МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Веб-орієнтований сервіс інформаційної підтримки соціально-культурних проектів для молоді»

Завідувач
випускаючої кафедри **Довбиш А. С.**

Керівник роботи **Ободяк В. К.**

Студент групи ІН-64-8 **Титов П.О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедри Довбиш А. С.

«_____» _____20___ р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-64-8 спеціальності «Інформатика» денної форми навчання Титова Павла Олеговича.

Тема: «Веб-орієнтований сервіс інформаційної підтримки соціально-культурних проектів для молоді»

Затверджена наказом по СумДУ

№_____ від _____20___ р.

Зміст пояснювальної записки: 1) аналітичний огляд веб-русурсів підтримки соціально-культурних проектів; 2) постановка задачі; 3) методи розробки веб-інтерфейсу та програмного забезпечення веб-сервісу; 4) практична реалізація; 5) тестування веб-сервісу.

Дата видачі завдання «_____»_____ 20___ р.

Керівник випускної роботи _____ Ободяк В. К.

Завдання прийняв до виконання _____ Титов П.О.

РЕФЕРАТ

Записка: 53 стор., 22 рис., 1 додаток, 10 джерел.

Об’єкт дослідження – процесс проектування веб-орієнтованного сервісу інформаційної підтримки соціально-культурних проектів для молоді.

Мета роботи – розробка веб-орієнтованого сервісу інформаційної підтримки соціально-культурних проектів для молоді

Методи дослідження – в процесі розробки веб-сервісу було використано мову розмітки HTML5, препроцессор SCSS, мову програмування JavaScript, фреймворк Angular, NodeJS, Bootstrap, базу даних MongoDB.

Результати – розроблений веб-орієнтований сервіс інформаційної підтримки соціально-культурних проектів для молоді, головним призначенням якого є пошук людей для спільного проводження дозвілля.

ВЕБ-СЕРВІС, ДОЗВІЛЛЯ, JAVASCRIPT, ANGULAR,
SCSS, BOOTSTRAP, MONGODB, CSS

Зміст

ВСТУП

В умовах сучасного інформаційного суспільства все актуальнішою стає потреба у новітніх технологіях, а саме використання сайтів, наприклад для покращення продажу різноманітних товарів, реклами фірм, навчання, перегляду фільмів, проводження вільного часу, перегляд новин чи прогнозу погоди.

У період загальної комп'ютеризації встановлення зв’язку між бізнес-партнерами чи іншими особами стало набагато простіше та зручніше.

Сучасна епоха характеризується як епоха глобального інформаційного суспільства, зміст якої становить експоненціальне зростання інформаційних технологій. Одним з головних проявів є виникнення глобальної мережі Інтернет, стрімке і неухильне розширення її використання у всіх сферах життя суспільства.

У міру розвитку Інтернету комунікативні можливості зростають, і все більш значна частина комунікації як ділового, так і особистого характеру здійснюється у віртуальному середовищі.

Технічні переваги віртуального спілкування полягають у дешевій передачі великих обсягів інформації на будь-які відстані, можливість корекції і зберігання інформації, що передається та ін.

На сьогодні Інтернет не просто місце збереження та обміну текстової, графічної, аудіо, відео інформації, а середовище існування на рівні з реальним світом. Різні інтернет-сервіси допомагають сучасній молоді знаходити нові корисні знайомства, розвиватися та організовувати соціально-культурні проекти.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз предметної області застосування

Якщо говорити про веб-ресурс, то потрібно турбуватися про розвиток самого Інтернету, все більше людей знайомі з веб-сайтом. Веб-сайт – це те, що користувачі відвідуєте щодня в Інтернеті, це як електронна книга, яка складається з багатьох книг, які можна відкрити та прочитати. Не обмежуючись певними книгами, є можливість знайти історії, різноманітний розважальний контент тощо.

Якщо припустити існування міста, в цьому місті багато будинків. Те саме з веб-сайтом. Веб-сайт – це як колекція будинків у місті. Хоча саме місто – це те, що ми називаємо Інтернетом. У будь-якому місті існує багато типів будинків, починаючи від класичних, звичайних моделей будинку до сучасних моделей. Так і вміст Інтернету. Потрібно лише з’ясувати, який з багатьох веб-сайтів обрати.

Від Google, Facebook, Twitter, CNN, Kompas тощо. Коли користувач відкриває ці сервіси на робочому столі або мобільному пристрої, відкривається веб-сайт. Хоча люди спілкуються із веб-сайтом щодня, можливо, багато хто не знає, що означає цей веб-сайт. У наш час необхідне розуміння веб-сайту, а також його історію, типи та переваги [14].

Веб-сайт веб-ресурсів - це сукупність сторінок у домені в Інтернеті, які створені з певною метою та взаємопов'язані між собою та можуть бути широко доступними через головну сторінку (домашню сторінку) за допомогою браузера за допомогою URL-адреси веб-сайту.

Веб-сайт був вперше створений Тімом Бернерсом-Лі наприкінці 1980-х, і був офіційно в Інтернеті лише в 1991 році [11].

Початкова мета команди Berners-Lee створити веб-сайт полягала в тому, щоб полегшити дослідників на своїх робочих місцях, коли вони обмінюватимуться чи вносять зміни до інформації.

Веб-сайт може належати фізичним особам, організаціям або компаніям. Загалом веб-сайт відображатиме інформацію або одну конкретну тему, хоча в даний час багато веб-сайтів відображають різну інформацію на різні теми.

Доступ до веб-сайту можна отримати на будь-якому пристрої чи гаджеті. За умови, що гаджет відповідає технологічним стандартам, необхідним для запуску сценарію веб-сайту, скрипт також є частиною веб-ресурсу. Тому що не всі веб-сайти використовують однакову технологію. Є веб-сайти, які містять лише просту інформацію, яка дуже легка для відкриття та доступу до них. І навпаки, є також багато веб-сайтів, які містять цікаві зображення та анімації, але дуже важкі для відкриття та завантаження сторінки за сторінкою [11, 14].

Є три елементи, які дуже важливі на веб-сайті, а також все ще є частиною самого веб-ресурсу. Без усіх цих елементів користувачі в Інтернеті ніколи не знайдуть і не мають доступу до них. Ці три елементи:

– домен. Якщо веб-сайт уподібнюється продукту, то домен - це бренд. Використання привабливого домену змусить зацікавити людей зайти на веб-сайт. Вибір унікального доменного імені також дозволяє людям легко запам'ятати його для подальшого повторного відвідування. Багато сторін готові витратити величезну суму коштів на придбання унікального домену. Унікальний домен забезпечує легкий доступ до користувачів. Цей веб-ресурс також дуже важливий для адміністратора.

Простота запам'ятовування забезпечує важливий момент для користувачів. Надаючи унікальне та просте доменне ім’я, людям, які мають доступ до нього, легко вимовляти та вводити його на клавіатурі.

Наявність незначної помилки в написанні лише однієї літери в домені може знайти веб-сайт не той, який шукав користувач.

– хостинг. Не менш важливий, ніж домени, хостинг має роль зберігати всі бази даних (сценарії, зображення, відео, текст тощо), необхідні для формування веб-сайту. Багато хостингових постачальників послуг з різними

можливостями, які користувач можете вибрати. Хостинг - це найважливіший інструмент веб-ресурсів.

Послуги хостингу, безумовно, сприяють потужності накопичувача, яка може зберігати багато даних для вашого веб-сайту.

Від вибору хостингу залежить не лише обсяг пам’яті, а й швидкість доступу до даних. Оскільки хостинг тут також підтримується обладнанням від хостинг-провайдера. Звичайно, це будуть дуже різні можливості між брендованими пристроями та мають високий авторитет із звичайними пристроями, ціни яких також не надто дорогі.

– зміст. Без вмісту на веб-сайті, можна сказати, що веб-сайт не має чіткої мети. Вміст є частиною веб-ресурсу. На веб-сайті можуть бути у вигляді тексту, зображень або відео. Якщо переглядати вміст, який подається, існує кілька видів веб-сайтів. Наприклад, соціальні медіа, новинні веб-сайти, купівля та продаж веб-сайтів або веб-сайтів, що містять вміст, заснований на інтересах, талантах та захопленнях.

За допомогою вмісту відвідувачі будуть шукати та перевіряти, чи відповідає веб-сайт тому, що вони шукають. Неопрацьований контент змушує відвідувачів плутати та не подобатися вмісту веб-сайту. Тому створення контенту є одним із ключових моментів, які слід враховувати при створенні веб-сайту, щоб він став хорошим веб-сайтом. Одне з речей, які роблять живим веб-ресурс – це вміст.

Персональний веб-сайт. На даний момент доступні різні сервіси, які можна використовувати для створення особистого веб-сайту.

Платні веб-сайти надають користувачам багато можливостей, які дозволяють вашому веб-сайту виглядати добре і легко доступний існуючим користувачам.

Якщо потрібно створити інтернет-магазин з функціями, які не сильно відрізняються від інтернет-магазинів, які існують сьогодні, то скориставшись

відповідним інструментом, можна зробити це за години або навіть хвилини, а веб-сайт інтернет-магазину можна отримати через Інтернет.

Існує багато видів інтернет-магазинів, які виготовляються відповідно до своїх побажань. Починаючи з інтернет-магазину із засобом відстеження, що користувачеві дуже легко дізнатися, в якій мірі замовлені товари.

Ще одним засобом є система автоматичного порівняння цін з іншими подібними предметами. Цей інтернет-магазин має більш складну логіку, але це може бути дуже корисно для його користувачів. Вони можуть дізнатися, які ціни на ринку, не потрібно відкривати іншу вкладку для порівняння.

