріч яка відбувається у місті Київ та відбудеться 10.10.2020

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання серверної частини веб-сервісу треба встановити Node.js та MongoDB. У ролі серверу можна обрати сервери сервісу Heroku. Схема розгортання компонентів системи представлена на рисунку 4.1 – Схема структурна розгортання компонентів системи.

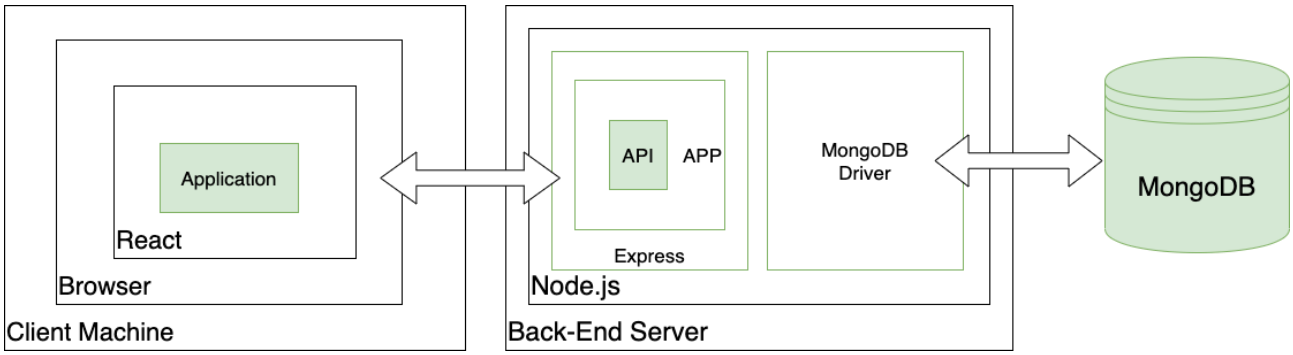


Рисунок 4.1 - Схема структурна розгортання компонентів системи

Для розгортання серверної частини веб-сервісу треба встановити Node.js та MongoDB. У ролі серверу можна обрати сервери сервісу Heroku.

Для розгортання клієнтської частини веб-сервісу обов’язковою умовою є наявність веб-браузеру.

4.2 Робота з програмним забезпеченням

Детальна інструкція користувача наведена у Керівництві користувача.

ВИСНОВКИ

При створенні даного дипломного проекту досить детально було проаналізовано предметну область, а також розглянуті вже відомі технічні рішення та найбільш відомі аналоги. Після аналізу предметної області та існуючих аналогів було з’ясовано, що потреба в розробкі веб-сервісу дійсно присутня. Також були описані головні вимоги до застосунку та функціоналу. Було спроєктовано архітектуру основану на компонентному підході, що є легко масштабованою. Для розробки додатку використовувалася мова програмування JavaScript. У ролі бази даних була використана не реляційна база даних MongoDB.

Також у ході створення додатку було проведено його димне тестування та доведено, що якість відповідає вимогам які були описані на початку роботи. Створене веб-застосування полегшує організаційні процеси та надає нові можливості для масштабування. Дане програмне забезпечення може використовуватись для будь-якої організації яка займається проведенням власних тематичних зучтрічей офлайн.

ПЕРЕЛІК ПОСИЛАНЬ

1) Калініченко В.С. ОПТИМІЗАЦІЯ РОБОТИ З DOM ШЛЯХОМ ІНТЕГРАЦІЇ JAVASCRIPT-ОБ'ЄКТА (VDOM) // 4 Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління – ІСТУ-2020» Матеріали конференції. – 2020. – С.

2) BPMN [Електронний ресурс]. – 2020. – Режим доступу доступу до ресурсу: https://uk.wikipadia.org/wiki/BPMN.

3) React.js - Документація [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://ru.reactjs.org.

4) Redux.js – Документація [Електронний ресурс]. – 2020 – Режим доступу до ресурсу: https://redux.js.org.

5) MongoDB documentation [Електронний ресурс] // MongoDB. – 2019. – Режим доступу до ресурсу: https://docs.mongodb.com/.

6) GitLab documentation [Електронний ресурс] // GitLab. – 2020. – Режим доступу до ресурсу: https://docs.gitlab.com/.

7) JWT documentation [Електронний ресурс] // JWT. – 2020. – Режим доступу до ресурсу: https://jwt.io/.

8) Meetup.com [Електронний ресурс] // Meetup. – 2020. – Режим доступу до ресурсу: https://www.meetup.com/.

9) Diobox [Електронний ресурс] // Diobox. – 2020. – Режим доступу до ресурсу: https://home.d.io/.

10) Heroku documentation [Електронний ресурс] // Heroku. – 2020. – Режим доступу до ресурсу: https://www.heroku.com/.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-СЕРВІС АДМІНІСТРУВАННЯ ІТ-ЗУСТРІЧЕЙ

Технічне завдання

КПІ.ІП-6313.045440.03.91

“ПОГОДЖЕНО”

Керівник проекту:

О.А. Халус

Нормоконтроль:

К.І. Ліщук

Виконавець:

В.С. Калініченко

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-сервіс адміністрування ІТ-зустрічей

Галузь застосування: Проекти, що працюють у організації заходів

Наведене технічне завдання поширюється на розробку веб-сервісу адміністрування ІТ-зустрічей КПІ.ІП-6313.045440.03.91, котра використовується для полегшення роботи власників великих клубів, які мають свої філії в різних куточках планети, шляхом делегування основних організаційних процесів довіреним особам. Кожен з амбасадорів може створити зустріч надавши сервісу достатньо інформації про спікерів, тематику зустрічі, місце та план проведення.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-сервісу адміністрування ІТ-зустрічей є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Призначенням розробки надання актуальної інформації про ІТ-зустрічі в різних куточках світу, які закріплені за одним клубом. Також сервіс надає можливість забронювати місце на обраній зустрічі.

Цей проект націлений на покращення процесів масштабування шляхом делегування організаційних процессів на довірених осіб, які можуть знаходитись у будьякому куточку світу.

Сервіс має багатоцільове призначення, але головним є – покращення економічної ситуації на ринку ІТ шляхом поширення коммунікації між розробниками різних рівнів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

- реєстрація у системі;
- авторизація у системі;
- отримання інформації про зустріч;
- пошук зустрічей за місцем проведення;
- пошук зустрічей за датою проведення;
- бронювання місця на обраній зустрічі.

4.1.1.2 Для адміністратора:

- авторизація у системі;
- створення нових зустрічей;
- редагування вже створених зустрчіей;
- видалення зустрічей;
- створення амбасадорів;
- видалення амбасадорів.

4.1.1.3 Для амбасадору:

- авторизація у системі;
- створення нових зустрічей;
- редагування вже створених зустрічей;
- видалення зустрічей.

4.1.2 Серверну частину додатку виконати на платформі Linux.

4.1.3 Додаткові вимоги: відсутні

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування.

Не висуваються.