Блоги зазвичай містять статті, спрямовані на обмін знаннями, ідеями чи досвідом автора. Блоги можуть містити нотатки з щоденних занять, особисті записки, навчальні посібники тощо. Метою цього блогу є обмін знаннями, пов’язаними з веб-хостингом, інтернет-маркетингом та іншими речами.

Для опублікованих статей кожен блог відрізняється, починаючи з одного дня однієї статті або навіть однієї статті за один тиждень. Це залежить від якості зробленої статті та кількості авторів [15].

Є блоги, які мають на меті поділитися, а деякі свідомо створюються з діловими цілями. Блоги, створені для ділових цілей, зазвичай є типом блогів, які розкривають інформацію для користувачів, і ці блоги зазвичай розміщують рекламу на своїх сторінках як засіб заробити гроші.

Професійні та непрофесійні блоги можна побачити із відображеної реклами. Є блоги, які показують рекламу з такою кількістю композицій, що людям важко зрозуміти, який саме вміст блогу, тому що майже 50% містять лише менш корисні оголошення. Хороший блог буде коригувати композицію рекламних матеріалів до складу самого веб-матеріалу.

Веб-сайт має багато переваг у нашому житті. Користувач можете отримати доступ до соціальних медіа за допомогою веб-сайту. Інтернет-покупки також відбуваються на веб-сайті. Коли потрібна важлива інформація, також є можливість отримати доступ до неї через веб-сайт.

Переваги веб-сайту не тільки в тому, що є ще багато переваг, які можна отримати від веб-сайту. Починаючи з переваг веб-сайтів для особистого, а саме веб-сайтів для особистого брендингу. За допомогою веб-сайту є можливість створити надійний та надійний особистий брендинг.

Можна поділитися своєю роботою та портфоліо на веб-сайті. Будь то написання, фотографії, картини, малюнки, графічні малюнки, музика. Є можливість створити професійний веб-сервіс, який буде допомагати людям.

Корисний веб-сервіс дійсно зробити непросто. Розробник повинен з самого початку представити себе через Інтернет. Складання хорошого та цікавого контенту який буде корисним багатьом користувачам стає тут важливим моментом. Якщо контент, який створюється, привертає інших людей відвідувати наш веб-сайт, то можна бути впевненими, що бренд є успішним, і лише думати про наступний спосіб зробити наш бренд більш відомим [15].

1.2 Актуальність виконання проєкту

Інтернет сьогодні – це те, без чого не може жити переважна частина населення. Дана технологія пов'язує людей з усього світу, з різних куточків землі. Дуже важко знайти людину, яка не користувався хоч раз Інтернетом.

Для когось Інтернет - це просто невичерпне джерело інформації, для когось - це робота, тут людина трудиться і заробляє собі на життя.

А для когось Інтернет - це середовище проживання, невід'ємна частина його життя, без якої він вже не уявляє свого існування.

Інтернет настільки зручний для обміну інформацією, що просто не міг не завоювали ту популярність, якою він володіє сьогодні. при розробці сучасного програмного забезпечення для Інтернету все більше уваги видаляється розробці такого типу додатків, як web-сервіси. У розвинених країнах, Інтернет вже давно домінує в області соціального обслуговування: за допомогою використання web-сервісів можна купувати квитки на потяги та літаки, оплачувати комунальні послуги, набувати побутової техніки. Це

позбавляє населення від марної трати часу стоячи в чергах, пошуки магазинів. У зв'язку зі сказаним, можна вважати, що розробка і застосування web-сервісів для надання соціальних послуг, в тому числі для дозвілля, є досить актуальною темою в сучасному суспільстві.

Існує досить багато визначень поняття «web-сервіс». Але найчастіше web-сервіс визначається як «web-ресурс з стандартизованими інтерфейсами, який може взаємодіяти з іншими програмними модулями за допомогою повідомлень, заснованих на визначених стандартах і протоколах » [15].

Головною відмінністю web-сервісу від web-сайту є, то, що web-сервіс може надавати певні послуги. Наприклад, такий web-сервіс як інтернет-магазин пропонує користувачеві певний товар, який він може придбати безпосередньо на даному web-сервісі.

Web-сайт – це сайт на якому розташовується будь-яка інформація, з якої користувач може ознайомитися. Аналіз робіт присвячених розробці web-сервісів дозволяє виділити два найбільш важливі напрямки: дизайн і навігація. Дизайн грає важливу роль в залученні уваги користувачів мережі Інтернет до web-сервісу. Навігація в web-сервісі повинна бути простою і зрозумілою – це позитивно позначається на конверсії і навіть на ранжируванні в пошукових системах.

Технології створення web-сервісів, що базуються на можливостях Інтернет, покликані кардинально поліпшити взаємодію людей і інформаційних систем один з одним і забезпечити взаємне проникнення різних систем і процесів. Вони утворені з цілого ряду стандартних протоколів взаємодії, засобів опису моделей даних і інтерфейсів, а також допоміжних мережевих служб, що забезпечують доступність бізнесфункцій організацій авторизованим користувачам через Інтернет з будь-якого підключеного до нього пристрою.

Існує безліч платформ і мов, на яких може бути розроблений web-сервіс. Виділяють кілька етапів розробки web-сервісу: планування, реалізація,

тестування, публікація, рекламування, супровід. У сучасному світі все актульнішою стає проблема пошуку друзів та знайомих Іноді непросто знайти собі компанію, наприклад, для катання на ковзанах, спілкування з новими людьми або завести нових друзів. Але і в тому і в іншому випадку бажано знайти відповідних людей швидше, і цілком можна зробити це всього за декілька хвилин за допомогою спеціального сервісу [15].

Веб-сервіс повинен бути створений в першу чергу для реального спілкування: користувач знаходите цікаві події та нові заклади, знайомитеся з людьми, призначаєте зустрічі і спілкуєтеся в реальності [1].

1.3 Постановка задачі

Метою дипломної роботи є створення веб-сервісу пошуку компанії користувачам для проведення спільного дозвілля.

Після проведення аналітичного огляду області розробки і визначення актуальності веб-сервісу, були поставлені такі задачі:

1) вибір середовища розробки;
2) розробка структури веб-сервісу;
3) створення інтерфейсу;
4) розробка веб-сервісу;
5) тестування роботи веб-сервісу.

Web-сервіс повинен містити наступні функціональні можливості:

1) реєстрація користувачів;
2) налаштування облікових записів користувача;
3) перегляд списку зустрічей;
4) бронювання зустрічі.

2 МЕТОДИ РОЗРОБЛЕННЯ ВЕБ-СЕРВІСУ

2.1 Середовище розробки

На переддипломній практиці було вирішено розробити веб-орієнтований сервіс інформаційної підтримки соціально-культурних проектів для молоді, який дозволить швидко знаходити людей, які хочуть провести разом час. Це можуть бути прогулянку в парку, виїзд на річку, похід в кафе або кінотеатрі, тощо.

Основою сайтів є HTML (Hypertext Markup Language) який був вперше розроблений Тімом Бернерс-Лі. Перша специфікація цього універсального і загальнодоступного мови-розмітки – HTML 0 була затверджена в 1991 році. Уже через два роки W3C оголосив про вихід HTML 3. Потім, майже одразу, пішла специфікація версії 3.2, з якою ми маємо справу зараз на переважній більшості Web-торінок. В HTML 4 сталася лише переробка концепцій, і не було додано нічого принципово нового [14].

На сьогоднішній день HTML вважається самим найкращим, скоріш за всього незамінним засобом розмітки гіпертексту та публікації в Інтернет. Мова розмітки гіпертексту за визначенням повинен інтерпретуватися браузером. Це, безумовно, накладає деякі обмеження на можливості мови і на сумісність нових конструкцій зі старими версіями браузерів.

Однак, саме ця особливість мов розмітки залишає чудову можливість генерувати HTML-код іншими програмами (CGI-скриптами). Але сучасні Web-сторінки вже не обходяться одним тільки HTML, так як його гармонійно доповнюють такі технології як: скрипт мови JavaScript і / або VBScript, каскадні таблиці стилів (CSS), іноді присутні Java-аплети.

Web-сторінка – це звичайний текстовий файл у відповідній кодуванні. У ньому описується вся сторінка за допомогою мови гіпертекстової розмітки – HTML (HyperText Markup Language). Коли користувач в своєму браузері (Internet Explorer / Nestcape Navigator) завантажуєте web-сторінку, то браузер виконує команди, записані на мові HTML, і, підкоряючись їм, виводить на

екран сторінку. Створювати HTML-файли потрібно в текстових редакторах, які вміють зберігати файли у форматі "Тільки текст" (розширення .txt). Якщо користувач створите такий файл в Word'е і збережіть у форматі "Документ", а потім поміняєте розширення .doc на .html, то у вас вийде повна абракадабра, тому що Word і сам не ликом шитий і додає в свої "фірмові файли" масу спеціальних команд форматування, дуже сильно відрізняються від тегів HTML. Тому на перших порах, щоб не заплутатися, ідеальним редактором є "Блокнот" (Notepad) зі стандартної поставки Windows.

Для того, щоб текстовий файл перетворився в HTML-файл, змінити його розширення з ".txt" на ".html" недостатньо. Треба також дотриматися "правило першого рядка". Цей рядок потрібен браузеру щоб визначити, як правильно інтерпретувати даний документ. В даному випадку ми говоримо браузеру, що HTML відповідає міжнародній специфікації версії 4.1, яка хоч і не відрізняється новизною, але, на відміну від більш пізніх версій, є повноцінним, широко поширеним стандартом без будь-яких невизначеностей. Після оголошення версії і типу документа необхідно позначити його початок і кінець. Це робиться за допомогою тега-контейнера . Необхідно відзначити, що будь-який HTML-документ відкривається тегом і їм же закривається. Ці теги повідомляють браузеру, що текст між ними слід інтерпретувати як HTML-текст. Оскільки документи HTML чисто текстові, тег говорить про те, що файл написаний на мові HTML (HyperTextMarkupLanguage - Мова гіпертекстової розмітки) [1, 11].