4.3.3 Обслуговуючий персонал.

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на серверах та комп'ютерах з операційною системою Windows/Ubuntu/MacOS.

4.4.2 Мінімальна конфігурація технічних засобів:

– тип процесору ……. Pentium;
– об'єм ОЗП ……..….. 512 Мб;
– об'єм вільного дискового простору ……. 512 Мб.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем Windows 10/Ubuntu/MacOS.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: HTTP запити.

4.5.3 Результати повинні бути представлені в наступному форматі: веб-сторінка.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи.

5.3.1 Пояснювальна записка не менше ніж на 100 аркушах формату А4 (без додатків 5.3.2 - 5.3.4).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування.

5.4 Графічна частина повинна бути виконана у формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структурна варіантів використань.

5.4.2 Схема структурна класів програмного забезпечення.

5.4.3 Креслення вигляду екранних форм.

5.4.4 Схема бази даних.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Таблиця 6.1 – Стадії і етапи розробки

№	Назва етапу	Строк	Звітність
1	Вивчення рекомендованої літератури	19.03.2020	
2	Аналіз існуючих методів розв’язання задачі	26.03.2020	
3	Постановка та формалізація задачі	26.03.2020	Технічне завдання
4	Аналіз вимог до програмного забезпечення	02.04.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5	Алгоритмізація задачі	02.04.2020	
6	Моделювання програмного забезпечення	09.04.2020	
7	Обґрунтування використовуваних технічних засобів	16.04.2020	
8	Розробка архітектури програмного забезпечення	23.04.2020	Схема структурна класів програмного забезпечення
9	Розробка програмного забезпечення	30.04.2020	Тексти програмного забезпечення
10	Налагодження програми	07.05.2020	Програма та методика тестування
11	Виконання графічних документів	14.05.2020	Графічний матеріал проекту

Продовження таблиці 6.1

12	Оформлення пояснювальної записки	21.05.2020	Пояснювальна записка проекту
13	Подання ДП на попередній захист	06.06.2020	
14	Подання ДП рецензенту	07.06.2020	
15	Подання ДП на основний захист	19.06.2020	

7 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

7.1. Види випробувань

Тестування розробленого програминого продукту виконується відповідно до “Програми та методикии тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-СЕРВІС АДМІНІСТРУВАННЯ ІТ-ЗУСТРІЧЕЙ

Програма та методика тестування

КПІ.ІП-6313.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.А. Халус

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С. Калініченко

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення з аналізу результатів інтеграційних тестів, який являє собою інтерфейс командного рядка, створений з використанням мови програмування Python з використанням ряду зовнішніх бібліотек, таких як python-Levenshtein, re, xml, os, argparse, pymongo.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

– функціональна працездатність елементів сторінок інтерфейсу командного рядка;

– коректне зчитування вхідних параметрів;

– відповідність форматів та протоколів файлів реєстрації тестування;

– наявність доступу до бази даних та коректна робота з нею;

– зручність роботи з інтерфейсом командного рядка;

– відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні модульного тестування та системного тестування.

Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Critical path test (ба-зове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності);

– тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію pytest.

Працездатність інтерфейсу командного рядка перевіряється шляхом:

– динамічного ручного тестування – введенням доступних та недопустимих значень в вхідні параметри;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування коректності обчислень за допомогою підготовних наборів даних;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-СЕРВІС АДМІНІСТРУВАННЯ ІТ-ЗУСТРІЧЕЙ

Керівництво користувача

КПІ.ІП-6313.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.А. Халус

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С. Калініченко

Київ – 2020 року

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Запуск програмного забезпечення

Для роботи з програмним забезпеченням користувачу не обов’язково бути зареєстрованим і проходити етап авторизації але у випадку коли користувачу необхідно забронювати місце процедура авторизацію є необхідною.

Для запуска клієнтської частини програмного забезпечення необхідно виконати команду «npm start», в командному рядку.

Для запуска серверної частини додатку необхідно виконати команду «nodemon server.js».

1.2 Реєстрація та авторизація користувача

Для реєстрації користувач повинен обрати будьяку зустріч з наданого списку після чого натиснути на вільне місце за столом. У випадку якщо користувач не авторизований у системі йому буде надана можливість зареєєструватися, або авторизуватися. У випадку, коли користувач обрав авторизацію, йому потрібно заповнити небхідні поля, а саме поля «пароль» та «пошта» після чого натиснути на кнопу «авторизуватись». Після успішної авторизації, він автоматично буде зареєстрований на обрану зустріч у ролі гостя. У випадку, якщо користувач обрав реєстрацію він потрібен заповнити поля «Email», «Password» та «Name» . Вигляд форми авторизації наведено на рисунку 1.1.

Рисунок 1.1- Форма авторизації користувача

1.3 Пошук зустрічей

Для пошуку зустрічей користувачу необхідно натиснути на кнопку «меню» на головній сторінці та обрати критерії пошуку. Після кожної зміни критеріїв пошуку, список зустрічей буде оновлюватись відповідно обраним критеріям. Після того я список відповідних зустрічей був відображений користувач може натиснути на обрану зустріч та забронювати місце. Вигляд сторінки при здійсненні пошуку зображено на рисунку 1.2.

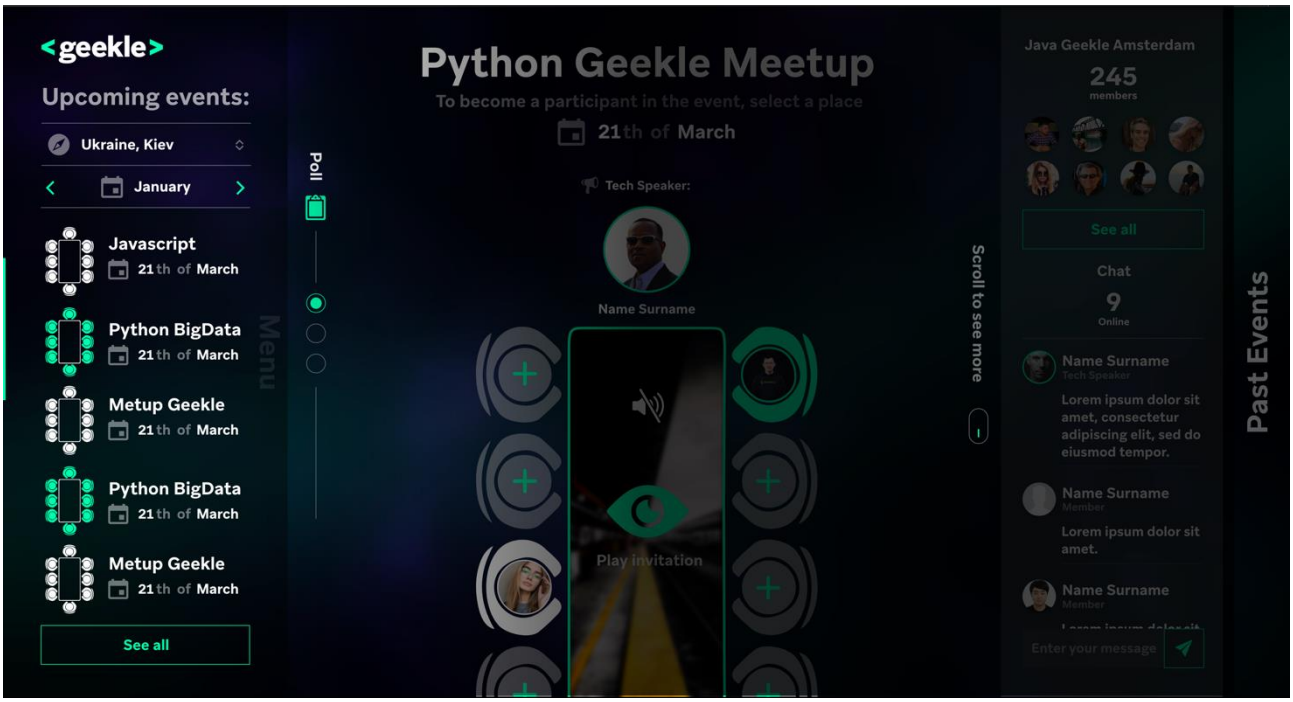


Рисунок 1.2 – Пошку зустрічей

1.4 Перегляд інформації про зустріч

Для перегляду основної інформації про зустріч достатньо натиснути на поле обраної зустрічі зі списку зустрічей у пошуковому меню. Для перегляду додаткової інформації необхідно натиснути на кнопку «More info», яка знаходиться в нижній частині екрану на сторінці обраної зустрічі. Вигляд додаткової інформації можна побачити на рисунку 1.3.

Рисунок 1.3 – Сторінка переглядку додаткової інформації про зустріч

1.5 Створення зустрічей

Адміністратор має можливість створювати зустрічі, для цього необхідно пройти авторизацію в панелі адміністратора, після чого перейти на сторінку «Зустріч» та заповнити всі необхідні поля. Якщо введена вся необхідна інформація адміністратор повинен натиснути кнопку «Create Meetup». У базі даних одразу з'явиться нова зустріч, яку можна буде побачити у списку всіх зустрічей. Вигляд форми для створення зустрічі можна побачити на рисунку 1.4.

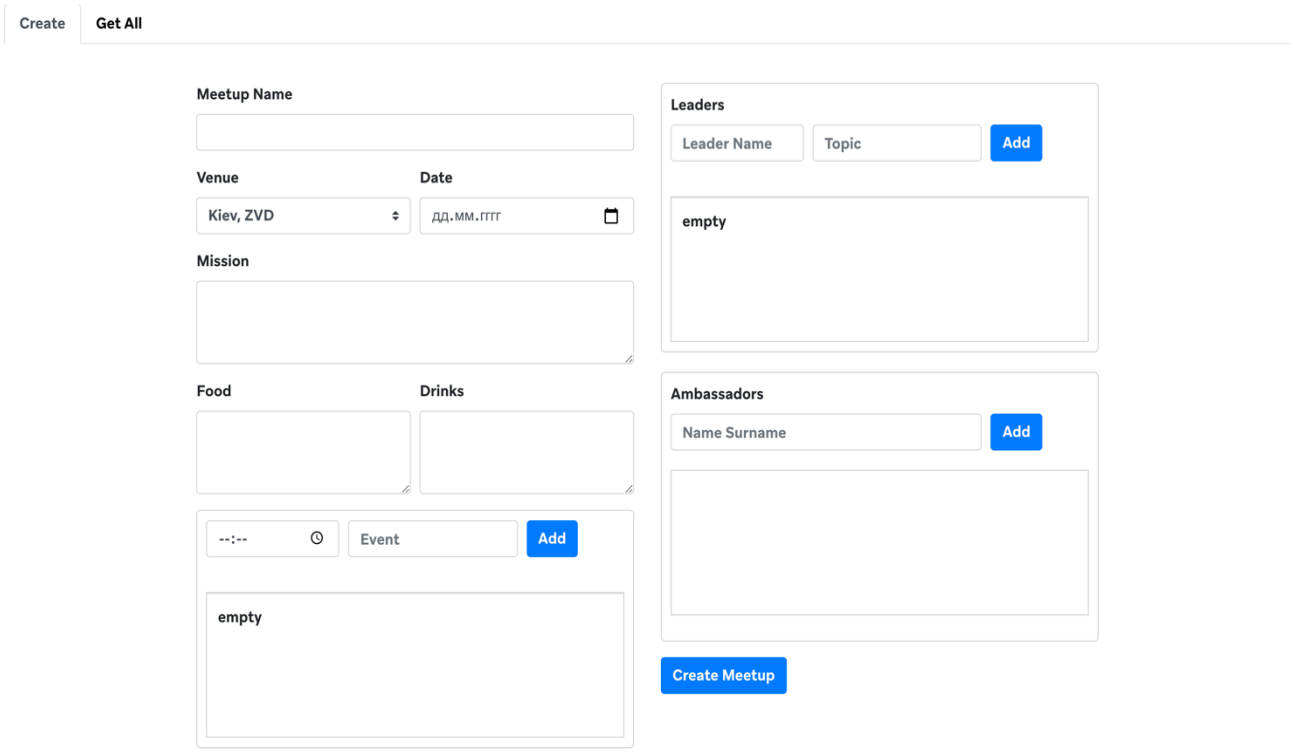


Рисунок 1.4 – Сторінка створення нової зустрічі

1.6 Редагування та видалення зустрічей

Адміністратор має можливість редагувати та видаляти вже наявні зустрічі. Для цього необхідно бути авторизованим у ролі адміністратора або амбасадора та перейти на сторінку «Get All». У списку всіх зустрчіей поряд з кожною зустріччю будуть дві кнопки «Delete» та «Update».

Для оновлення інформації про зустріч, треба натиснути на кнопку «Update» після чого на сторінці редагування оновити потрібні данні та натиснути кнопку «Update Meetup».

Для видалення зустрічі треба натиснути на кнопку «Delete» у полі обраної зустрічі. Після чого ця зустріч зникне з бази даних.

Вигляд сторінки перегляду усіх зустрічей яка містить описані вище можливості можна побачити на рисунку 1.5.