Javascript – один із трьох стовпів веб-розробки поряд із HTML та CSS. Хороший веб-розробник повинен ознайомитися з усіма трьома мовами, оскільки всі вони мають свої сильні сторони та особливості, унікальні для веб-контенту. HTML-код використовується для відображення тексту та медіа в браузері; він робить веб-сайт читабельним через браузери. CSS описує елементи HTML та спосіб їх відображення в браузері. Javascript, з іншого боку, використовується для того, щоб зробити веб-сторінки інтерактивними і є

мовою розмітки, яка широко використовується для створення веб-додатків. Можна безперешкодно поєднувати всі ці веб-технології, щоб зробити справді захоплюючі веб-сайти. Приємна річ сьогоднішньої веб-розробки полягає в тому, що більшість сучасних браузерів, як правило, підтримують веб-додатки та інтерактивні сторінки, у яких вбудований код Javascript. Таким чином, додаткові додатки навіть не потрібні. Усі користувачі повинні ввімкнути Javascript, який він, як правило, у своїх браузерах за замовчуванням [2].

Javascript – це об'єктно-орієнтована мова, яка в даний час оцінюється для використання на більш ніж 90% веб-сайтів. Однак не плутайте її з Java, яка є окремою мовою програмування, яку потрібно скомпілювати перед запуском і не є сценарієм. Кожен, хто хоче зайнятися серйозною веб-розробкою та передовим програмуванням, повинен вивчити Javascript.

Є багато причин для вивчення Javascript та для додавання його коду до стандартної розмітки HTML. Від інтерактивних кнопок до ігор, Javascript справді зробить веб-сайт унікальним і виділиться серед багатьох доступних стандартних веб-сайтів, заснованих на HTML5 та CMS (за допомогою Wordpress чи інших шаблонів) [12].

Сьогоднішнє покоління веб-користувачів також використовується для запуску програм, званих веб-додатками, безпосередньо через їх браузери. Ці програми настільки ж функціональні, як і стандартні програми, які мають бути встановлені в операційних системах користувачів і доступні миттєво. Розробка веб-додатків з інтерактивними функціями, як, наприклад, можливість користувачів безпосередньо вводити свої дані або грати в гру, є однією з сильних сторін Javascript.

Приклади реалізації Javascript включають форму, яку користувач можете додати користувачам, щоб заповнити, інтерактивну карту, яка пропонує користувачам бачити близькість до їх офісного місця, провідні елементи, такі як написання на сторінці, схоже на те, що вони були

намальовані вручну, цікаві ефекти анімації, коли користувачі натискають кнопку.

Javascript також дуже динамічний і простий в реалізації в стандартний HTML-код. Це мова розмітки, яка пишеться в простому тексті, як і HTML, і мова на стороні клієнта, яка не вимагає збирання даних перед виконанням. Оскільки це фронтальна та клієнтська мова, код можна читати безпосередньо через веб-переглядач швидко і без того, щоб він не надсилався та читався далеко віддаленим сервером на додаток до клієнта. Це означає, що хороші реалізації Javascript можуть додати швидкість та ефективність веб-розробки.

На додаток до цього, Javascript – це об'єктно-орієнтована мова. Це чудово підходить для нових розробників, оскільки її легко зрозуміти. Javascript підтримує успадкування завдяки прототипуванню поряд із властивостями та методами. Події навіть можуть бути реалізовані за бажанням у межах коду, який, можливо, також буде інкапсульований для визначених об'єктів [3].

Але для полегшення та прискорення роботи треба використовувати фреймворки. Найкращими можна вважати:

React. Для того, щоб отримати можливість використовувати React в розробці веб-проектів, потрібно вивчити багато додаткових інструментів. Користувач може, наприклад, використовувати список різних інструментів, представлених бібліотеками, які працюють спільно з React. Це - Redux, MobX, Fluxy, Fluxible, RefluxJS. В React-розробці, окрім цього, можна використовувати jQuery AJAX, Superagent, Axios та Fetch API[13].

Технологія React.Suspense дозволяє вдосконалити обробку асинхронної завантажувальних даних у програмі React. Якщо можна описати цю технологію в двох словах, то можна сказати, що вона може організувати оглядові компоненти, виконати невдале використання і при цьому не надіслати роботу всього додатка.

Ще одна новинка з’являється в React 16.8 і це Huki React, що дозволяє просто розроблювати користувачі, які мають можливості реагувати та

створити цей компонент без застосування коментарів компонентів, заснованих на класах. Серед таких можливостей – управління станційним компонентом і роботою з методами його живого цикла. Реагуйте, що міститься кілька вбудованих хуків, але завдяки цій програмі можливо створити власні хуки.

React-додатки вміщують із компонентів, які містять логіку роботи додатків та HTML-розмітки для формування інтерфейсу. Для того, щоб покращити взаємодію між компонентами, розробник може використати Flux або використати JavaScript-бібліотеку.

В React-програмуванні використовуються такі поняття як стан (стан) і властивості (реквізит) компонента. Вони представлені відповідними об'єктами. Користувач може використовував можливість організованого зберігання даних у компонентах та обмінних даних між компонентами. Наприклад - передані дані з програмної логіки, реалізованої компоненти, в інтерфейсі, або передачі даних від батьківських компонентів дочернього компонента.

Біля бібліотек React склалася ціла екосистема, представлена різними допоміжними інструментами та бібліотеками. Деякі составні частини цієї екосистеми:

Сама бібліотека React and React Router – засіб для управління маршрутами в пропозиціях.

Пакет react-dom, призначений для роботи з DOM.

Інструменти розробки Реакт для браузерів Firefox та Chrome.

JSX – мова розміщення, яка дозволяє описувати HTML-елементи в JavaScript-коді.

Середні командні строки create-react-app, які продаються для створення шаблонів React-проектів.

Різні допоміжні бібліотеки. Слід використовувати їх, наприклад, можна відредагувати бібліотеку Redux, використовуючи для керування станцією

запропоновані умови та бібліотеку Axios, використовуючи для обміну даними з серверними API.

Переваги:

− безліч документації та онлайн-ресурсів. Завдяки підтримці Facebook існує безліч можливостей використовувати тонну документації та онлайн-ресурсів для навчання та використання фреймворка Javascript для React.

− швидка, гнучка, ефективна і легка технологія: система JS широко рекомендується завдяки своїй ефективності, невеликому розміру блоків, гнучкості та швидшому підходу до роботи завдяки простій компонентної моделі і функціональності рендеринга на стороні сервера;

− перехід між версіями. Міграція між версіями, як правило, дуже проста, Facebook надає «codemods» для автоматизації величезної частини процесів.

− відмінно себе почуває при роботі з ES6 / 7 ReactJS, може брати на себе будь-яке навантаження;

− структура має компонентну архітектуру, яка революціонізувала веб-розробку додатків і вплинула на інші технології;

− DOM дозволяє об'єднувати HTML, XHTML або XML документи за певними критеріями, найчастіше в дерево, тому React відмінно підходить для веб-браузерів при аналізі різноманітних елементів веб-додатків.

Недоліки:

− необхідні кошти збірки: дана інфраструктура JavaScript може працювати некоректно без адекватних інструментів збірки або може відображати несумісність з іншими бібліотеками та кодами через високу DOM;

− велика крива навчання: на відміну від Vue, React вимагає більше часу для вивчення концепцій і реалізації. React JS вимагає величезну кількість знань в тому як інтегрувати користувальницький інтерфейс в структуру MVC;

– відсутність впорядкованої документації - надшвидкий обмін рішеннями в ReactJS не залишає місця для впорядкування документації, документи розміщені трохи хаотично, оскільки багато розробників індивідуально вносять їх в базу даних без будь-якого систематичного підходу.

Отже, React – бібліотека, що в повному обсязі може бути найкращою для тих, хто планує створити продвинутий веб-проект [3].

Сучасний Angular - це просунутий модульний фреймворк для фронтенд-розробки. Раніше для підключення Angular на сторінках досить просто було додати у свій HTML-код відповідний тег, тепер він може розробити свій власний проект, необхідний для нього.

Angular відомий свою гнучкість. Angular 1.x. Проте багато розробників в наші дні використовують кутовий 2+ з-за MVC-архітектури фреймворка, який значно змінює в стороні архітектури, заснованої на компонентах.

Тому, хто хоче користуватися Angular, доводиться, при освоєнні цього фреймворка, зіткнутися з деякими труднощами. Так, для створення Angular програм, запропонованих практично, обов'язково використовувати TypeScript.

Хоч це і ускладнює роботу з Angular, але є свої плюси. У приватності, це підвищує надмірність, запропоновану для зйомки просунутого контролю типових, це дає програму додаткові засоби розробки. Angular - це повноцінний JS-фреймворк, який надає сучасну веб-програму, яка потрібна для продуктової роботи. На Angular стоїть повернути увагу тем, кому хочеться отримати в своєму розпорядженні обширний набір стандартних середніх і свіжих мінімуму з використанням сторонніх бібліотек.

Переваги Angular:

– двостороння прив'язка даних;

– мобільний підхід до веб-розробці;

– стабільна і довгострокова підтримка google;

– універсальний mvvm модуль (дає можливість окремо оперувати в одному розділі програми, використовуючи той же набір даних);

− взаємозалежність функцій, оскільки вони мають пов'язаності з компонентами і модулями;

− RXJS, блискавична компіляція (менше 2,9 секунд), змінений пуск HttpClient.

Недоліки Angular:

− проблема з дренажем батареї: додатки, створені за допомогою рамкової системи Javascript, відомі тим, що надмірно розряджають акумулятор пристрою;

− велика крива навчання: Angular показує високу шкалу навчання, вам потрібно більше часу, щоб освоїти цю структуру;

− інтеграційні помилки, які можуть виникати при переході від старої версії до нової;

− складну мову програмування, незважаючи на те, що Angular використовує TypeScript [3].

Vue. Ідеї, що лежать в основі Vue, запозичені з Angular і React. Цього року Vue завантажили більше 40 мільйонів разів.

При роботі з Vue логіка компонента, його макет і стилі зберігаються в одному файлі. Так само, за винятком стилів, матеріали проектів зберігаються і в React. Взаємодія компонентів в Vue забезпечується за допомогою об'єктів, які зберігають властивості і стан компонентів. Цей підхід теж, ще до того, як він з'явився в Vue, був використаний в React.

Vue, що ріднить його з Angular, дозволяє змішувати HTML-розмітку і JavaScript-код. Для того щоб інтегрувати дані компонента в шаблон, потрібно використовувати директиви Vue. Такі, як v-bind або v-if.

Одна з причин, по якій Vue варто розглядати як гідну альтернативу React, полягає в тому, як тут організовано управління станом додатки. У React-проектах, при використанні зв'язки React + Redux, у міру зростання розмірів додатки ускладнюються і процедури, необхідні для управління його станом. Це може звестися до того, що програмісту, замість роботи над самим

додатком, доводиться витрачати чимало часу на налаштування механізмів Redux. В Vue для управління станом використовується бібліотека Vuex. Вона схожа на Flux і створена спеціально для Vue. Працювати з нею набагато зручніше, ніж з Redux.

Vue ж набагато простіше ніж Angular і не так сильно обмежує програмістів. Отже:

Переваги Vue:

− швидкий набір популярності: всього за кілька років з моменту свого створення велика кількість підприємств додали Vue.js в свій технічний стек.

− швидка настройка: Vue має вбудовану прив'язку даних і модель MVC (модель, вид, контролер), що значно спрощує настройку в порівнянні з Angular.js і React.js.

− більш проста інтеграція: платформа підтримує легшу інтеграцію з елементами HTML.

− маленька крива навчання: в порівнянні з Angular JS Framework, Vue набагато легше вивчати, розуміти і використовувати.

Недоліки Vue:

− мало ресурсів: структура і раніше занадто молода для пошуку корисних рішень в інтернеті і самонавчання.

− маленьке залучення спільноти. як уже згадувалося в останньому пункті, vue.js є новачком на ринку і, отже, має меншу підтримку спільноти в порівнянні з angular і react технологіями.

Для бекенд частини була використаний фремворк Node.js - теж є одним з кращих JavaScript фреймворків завдяки управлінню подіями (принципом введення-виведення) для розробки ефективних і легких додатків працюють в безлічі розподілених пристроїв. Фреймворк з кожним роком стає все більш і більш популярним. Це відповідний вибір для тих, кому необхідна повнофункціональна JS середа з доступом до всіх інструментів [3].

Переваги Node.js:

− можливість відправки кількох запитів: неблокіруючіх середу введення-виведення Node.js дозволяє розробникам додатків обробляти декілька запитів одночасно.

− create from Scratch: технологія дозволяє розробникам створювати кожен окремий елемент з нуля; немає ніяких обмежень в цьому плані.

− дуже активна спільнота: спільнота розробників Node.js надзвичайно активно, що має на увазі швидкий і надійний доступ до всіх кодами і рішенням.

− клієнтський підхід: ця структура JS повністю враховується при розробці клієнтської сторони.

Недоліки Node.js:

− відсутність многопоточного програмування: Node.js не підтримує багатопотокове програмування. Це має на увазі, що інфраструктура JS як і раніше не є ідеальним варіантом для виконання тривалих обчислень. Якщо користувач спробує створити на даній технології додаток, продуктивність програми може бути дуже низькою.

− нестабільність інтерфейсу вузла. Були зареєстровані різні випадки нестійкості Node API [4, 5].

Для зберігання даних було порівняно 2 найпопулярніші бази даних , це SQL(Реляційні бази даних) та MongoDB (нереляційні бази даних).

Для визначення і обробки даних реляційні бази даних використовують структурований мову запитів (Structured Query Language, SQL). З одного боку, це відкриває великі можливості для розробки: прийнято вважати, що один з найбільш гнучких і поширених мов запитів є SQL, так що його вибір дозволяє мінімізувати ряд ризиків, і буде особливо до речі, якщо має бути робота з комплексними запитами. З іншого боку, в SQL є ряд обмежень. Побудова запитів на цій мові зобов'язує визначати структуру даних і, як у випадку з Містом А, подальша зміна структури даних може бути згубним для всієї системи [6].

Динамічну структуру даних, пропонують нереляційні бази даних, що мають декілька способів зберігання: орієнтовано по колонках, в вигляді графів, документо-орієнтовано або на основі пар «ключ-значення». Така гнучкість означає наступне:

- користувач може створювати документи, не ставлячи їх структуру заздалегідь;
- кожен документ може мати власну структурою;
- у кожної бази даних може бути власний синтаксис;
- користувач має можливість додавати поля прямо під час роботи з даними.

Зазвичай SQL бази даних є вертикально масштабовані, тобто користувач може збільшувати навантаження на окремо взятий сервер, нарощуючи потужність центральних процесорів, обсяги ОЗУ або системи зберігання даних. Якщо говорити про NoSQL бази даних, то вони являються горизонтально масштабованими. Це означає, що користувач може збільшувати трафік, розподіляючи його або додаючи більше серверів до вашої СУБД. У другому випадку, означає, що система має можливість ставати більшою і потужнішою, роблячи вибір NoSQL бази даних віддається перевага для великих або постійно мінливих структур даних.

Дані представлені в вигляді таблиць є у реляційних СУБД, а ось в нереляційних - у вигляді графів, документів, пар «ключ-значення» або wide-column сховищ. SQL бази даних мають переваги за вибором для додатків, які повинні мати можливість транзакції з декількома записами - як, наприклад, система облікових записів - або для застарілих систем, які були побудовані для реляційних структур [7].

У число СУБД для SQL баз даних входять MySQL, Oracle, PostgreSQL і Microsoft SQL Server. Для роботи з NoSQL підійдуть MongoDB, BigTable, Redis, RavenDB Cassandra, HBase, Neo4j і CouchDB.

SQL vs. NoSQL: MySQL або MongoDB.

Розібравшись з ключовими структурними відмінностями SQL і NoSQL баз даних, варто уважно розглянути їх функціональні особливості на прикладі MySQL і MongoDB.

MySQL: реляційна СУБД, її переваги:

− перевірено часом: MySQL - вкрай розвинена СУБД, що означає наявність великої спільноти навколо неї, безліч прикладів і високу надійність;

− сумісність: MySQL доступна на всіх основних платформах, включаючи Linux, Windows, Mac, BSD і Solaris. Також у неї є бібліотеки для мов на зразок Node.js, Ruby, C #, C ++, Java, Perl, Python і PHP;

− окупність: Це СУБД з відкритим вихідним кодом, що знаходиться у вільному доступі;

− реплицируємість: Базу даних MySQL можна розподіляти між кількома вузлами, таким чином зменшуючи навантаження і покращуючи масштабованість і доступність додатка;

− шардінг: В той час як шардінг неможливий на більшості SQL баз даних, MySQL є винятком [6].

MongoDB: нереляційних СУБД, переваги:

− динамічна схема: Як згадувалося вище, ця СУБД дозволяє гнучко працювати зі схемою даних без необхідності змінювати самі дані;

− масштабованість: MongoDB горизонтально масштабована, що дозволяє легко зменшити навантаження на сервера при великих обсягах даних;

− зручність в управлінні: СУБД не потребує окремому адміністратора бази даних. Завдяки достатній зручності у використанні, їй легко можуть користуватися як розробники, так і системні адміністратори;

− швидкість: Висока продуктивність при виконанні простих запитів;

− гнучкість: В MongoDB можна без шкоди для існуючих даних, їх структури і продуктивності СУБД додавати поля або колонки. Для дипломного проекту була вибрана MongoDB [6–8].

Для стилізації сайту використовувався CSS.

Каскадні таблиці стилів (Cascading Style Sheets = CSS) – це мова, яка відповідає за візуальне уявлення документів користувачеві. Під документом буде вважатися набір інформації про структуру сторінки, описуваний мовою розмітки.

А уявлення документа користувачеві, в свою чергу, означає його перетворення в зручну для сприйняття форму. Браузери, такі як Firefox, Chrome або Internet Explorer, були створені для візуального відображення документів, наприклад, на екрані комп'ютера, проекторі або виведення через принтер. Для зручності використовувався препроцесор Sass з синтаксисом SCSS.[9]

CSS препроцесор (від англ. CSS preprocessor) – являє собою надбудову над CSS, що дозволяє використовувати раніше недоступні можливості для CSS, за допомогою нових синтаксичних конструкцій.

Основне завдання препроцесора - це надання зручних синтаксичних конструкцій для розробника, щоб спростити, і тим самим, прискорити розробку та підтримку стилів в проектах.

За допомогою препроцесорів користувач може писати код, який націлений на:

- читабельність для людини;
- структурованість і логічність;
- продуктивність;

Спочатку, мабуть, немає причин, що SASS необхідний. Зрештою, CSS давав можливість розробляти привабливо оформлені та технічні сторінки протягом десятиліть. Навіть якщо користувач хочете працювати з SASS з цього моменту, користувач не можете уникнути CSS. Старомодна мова таблиці стилів і надалі залишатиметься основою презентації. SASS та інші мови просто будуються на скелеті CSS.