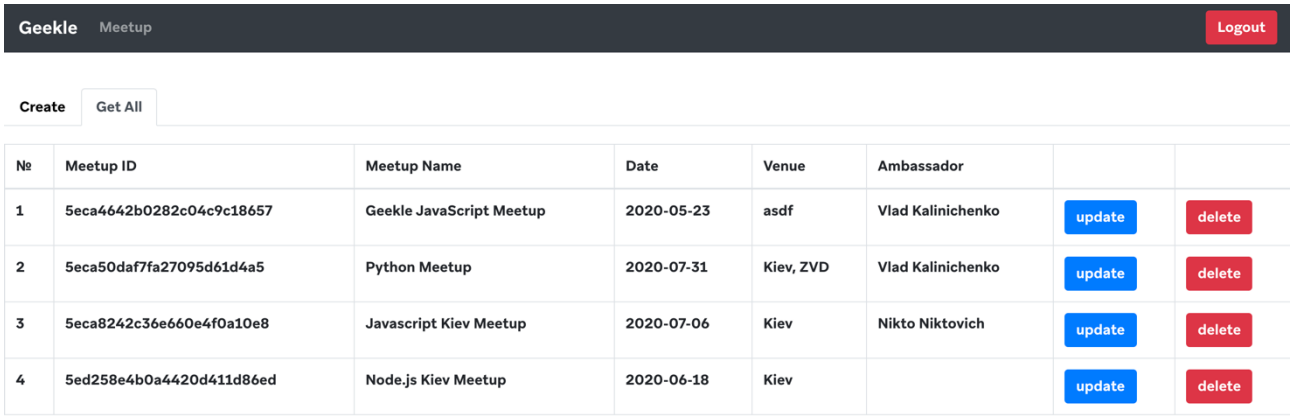


Рисунок 1.5 - Сторінка всіх зустрічей

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-сервіс адміністрування ІТ-зустрічей

Опис програми

КПІ.ІП-6313.045440.06.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.А. Халус

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.С. Калініченко

Київ – 2020 року

Тексти програмного коду

Веб-сервіс адміністрування IT-зустрічей

(Найменування програми (документа))

DVD-R

(Вид носія даних)

17 арк, 55 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020

Schema.js

```
const {
  GraphQLObjectType,
  GraphQLString,
  GraphQLID,
  GraphQLInt,
  GraphQLInputObjectType,
  GraphQLSchema,
  GraphQLList,
  GraphQLNonNull
 } = graphql;

const Meetup = require('../models/meetup');
const Guest = require('../models/guest');


const AdgendaType = new GraphQLObjectType({
  name: `Adgenda`,
  fields: () => ({
    time: { type: new GraphQLNonNull(GraphQLString) },
    event: { type: new GraphQLNonNull(GraphQLString) }
  }),
});

const adgendaInput = new GraphQLInputObjectType({
  name: "AdgendaInput",
  description: "Adgenda",
  fields: () => ({
    time: { type: new GraphQLNonNull(GraphQLString) },
    event: { type: new GraphQLNonNull(GraphQLString) }
  })
```

```
});

const LeaderType = new GraphQLObjectType({
  name: `Leader`,
  fields: () => ({
    id: { type: GraphQLID },
    name: { type: new GraphQLNonNull(GraphQLString) },
    topic: { type: new GraphQLNonNull(GraphQLString) },
    photo: { type: new GraphQLNonNull(GraphQLString) }
  }),
});

const leaderInput = new GraphQLInputObjectType({
  name: "LeaderInput",
  description: "Leader",
  fields: () => ({
    name: { type: new GraphQLNonNull(GraphQLString) },
    topic: { type: new GraphQLNonNull(GraphQLString) },
    photo: { type: new GraphQLNonNull(GraphQLString) }
  })
});

const MeetupType = new GraphQLObjectType({
  name: `Meetup`,
  fields: () => ({
    id: { type: GraphQLID },
    name: { type: new GraphQLNonNull(GraphQLString) },
    date: { type: new GraphQLNonNull(GraphQLString) },
    food: { type: new GraphQLNonNull(GraphQLString) },
    drinks: { type: new GraphQLNonNull(GraphQLString) },
    venue: { type: new GraphQLNonNull(GraphQLString) },
    adgenda: { type: new GraphQLList(AdgendaType) },
```

```
    leaders: { type: new GraphQLList(LeaderType) },
    ambassadors: { type: new GraphQLList(GraphQLString) },
    mission: { type: GraphQLString }
  })
});

const GuestType = new GraphQLObjectType({
  name: `Guest`,
  fields: () => ({
    id: { type: GraphQLID },
    name: { type: new GraphQLNonNull(GraphQLString) },
    email: { type: new GraphQLNonNull(GraphQLString) },
    password: { type: new GraphQLNonNull(GraphQLString) },
    meetups: { type: new GraphQLList(GraphQLID)},
    photo: { type: GraphQLString }
  })
})

const Mutation = new GraphQLObjectType({
  name: `Mutation`,
  fields: {
    addGuest: {
      type: GuestType,
      args: {
        name: { type: new GraphQLNonNull(GraphQLString) },
        email: { type: new GraphQLNonNull(GraphQLString) },
        password: { type: new GraphQLNonNull(GraphQLString) },
        meetups: { type: new GraphQLList(GraphQLID) },
        photo: { type: GraphQLString }
      },
      resolve(parent, args) {
        const guest = new Guest({
```

```
        name: args.name,
        email: args.email,
        password: args.password,
        photo: args.photo
      });
      return guest.save();
    }
  },
  addMeetup: {
    type: MeetupType,
    args: {
      name: { type: new GraphQLNonNull(GraphQLString) },
      date: { type: new GraphQLNonNull(GraphQLString) },
      venue: { type: new GraphQLNonNull(GraphQLString) },
      food: { type: new GraphQLNonNull(GraphQLString) },
      drinks: { type: new GraphQLNonNull(GraphQLString) },
      adgenda: { type: new GraphQLList(adgendaInput) },
      leaders: { type: new GraphQLList(leaderInput) },
      ambassadors: { type: new GraphQLList(GraphQLString) },
      mission: { type: GraphQLString }
    },
    resolve(parent, args) {
      const meetup = new Meetup({
        name: args.name,
        date: args.date,
        venue: args.venue,
        food: args.food,
        drinks: args.drinks,
        ambassadors: args.ambassadors,
        adgenda: args.adgenda,
        leaders: args.leaders,
        mission: args.mission
```

```
    });
    return meetup.save();
   }
  },
  deleteMeetup: {
   type: MeetupType,
   args: {
    id: { type: GraphQLID }
   },
   resolve(parent, args) {
    return Meetup.findByIdAndDelete(args.id);
   }
  },
  updateMeetup: {
   type: MeetupType,
   args: {
    id: { type: new GraphQLNonNull(GraphQLID) },
    name: { type: new GraphQLNonNull(GraphQLString) },
    date: { type: new GraphQLNonNull(GraphQLString) },
    venue: { type: new GraphQLNonNull(GraphQLString) },
    food: { type: new GraphQLNonNull(GraphQLString) },
    drinks: { type: new GraphQLNonNull(GraphQLString) },
    adgenda: { type: new GraphQLList(adgendaInput) },
    leaders: { type: new GraphQLList(leaderInput) },
    ambassadors: { type: new GraphQLList(GraphQLString) },
    mission: { type: GraphQLString }
   },
   async resolve(parent, args) {
    let meetup = await Meetup.findById(args.id);
    meetup.name = args.name;
    meetup.date = args.date;
    meetup.venue = args.venue;
```

```
        meetup.food = args.food;
        meetup.drinks = args.drink
```