Звичайно, у SASS у вас є кілька варіантів, які не пропонуються лише CSS:

Змінні: За допомогою SASS користувач можете зберігати інформацію у змінних, які будуть використані знову. Так, наприклад, можна центрально зберігати значення кольору під змінною.

Математичні функції: У SASS користувач також може використовувати математичні операції, такі як +, -, *, / або%. Це дозволяє, наприклад, впливати на технічні характеристики розміру.

Функції: Інші функції також полегшують роботу над дизайном. Вони дозволяють змінювати значення кольорів або аналізувати списки, серед іншого.

Петлі: Ще одна перевага SASS - це можливість налаштування циклів. Вони повторюються, поки вони не досягнуть визначеного вами статусу.

Відмінність випадків: вони мають подібну функцію до умовних інструкцій "якщо" та "інше". Дані команди виконуються лише за умови дотримання конкретних умов.

Мікінс, просто кажучи, - це шаблони. Користувач може або створити їх самостійно, або просто інтегрувати їх у власний код, використовуючи рамки.

Відступи: Початковий SASS (на відміну від SCSS) працює з відступами та розривами рядків, щоб структурувати код. Для позначення кінців рядків вам не потрібні дужки, щоб відображати гніздування або крапки з комою.

Вкладення: CSS не дозволяє працювати з вкладеним кодом. SASS, однак, надає користувачам можливість візуально представляти залежності, щоб зменшити надмірності та спростити процес написання [9].

Наслідування: Можна успадкувати властивості від одного селектора до іншого. Це економить певні зусилля і зберігає код стрункішим.

Часткові файли: щоб інтегрувати елементи коду у файл SASS, користувач також може використовувати частки. Це файли, які мають містити лише кілька рядків CSS і імпортуються у файл SASS командою.

Окрім SASS, LESS також зарекомендував себе у колах розробників. Ця мова таблиці стилів сильно орієнтована навколо CSS і нагадує SCSS у своєму

синтаксисі. Обидва препроцесори мають деякі однакові властивості: І SASS, і LESS дозволяють використовувати суміші та змінні. Однак, одна з різниць полягає в тому, що SASS базується на Ruby, тоді як LESS використовує JavaScript. Але навіть це не дає жодному з препроцесорів переваги перед іншими.

Натомість реальні відмінності виявляються в логічних функціях: LESS дає користувачам можливість активувати міксин тільки тоді, коли виникають конкретні ситуації. Це корисна функція, але вона відображає ступінь логічних посилань у LESS. SASS, з іншого боку, пропонує розмежування циклів і регістрів, як відомо з мов програмування.

За допомогою SASS користувачі можуть вибирати між "відступом синтаксис" або SCSS. Кожен розробник може вирішити для себе, чи хотів би він відійти від правил CSS чи залишитися ближче до оригіналу. LESS не пропонує цей вибір. Тут користувачі повинні дотримуватися старих правил. Код у LESS автоматично є набором CSS: Весь вихідний текст, сформульований у CSS, також функціонує в LESS - як і у SCSS [9].

SASS набагато популярніший серед веб-дизайнерів. Але це може бути тому, що SASS трохи старше. LESS спочатку підтримувався добре розглянутою рамкою Frontend Bootstrap, яка спиралася на молодшого препроцесора. Але з версією 4 проект офіційно перейшов на SASS, що ще більше підвищило популярність синтаксично дивовижних таблиць стилів.

Також для стилізації був використанний фреймворк, вібір робився між цими:

CSS фреймворки в 2019 Pure CssPure - це каскадна таблиця стилів (CSS), розроблена YAHOO. Вона допомагає створювати швидкі, красиві і чуйні веб-сайти. Даний фреймворк дуже сильно оптимізований, це покращує комфорт роботи з ним. Pure CSS один з найменших фреймворків, всього 3,8 КБ в форматі gzipped.

Має величезну кількість функцій. Pure CSS вважається найпростішою моделлю вибору вже протягом багатьох років, в тому числі при роботі в команді для розробки програмного забезпечення в аутсорсингової індустрії. Pure CSS має побудова на Normalize.css, забезпечує компоновку і стиль для власних HTML-елементів, а також для найбільш поширених компонентів для користувача інтерфейсу.

Це саме те, що вам потрібно для вільної роботи. Pure CSS чуйний фреймворк, його елементи відмінно виглядають під час роботи на всіх розмірах екрану. Pure CSS має мінімальну кількість стилів, дає можливість створювати свої стилі додатків поверх Pure CSS. Він був спеціально зроблений так, щоб мінімально не заважати розробнику при створенні свого власного стилю. Для роботи з Pure CSS необхідні базові знання в HTML, CSS, JavaScript, Document Object Model (DOM) і в будь-якому текстовому редакторі. Великим плюсом буде розуміння того, як працюють веб-додатки.

CSS фреймворки на сьогодні – Bootstrap Даний фреймворк був розроблений Марком Оттоном і Якобом Торнтоном з Twitter, для забезпечення узгодженості між внутрішніми інструментами Twitter. На рівні з Pure CSS Bootstrap є одним з найбільш поширених і широко використовуваних інтерфейсних систем. Використання Bootstrap зменшує кількість часу потрібного для написання великих кодів (з нуля).

Робота з таким фреймворком значно прискорює процес створення сторінок. Стандартні стилі легко міняти, що забезпечує гнучкий і простий процес створення макетів сайтів. HTML Resets знаходяться в одному модулі під назвою "Reboot". HTML Reset - це набір стилів, який замінює собою звичні CSS-стилі, встановлені в браузер за замовчуванням. Такі можна побачити, якщо сторінка працює без підключеного CSS. Прийнято відключати їх, щоб вони не псували процес верстки. Вбудована підтримка flexbox дає безліч переваг для користувачів [11].

Flexbox – це найпотужніший компонент html5, завдяки якому верстка поводиться в точності як таблиця або як набір блоків - в залежності від того як захоче розробник. Зазвичай розробники Перекомпілюйте CSS, щоб блокова верстка була запущена не з допомогою float: left, а на основі flexbox. Класними компонентами фреймворка вважається Sass-змінні і більш структурована система фреймворка. Не може не радувати той факт, що фреймворк існує і вдосконалюється, старі помилки виправляються і дуже часто виникають нові.

CSS фреймворки в 2019 – MilligramMilligram забезпечує мінімальне налаштування стилів для швидкого і чистого початку стилізації сторінки. Завдяки малій вазі файлу - тільки 2kb в форматі gzipped, ця крихітна, але потужна структура входить до трійки кращих полегшених фреймів. В арсеналі Milligram є все що потрібно: гріди, таблиці, форми, підказки, кнопки і інші інструменти. CSS Milligram вводить кілька нових одиниць, включаючи блок rem, який позначає «root em».

Модуль rem відноситься до розміру шрифту кореневого елементу html. Це означає, що ми можемо визначити один розмір шрифту на кореневому елементі і визначити всі одиниці rem в процентах від загальної кількості. Розмір шрифту друкарні 16 пікселів, висота рядка 24 пікселя.

Milligram має три типи списків: невпорядкований список використовує елемент ul, буде відзначений контурними колами, упорядкований список використовує елемент ol, а ваші елементи будуть відзначені цифрами, в списку опису використовується елемент dl, ваші елементи не будуть марковані.

Виходячи у усіх вище сказанних факторів, був обраний феріймворк – bootstrap [9].

У випускному проєкті використано архітектуру REST для обміну інформації між Front-End і BackEnd.

REST (REpresentational State Transfer) – це архітектура, тобто принципи побудови розподілених гіпермедіа систем, того що іншими словами

називається World Wide Web, включаючи універсальні способи обробки і передачі станів ресурсів по HTTP.

REST на сьогоднішній день практично витіснив всі інші підходи, в тому числі дизайн заснований на SOAP і WSDL.

REST дає масштабованості взаємодії компонентів системи (додатки), спільність інтерфейсів, незалежне впровадження компонентів та проміжні компоненти, що знижують затримку, які посилюють безпеку [10].

REST використовують коли:

- коли є обмеження пропускної здатності з'єднання;
- якщо необхідно кешувати запити;
- якщо система передбачає значне масштабування;
- у сервісах, що використовують AJAX.

Переваги REST:

- відсутність додаткових внутрішніх прошарків, що означає передачу даних в тому ж вигляді, що і самі дані. Тобто дані не обертаються в XML, як це робить SOAP і XML-RPC, не використовується AMF, як це робить Flash і т.д. Просто віддаються самі дані;
- кожна одиниця інформації (ресурс) однозначно визначається URL - це значить, що URL по суті є первинним ключем для одиниці даних. Причому абсолютно не має значення, в якому форматі знаходяться дані за адресою - це може бути і HTML, і jpeg, і документ Microsoft Word;

Як відбувається управління інформацією ресурсу – це цілком і повністю ґрунтується на протоколі передачі даних. Найбільш поширений протокол звичайно ж HTTP і він використовується в нашому сервісі . Для HTTP дію над даними задається за допомогою методів: GET (отримати), POST (додати, редагувати, видалити), PUT (додати, редагувати), DELETE (видалити). Таким чином, дії CRUD (Create-Read-Update-Delete) можуть виконуватися як з усіма 4-ма методами, так і тільки за допомогою GET і POST [10].

2.2 Проєктування структури веб-сервісу

Структура сайту – це його основа. Ще на етапі створення сайту необхідно подбати про зручне структуруванні всієї інформації. Фахівці з технічної підтримки сайту можуть запропонувати кілька типів структур, кожна, з яких має свої переваги і недоліки [15].

Лінійна структура – розташування сторінок в певному порядку. Строго визначений перехід з однієї сторінки на іншу. Така структура обумовлена, наприклад, при навчанні. Маючи в своєму розпорядженні сторінки в певному порядку, ви можете бути впевнені, що користувач не пропустить потрібний матеріал.

Ієрархічна – розбиття сторінок за категоріями та підкатегоріями. Вважається найбільш зручною структурою.

Довільна – розташування сторінок відбувається у вільному порядку.

Взаємодія між додатками – веб-сервіси дозволяють різним програмам спілкуватися між собою та обмінюватися даними та послугами між собою.

Веб-сервіси використовують SOAP через протокол HTTP, тому можна використовувати наявний недорогий Інтернет для впровадження веб-служб.

Кожен фреймворк потребує певної архітектури, щоб переконатися, що весь фреймворк працює за бажанням. Так само у веб-сервісах існує архітектура, яка складається з різних ролей, як зазначено нижче:

Постачальник - Провайдер створює веб-сервіс та робить його доступним для клієнтської програми, яка хоче ним користуватися.

Запитник – запитувач - це не що інше, як клієнтська програма, якій потрібно звернутися до веб-служби. Клієнтським додатком може бути .Net, Java або будь-яка інша мовна програма, яка шукає певну функціональність за допомогою веб-сервісу.

Клієнтська частина веб-сервісу повинна мати кілька сторінок: головна, всі зустрічі, деталі зустрічі, запропонувати зустріч, особистий кабінет

У правому верхньому кутку головної сторінки буде кнопка «Увійти», яка перенаправляє в особистий кабінет, також це трапиться, якщо користувач не авторизований і спробує запропонувати або піти на зустріч

Особистий кабінет, який має таку структуру:

На лівій частині сторінки 2 вкладки: «про мене і мої зустрічі».

На решті частини сторінки мінливий контент в залежності від того що вибрав користувач, якщо:

- Про мене – Форма, де користувач може змінити ім'я, прізвище, електронну адресу, номер телефону та інформацію про себе.

- Мої зустрічі - таблиця с усіма зустрічами користувача де він є власником або куди вій йде. Якщо користувач власник зустрічі, то в нього є можливість редагувати зустріч.

На головній сторінці кнопки «Знайти зустріч» та «Запропонувати зустріч», які при натисканні дають перехід на відповідні сторінки.

На сторінці зустрічей відкривається список зустрічей, де є фільтрація, при натисканні на зустріч користувач переходить в деталі зустрічі.

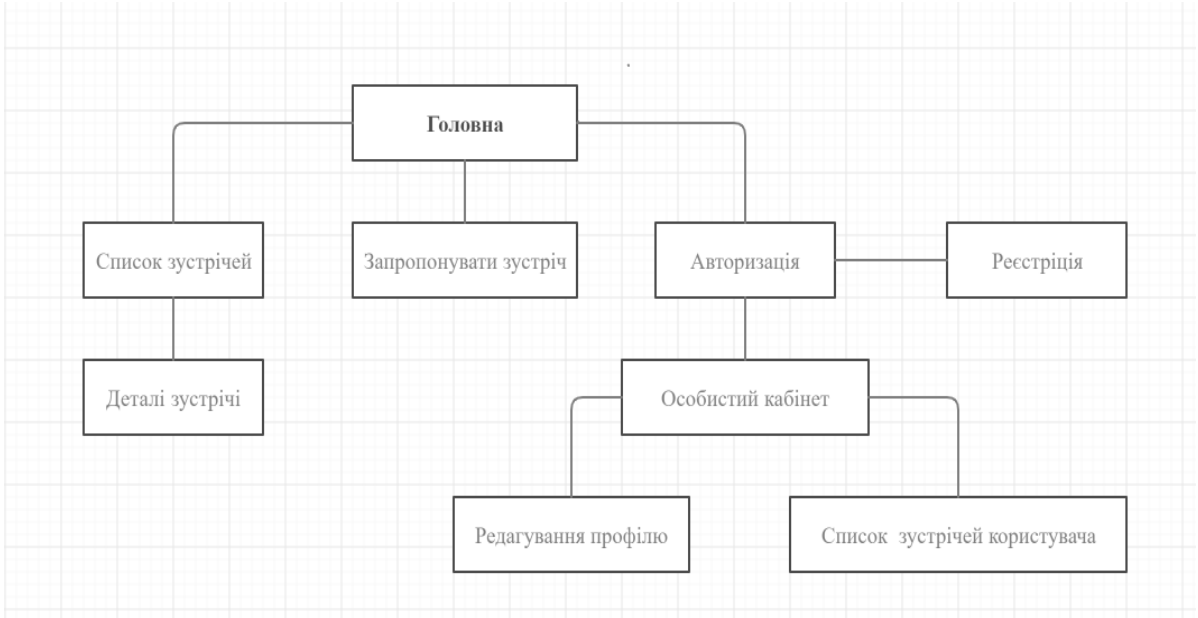

Рисунок 2.1 – Структура веб-додатку

Таким чином підготовлена структура веб-додатку для реалізаціїї проекту.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Розроблення інтерфейсу

Інтерфейс сайту – це вмонтований в ресурс механізм взаємодії з користувачем, коли той може певним чином діяти на сайті, активно користуватися його сервісами та службами (запитувати і додавати інформацію, робити замовлення, залишати заявки, заповнювати анкети і т.д.). Основним елементом інтерфейсу є так звана форма з полями для введення інформації, прапорцями вибору, кнопками виконання. «Дружній інтерфейс» – означає, що зовнішній вигляд ресурсу має відвідувача до себе, а його механізм взаємодії зрозумілий, призначений для користувача сервіс попереджувальний і доброзичливий, система чітко дає інструкції і підказки і т.д.

Часто інтерфейс порівнюють з дизайном сайту. Проте, веб-дизайн сайту - це графічне «обличчя» ресурсу, а інтерфейс - це механізм, якою користувач спілкується з системою.

Саме створення «дружнього інтерфейсу» (ергономічного, зрозумілого, легкого і простого у використанні) є основою для створення повноцінного ресурсу.

Критерії, що пред'являються до інтерфейсу сайтів:

1) природність (інтуїтивність) інтерфейсу. Робота з сайтом у користувача не повинна викликати складнощів при пошуку необхідної інформації (елементів інтерфейсу) для управління процесом вирішення завдання;

2) несуперечність інтерфейсу. Якщо в процесі взаємодії з сайтом користувачем були застосовані деякі прийоми роботи в системі, то в іншій частині ресурсу ці прийоми повинні бути подібні. Таким чином, інтеграція з інтерфейсом повинна відповідати звичним нормам (наприклад, використання клавіші Esc);

3) ненадлишковість інтрефейсу. Ненадмірність увазі громадності інформації, тобто користувачеві необхідно вводити мінімальну інформацію

для роботи з системою. Не потрібно вимагати від користувача ввести інформацію, яка була раніше введена або яка може бути логічно отримана (рис. 3.2, 3.3);

4) прямий доступ до системи допомоги. При роботі з сайтом, необхідно щоб він надав користувачеві зрозумілу і просту систему допомоги. Вона повинна відповідати наступним вимогам: якість оброблюваних команд, характер інформування про помилки і підтвердження виконуваних в даний момент сайтом команд;

5) гнучкість інтерфейсу. Інтерфейс сайту повинен бути зрозумілий людям з різними навичками поводження з ПК - як любителям, так і професіоналам. Для недосвідчених користувачів система може представляти ієрархічну структуру меню, а для досвідчених - управління за допомогою комбінації клавіш;

6) логічність інтерфейсу. Запит інформації, що стосується одного логічного блоку, має сенс об'єднати і структурувати. Наприклад, якщо необхідно ввести дані по декільком клієнтам, то необхідно запитувати всю інформацію разом, і прізвище, і номер поліса (рис. 3.4);

7) ергономіка інтерфейсу. Ергономіка інтерфейсу сайту означає мінімізацію розумових і фізичних зусиль користувача ресурсу. А саме - якщо є можливість замінити введення вибором, це необхідно реалізувати (рис. 3.5).