App.js

```
import React, { Suspense, useState, useEffect } from 'react';
import { Route, withRouter, useLocation, Redirect } from "react-router-dom";
import Loader from './components/common/Loader/Loader';
import AdgendaContainer from './components/Adgenda/AdgendaContainer';
import Login from './components/common/Login/Login';
import './App.css';
import AdminPanelContainer from './components/AdminPanel/AdminPanelContainer';
import MoreInfo from './components/Meetup/MoreInfo/MoreInfo';
const MeetupContainer = React.lazy(() => import('./components/Meetup/MeetupContainer'));

function App(props) {

  return (
    <div className="app-wrapper">
      <Route exact path='/' render={() => <Redirect to="/meetup/" />} />
      <Route path='/meetup/:id?'
        render={() => <Suspense fallback={<Loader/>}><MeetupContainer /></Suspense>} />

      <Route path='/adgenda/:id'
        render={() => <AdgendaContainer />} />

      <Route path='/admin'
        render={() => <AdminPanelContainer />} />

    </div>
  );
}
```

```
export default  (withRouter)(App);
```

Meetup.js

```
import React, { useState, Suspense, useEffect } from 'react';
import s from './Meetup.module.css';
import SpeakerImg from '../../assets/img/face.jpeg';
import Loader from '../common/Loader/Loader';
import { Route } from 'react-router-dom';
import MoreInfo from './MoreInfo/MoreInfo';
const TableContainer = React.lazy(() => import('./Table/TableContainer'));
const MenuContainer = React.lazy(() => import('../common/Menu/Menu.container'));
const PastEvents = React.lazy(() => import('../common/PastEvents/PastEventsContainer'));
const LeaderBoardContainer = React.lazy(() => import('./LeaderBoard/LeaderBoardContainer'));
const GuestBoardContainer = React.lazy(() => import('./GuestBoard/GuestBoardContainer'));
const MeetupHeader = React.lazy(() => import('./MeetupHeader/MeetupHeader'));

const Meetup = (props) => {

    const months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'];
    const [city, setCity] = useState(props.cities[0]);
    const [isAuthorize, setAuthorize] = useState(false);

    const selectCity = (id) => {
        props.cities.forEach(city => {
            if (city.id === id) {
                setCity(city);
            }
```

```
    })
  }

  useEffect(() => {
    localStorage.setItem('token', '');
  }, []);

  return (
    <div className={s.meetupContainer}>
      <Suspense fallback={<Loader />}>
        <MenuContainer
          selectedCity={city}
          selectCity={selectCity}
          cities={props.cities}
          months={months}
        />
        <PastEvents />
        <Route exact path='/meetup/:id?'
          render={() => <Suspense fallback={<Loader />}>
            <GuestBoardContainer />
            <LeaderBoardContainer loading={props.loading} data={props.data}
ambassador={props.ambassador} />
            <MeetupHeader loading={props.loading} data={props.data}
months={months}/>
            {
              props.data &&
              <TableContainer speakerImg={SpeakerImg}
meetupId={props.data.meetup.id} loading={props.loading} refetch={props.refetch}/>
            }
          </Suspense>} />
        <Route path='/meetup/more-info/:id?'
          render={() => <Suspense fallback={<Loader />}>
```

```
                    <MoreInfo />
                </Suspense>} />
            </Suspense>
        </div>
    )
}

export default Meetup;
```

MeetupCreator.js

```
import React, { useState, useEffect } from 'react';
import s from './MeetupCreator.module.css';

const MeetupCreator = (props) => {

    const [name, setName] = useState({ value: '', error: false });
    const [venue, setVenue] = useState('');
    const [date, setDate] = useState('');
    const [mission, setMission] = useState('');
    const [food, setFood] = useState('');
    const [drinks, setDrinks] = useState('');

    const [adgendaTime, setAdgendaTime] = useState('');
    const [adgendaEvent, setAdgendaEvent] = useState('');
    const [adgenda, setAdgenda] = useState([]);

    const [leaderName, setLeaderName] = useState('');
    const [leaderTopic, setLeaderTopic] = useState('');
    const [leaders, setLeaders] = useState([]);

    const [ambassadorName, setAmbassadorName] = useState('');
    const [ambassadors, setAmbassadors] = useState([]);
```

```
    useEffect(() => {
      if (props.data) {
        const meetup = props.data.meetup;
        setName({ value: meetup.name, error: false });
        setVenue(meetup.venue);
        setDate(meetup.date);
        setMission(meetup.mission);
        setFood(meetup.food);
        setDrinks(meetup.drinks);
        setAdgenda(meetup.adgenda.map((e, i) => ({ ...e, id: Date.now() * i })));
        setLeaders(meetup.leaders.map((e, i) => ({ ...e, id: Date.now() * i })));
        setAmbassadors(meetup.ambassadors.map((e, i) => ({ name: e, id: Date.now() * i })));
      }
      if(props.updatingData) {
        props.refetch();
      }
    }, [props.data, props.updatingData])

    const adgendaIsValid = () => {
      return (adgendaTime.trim() !== '') && (adgendaEvent.trim() !== '');
    }

    const addAdgenda = (e) => {
      e.preventDefault();
      if (adgendaIsValid()) {
        setAdgenda([...adgenda, { id: Date.now(), time: adgendaTime, event: adgendaEvent
}]);
      }
    }

    const addLeader = (e) => {
```

```
        e.preventDefault();
        if (basicValidate(leaderName)) {
            setLeaders([...leaders, { id: Date.now(), name: leaderName, topic: leaderTopic }]);
        }
    }

    const addAmbassador = (e) => {
        e.preventDefault();
        if (basicValidate(ambassadorName)) {
            setAmbassadors([...ambassadors, { id: Date.now(), name: ambassadorName }])
        }
    }

    const submit = (e) => {
        e.preventDefault();
        alert('saved');
        props.createMeetup({
            variables: {
                name: name.value,
                date,
                venue,
                food,
                drinks,
                ambassadors: ambassadors.map(ambassador => ambassador.name),
                mission,
                adgenda: adgenda.map(adgenda => ({ time: adgenda.time, event: adgenda.event
}})),
                leaders: leaders.map(leader => ({ name: leader.name, topic: leader.topic }))
            }
        })
    }
```

```
const update = (e) => {
  e.preventDefault();
  props.updateMeetup({
    variables: {
      id: props.data.meetup.id,
      name: name.value,
      date,
      venue,
      food,
      drinks,
      ambassadors: ambassadors.map(ambassador => ambassador.name),
      mission,
      adgenda: adgenda.map(adgenda => ({ time: adgenda.time, event: adgenda.event
})),
      leaders: leaders.map(leader => ({ name: leader.name, topic: leader.topic }))
    }
  })
}

const basicValidate = (value) => {
  return value.trim() !== '';
}

const deleteAdgenda = (e, id) => {
  e.preventDefault();
  setAdgenda([...adgenda.filter(e => e.id !== id)]);
}

const deleteLeader = (e, id) => {
  e.preventDefault();
  setLeaders([...leaders.filter(e => e.id !== id)]);
}
```

```
const deleteAmbassador = (e, id) => {
  e.preventDefault();
  setAmbassadors([...ambassadors.filter(e => e.id !== id)]);
}

return (
  <form className={s.margin + ' ' + 'container'}>
    {/* <div className={s.alert + ' ' + 'alert alert-warning'} role="alert">
      error
    </div> */}
    <div className="row">
      <div className="col-sm form-group">
        <div className="form-group">
          <label htmlFor="name">Meetup Name</label>
          <input
            type="text"
            className={'form-control' + ` ${name.error ? s.errorBorder : ' '}`}
            value={name.value}
            onChange={(e) => setName({ value: e.target.value, error:
basicValidate(e.target.value) ? " : true })}
            />
          </div>
```