Рисунок 3.1 – Розробка сторінки авторизації

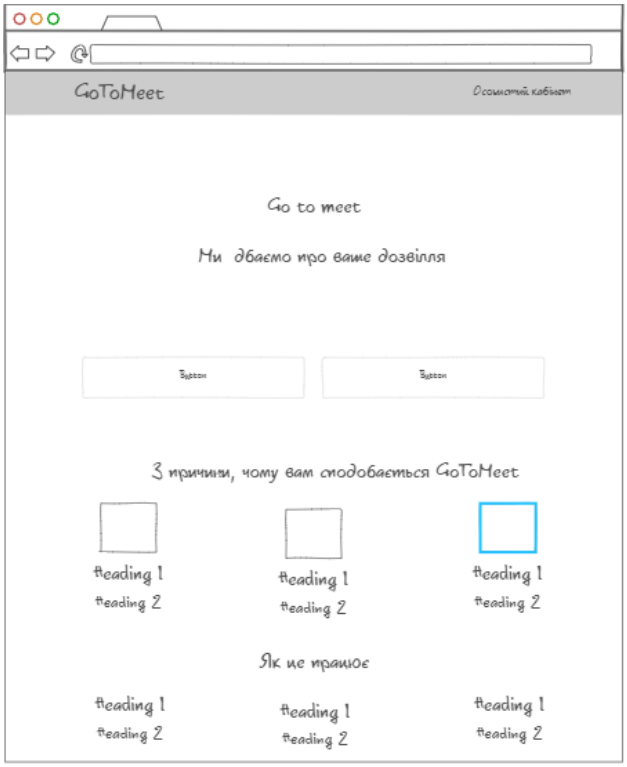

Рисунок 3.2 – Розробка інтерфейсу головної сторінки

Рисунок 3.3 – Розробки сторінки списку зустрічей

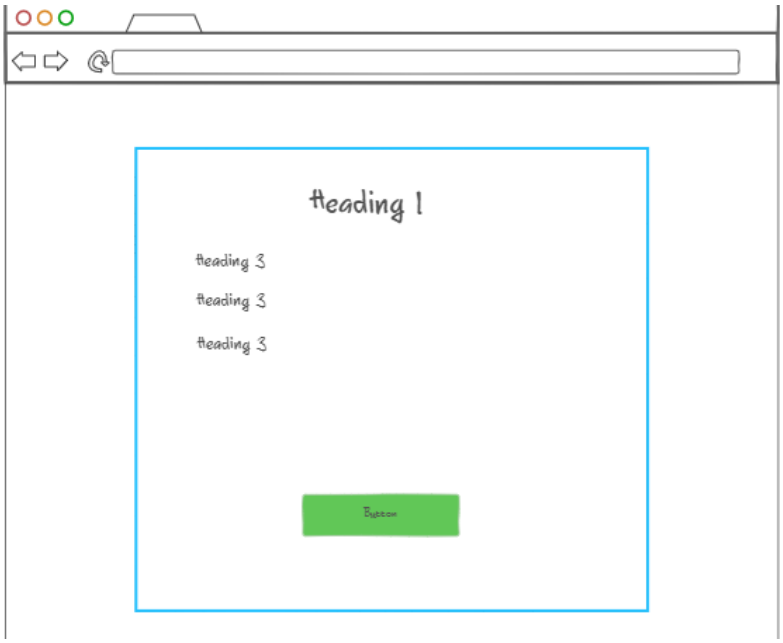

Рисунок 3.4 – Розробки сторінки деталей зустрічі

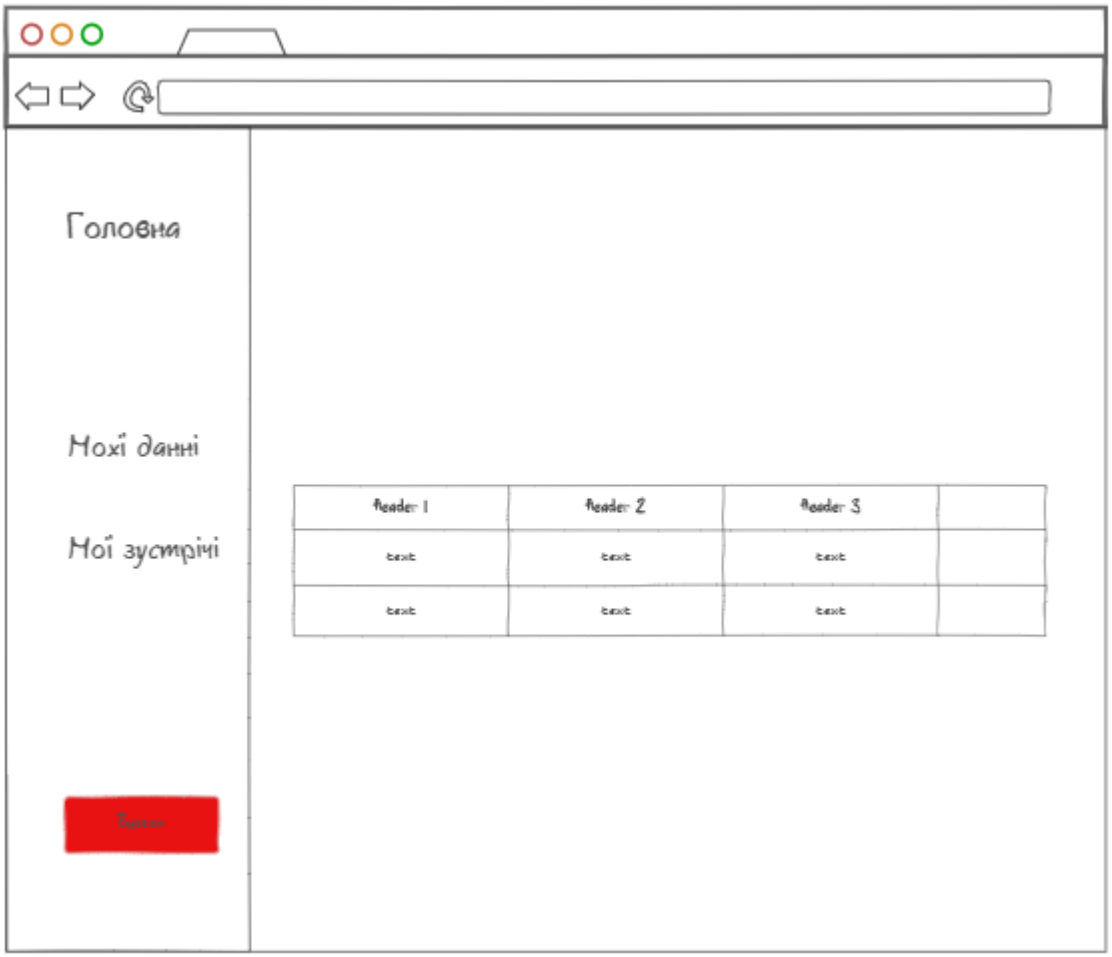

Рисунок 3.5 – Розробка особистого кабінету

3.2 Тестування веб-додатка

Важливою частиною любого веб-сервісу або веб-додатку є тестування. Дуже важливо зробити тестування додатків на потенційні помилки, використовуючи різні методики перед запуском або розгортанням.

Кожна веб-програма відрізняється одна від одної щодо деяких функцій, таких як розмір, складність, технологія та корпоративна політика, але ці фази або етапи тестування можуть бути застосовані до різних додатків залежно від вимог веб-тестування.

Обов’язковий крок для перевірки роботи системи відповідно до специфікацій або функціональних вимог, які користувач призначив для неї. Також тестування зручності використання веб-додатків тепер стає важливою

частиною будь-якого веб-проекту. Він включає тестування візуальних та текстових елементів.

Під час тестування веб-сервісу було перевірено на коректність роботи веб-сервісу. Здійснювалися такі перевірки:

1) додавання зустрічі з не авторизованого користувача. Оскільки для того щоб додати зустріч або піти на неї потрібно бути авторизованим (рис. 3.6);
2) перегляд на різних моніторах (планшет, мобільний телефон, екран ноутбуку та монітор) (рис. 3.7, 3.8);
3) відображення в особистому кабінеті тільки своїх зустрічей. Перевірено з двох різних аккаунтів та створено 2 зустрічі (тест та тест2). Зайшовши на перший аккаунт, на сторінці «Мої зустрічі», відображена та, яка була створена с цього аккаунту (рис. 3.9).