Table.js

```
import React, { useEffect, useState } from 'react';
import s from './Table.module.css';
import Seat from './Seat/Seat';
import { Link } from 'react-router-dom';
import Login from '../../common/Login/Login';
```

```
const Table = (props) => {

    const [isLogin, setLogin] = useState(false);
    const [tableSideSize, setTableSideSize] = useState(4);
    const [leftSideGuests, setLeftSideGuests] = useState([tableSideSize]);
    const [rightSideGuests, setRightSideGuests] = useState([tableSideSize]);

    useEffect(() => {
        if (props.data) {
            setLeftSideGuests(fillTableSide(0, tableSideSize));
            setRightSideGuests(fillTableSide(tableSideSize, tableSideSize * 2));
        }
    }, [props.data]);

    const fillTableSide = (start, end) => {
        let guests = [];
        for (let i = start; i < end; i++) {
            let guest = props.data.guestsByMeetupId[i]
                ? { ...props.data.guestsByMeetupId[i], isTaken: true }
                : { isTaken: false };
            guests.push(guest);
        }
        return guests;
    }

    const toggleLogin = () => {
        setLogin(!isLogin);
    }

    return (
        <div className={s.tableContainer}>
```

```
<div className={s.speakerBox}>
   <span className={s.speakerType}>Tech Speaker:</span>
   <div className={s.speakerImg}>
      <img src={props.speakerImg} />
   </div>
   <span>{ props.name }</span>
</div>
{
   isLogin &&  <Login toggleAuth={toggleLogin}/>
}
<div className={s.spaceBetween}>
   <ul className='seats'>
      {
         !props.loading &&
         leftSideGuests.map(guest => (
            <li key={ Math.random() }>
               <Seat
                  photo={guest.photo}
                  isLeft={true}
                  isTaken={guest.isTaken}
                  meetupId={props.meetupId}
                  toggleLogin={toggleLogin}
                  refetch={props.refetch}
                  meetupLoading={props.meetupLoading}
               />
            </li>
         ))
      }
   </ul>
   <div className={s.table}>
      <h4>Vlad Kalinichenko</h4>
   </div>
```

```
            <ul className='seats'>
              {
                !props.loading &&
                rightSideGuests.map(guest => (
                  <li key={Math.random()}>
                    <Seat
                      photo={guest.photo}
                      isLeft={false}
                      isTaken={guest.isTaken}
                      meetupId={props.meetupId}
                      toggleLogin={toggleLogin}
                    />
                  </li>
                ))
              }
            </ul>
          </div>
          <Link className={s.moreInfoBtn} to={`/meetup/more-info/${props.id}`} >
            MoreInfo
          </Link>
        </div>
      )
    }

    export default Table;
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” ___ 2020 р.

Веб-сервіс адміністрування IT-зустрічей

Графічний матеріал

КПІ.ІП-6313.045440.07.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.А. Халус

Нормоконтроль:

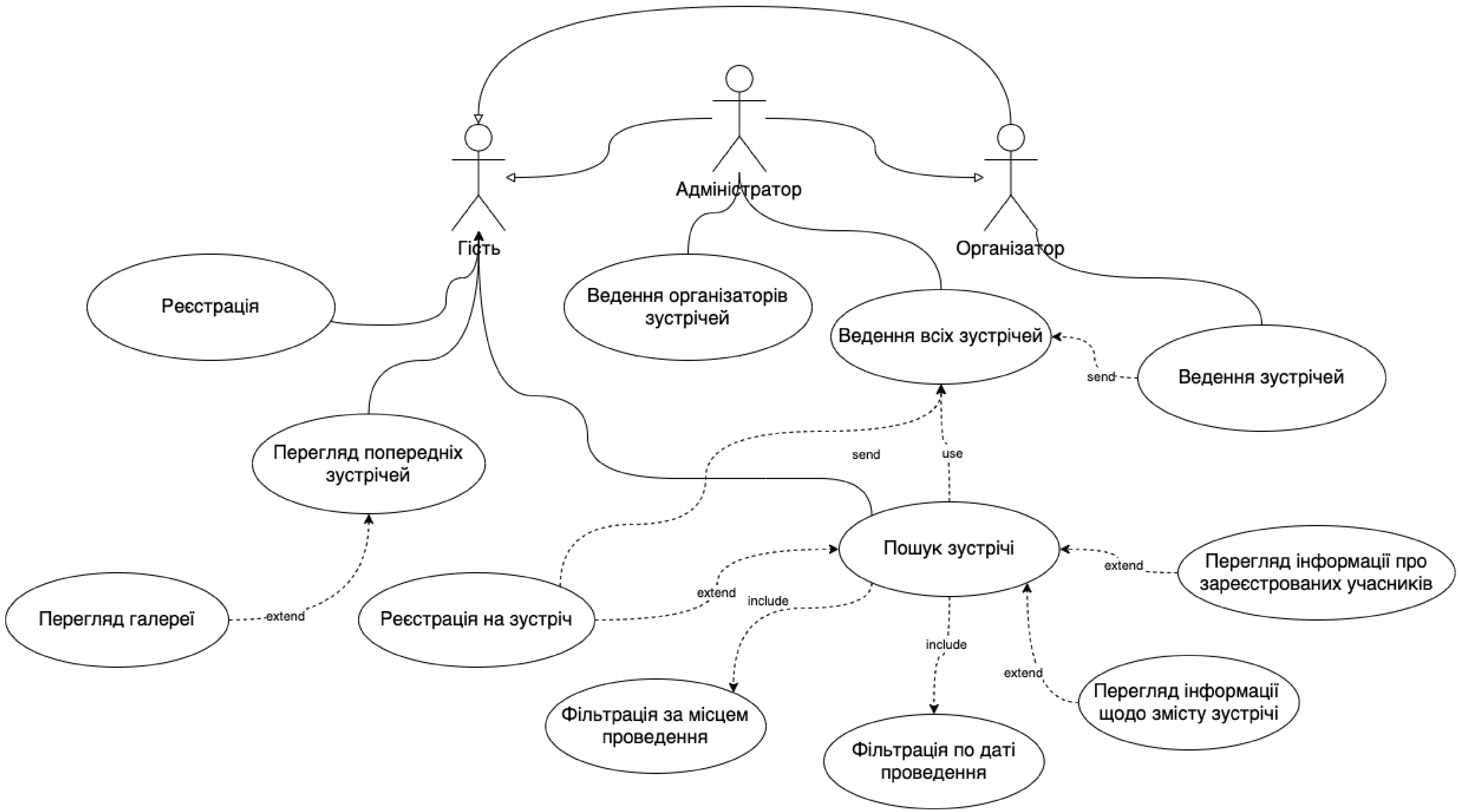
_____ К.І. Ліщук

Виконавець:

_____В.С. Калініченко

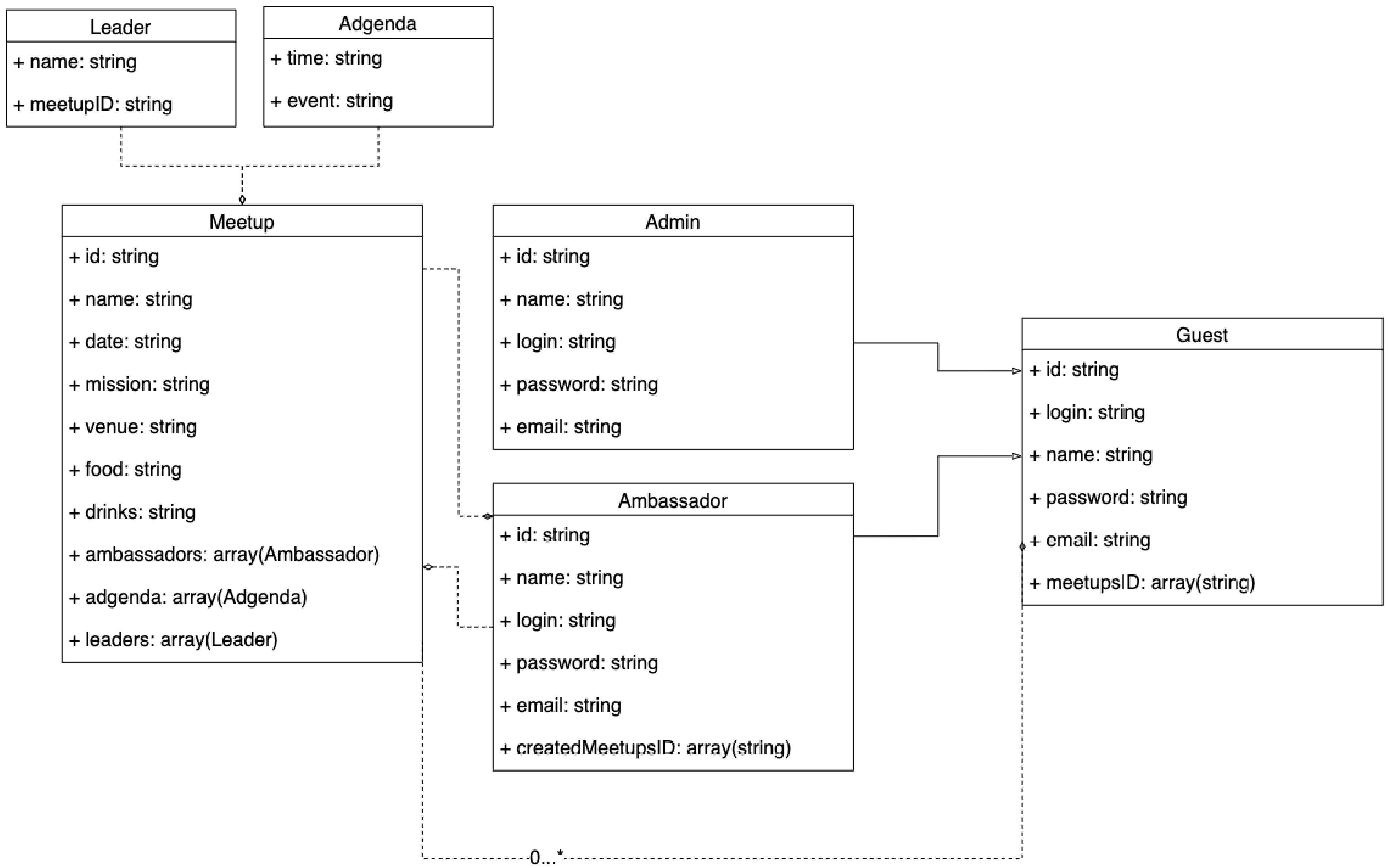
Київ – 2020 року

КПІ.ІП-6313.045440.07.99.СС



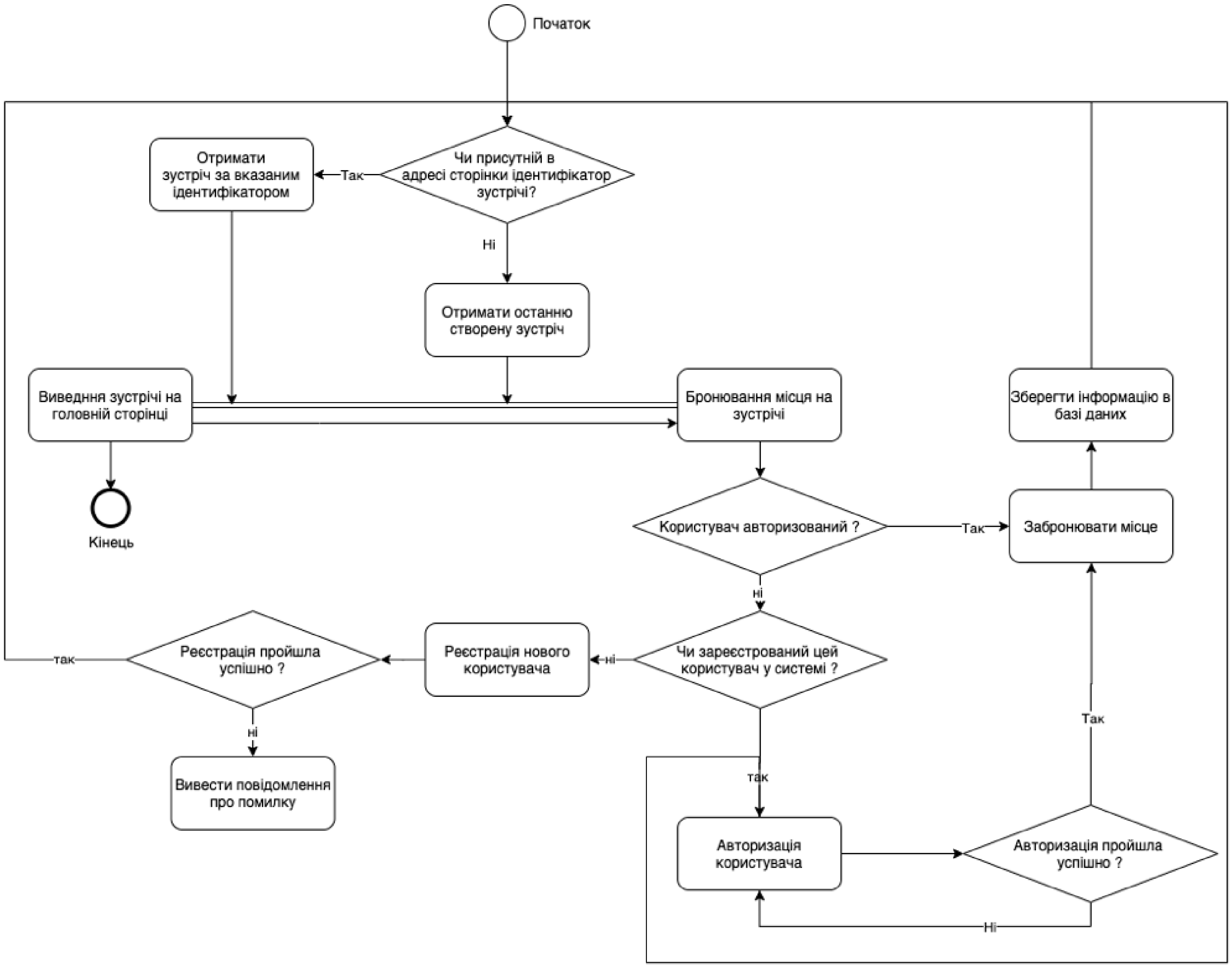
					КПІ.ІП-6313.045440.07.99.СС		
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Калініченко В.С.			Схема структурна варіантів використання	Літера	Маса	Масштаб
Перевірив		Халус О.А.					
Т. кон.						Аркуш	Аркушів
Н. кон.		Ліщук К.І.			Веб-сервіс адміністрування ІТ-зустрічей	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63	
Затвердив		Халус. О.А.					

КПІ.ІП-6313.045440.07.99.СБД



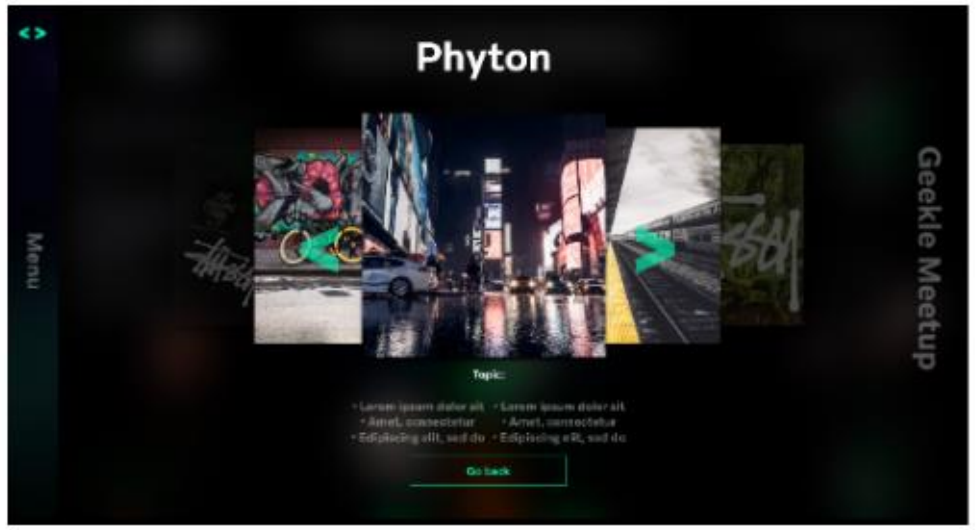
					КПІ.ІП-6313.045440.07.99.СБД			
Зм.	Арк.	№ документа	Підпис	Дата		Літера	Маса	Масштаб
Розробив		Калініченко В.С.			Схема бази даних			
Перевірив		Халус О.А.						
Т. кон.						Аркуш		Аркушів
Н. кон.		Ліщук К.І.			Веб-сервіс адміністрування ІТ-зустрічей	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Затвердив		Халус О.А.						

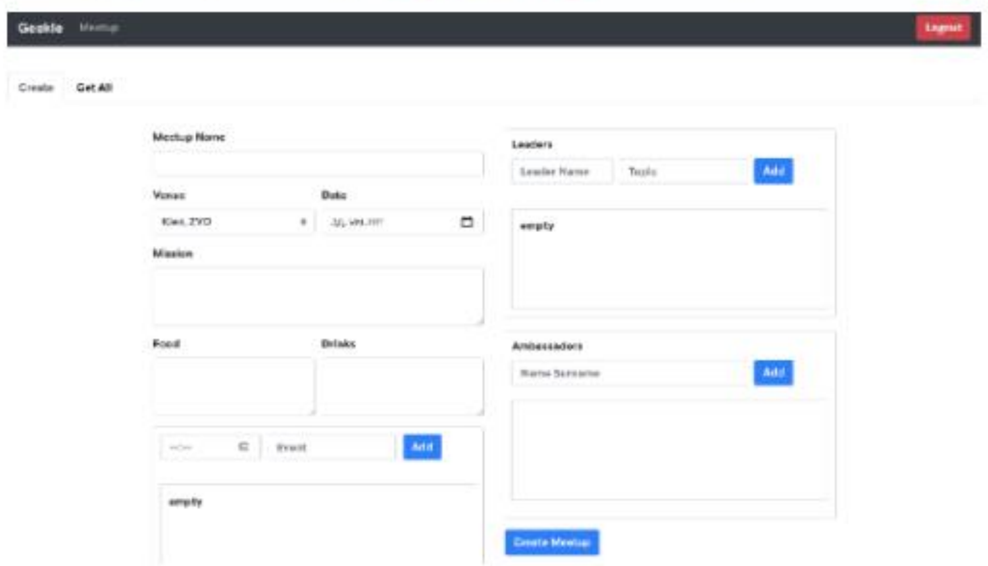
КПІ.ІП-6313.045440.07.99.СС



					КПІ.ІП-6313.045440.07.99.СС
Зм.	Арк.	№ документа	Підпис	Дата	
Розробив		Калініченко В.С.			Схема структурна діяльності
Перевірив		Халус О.А.			Літера / Маса / Масштаб
Т. кон.					Аркуш / Аркушів
Н. кон.		Ліщук К.І.			Веб-сервіс адміністрування ІТ-зустрічей
Затвердив		Халус. О.А.			КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63

КПІ.ІП-6313.045440.07.99.КЕ





					КПІ.ІП-6313.045440.07.99.КЕ			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Калініченко В.С.			Креслення вигляду екранних форм			
Перевірив		Халус О.А.						
Т. кон.						Аркуш	Аркушів	
					Веб-сервіс адміністрування ІТ-зустрічей	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Н. кон.		Ліщук К.І.						
Затвердив		Халус. О.А.						