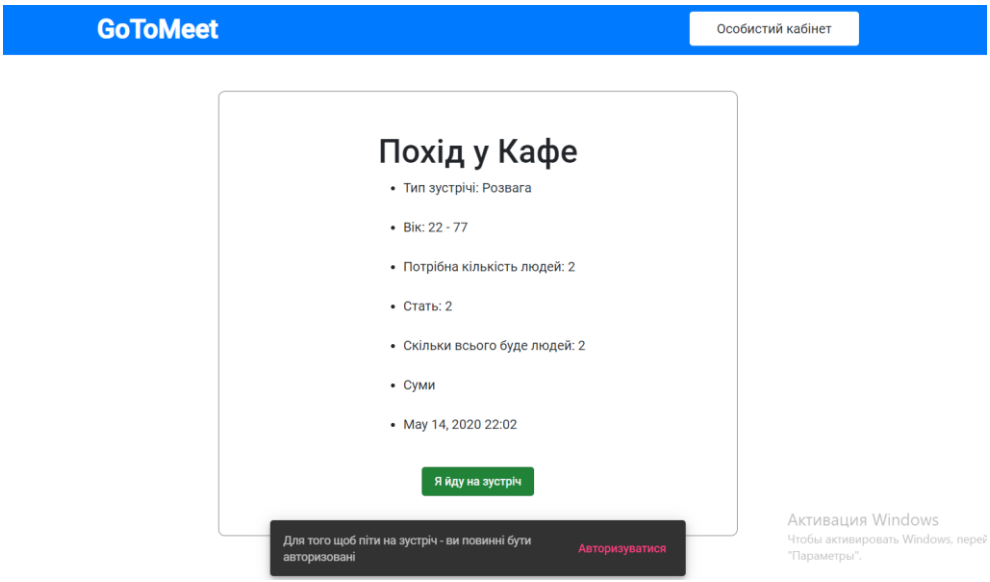

Рисунок 3.6 – Перегляд веб-ресурсу

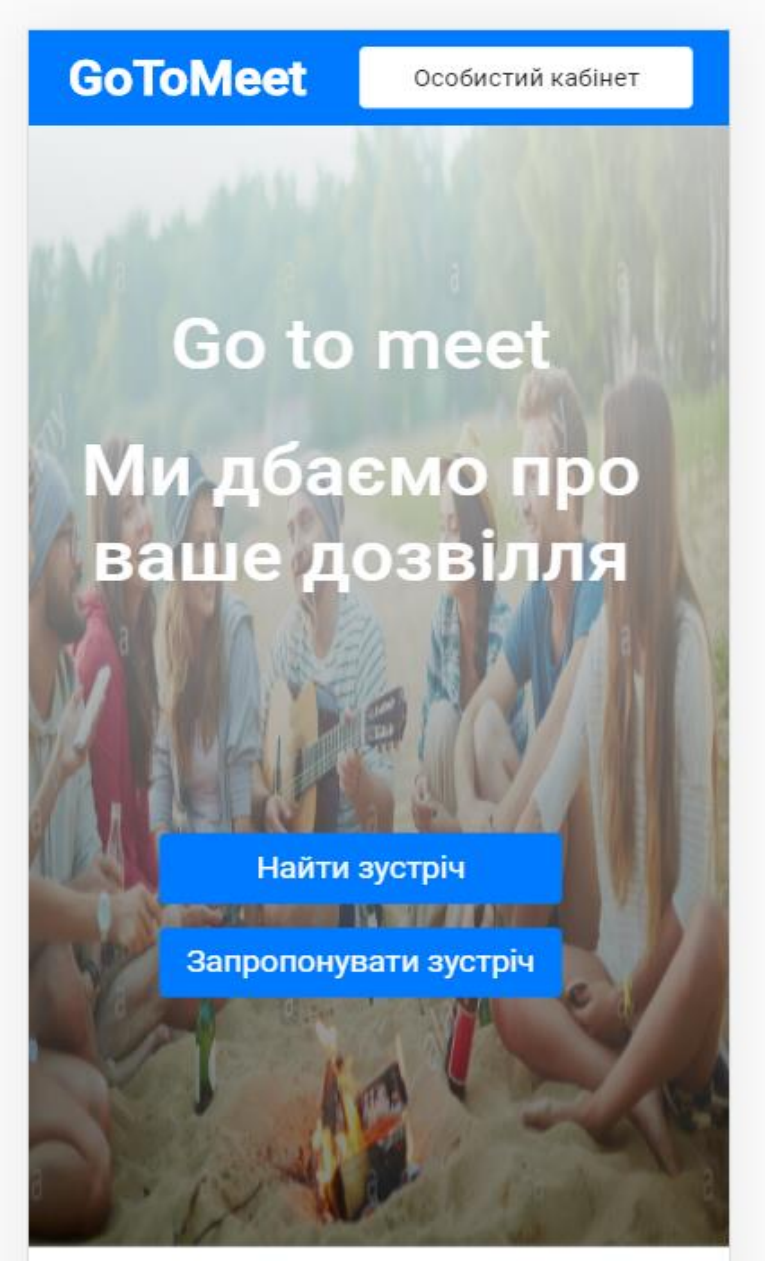

Рисунок 3.7 – Мобільна версія головної сторінки

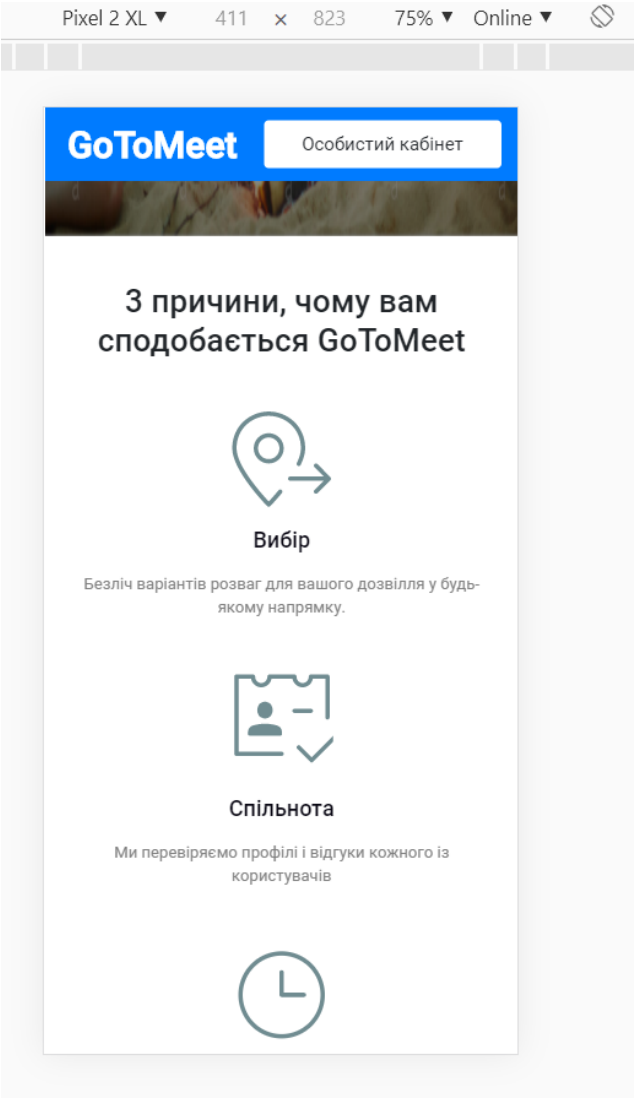

Рисунок 3.8 – Мобільна версія

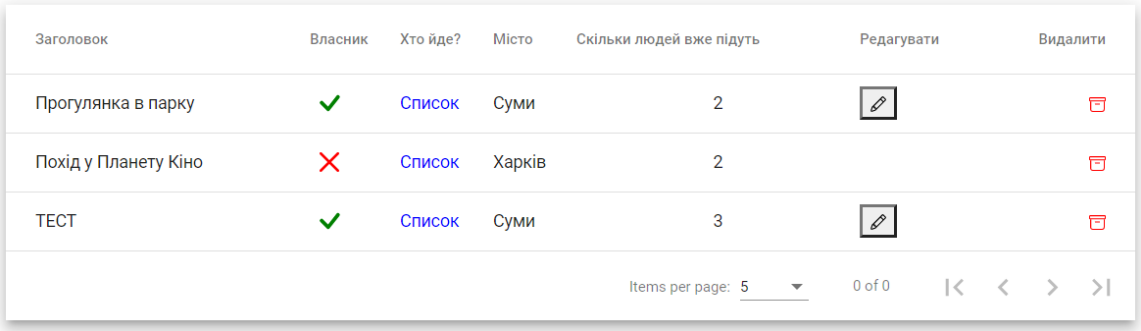

Рисунок 3.9 – Таблиця зустрічей користувача

3.3 Інструкція з використання веб-додатка

Перше, з чим зустрічається відвідувач, потрапивши на сайт – це головна сторінка, що надає користувачеві основні відомості, такі як тематика веб-сайту

та матеріали, які можна побачити на подальших сторінках. На початковій сторінці даного веб-сервісу є такі пункти меню як особистий кабінет, можливість знайти або запропонувати зустріч (рис. 3.10).

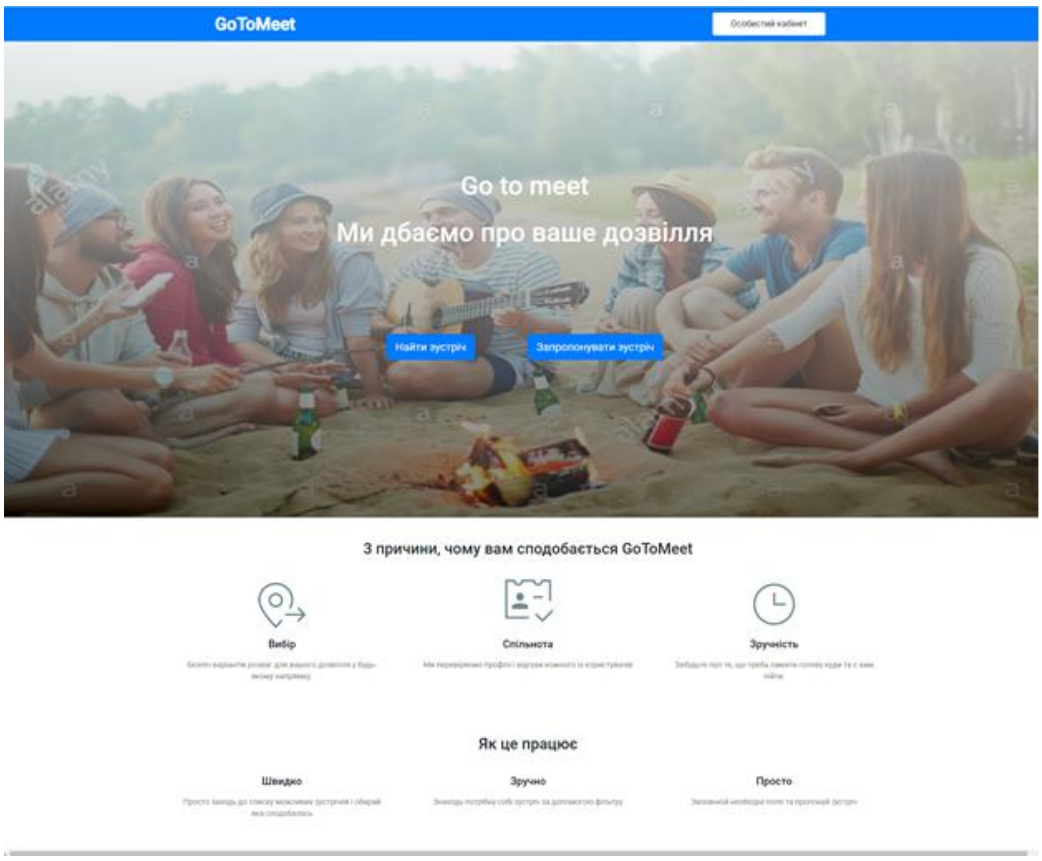

Рисунок 3.10 – Перегляд відгуків

При переході на сторінку «Особистий кабінет» користувач має змогу редагувати свої дані: своє прізвище, ім’я та номер телефону.

Якщо користувач натисне на кнопку «Найти зустріч», то він перейде на список доступних зустрічей, які будуть вже відфільтровані за віком користувача (рис. 3.11).

Користувач може відфільтрувати зустрічі за типом зустрічі (спортивна, прогулянка, кіно), статтю та кількістю людей. Щоб дізнатися деталі будь-якої зустрічі треба натиснути на одну з них (рис. 3.12). Кнопка «Я йду на зустріч» означає, що буде відправлене запрошення в особистий кабінет власника оголошення. Якщо користувач не авторизований, то його перенаправить на сторінку авторизації (рис. 3.13).

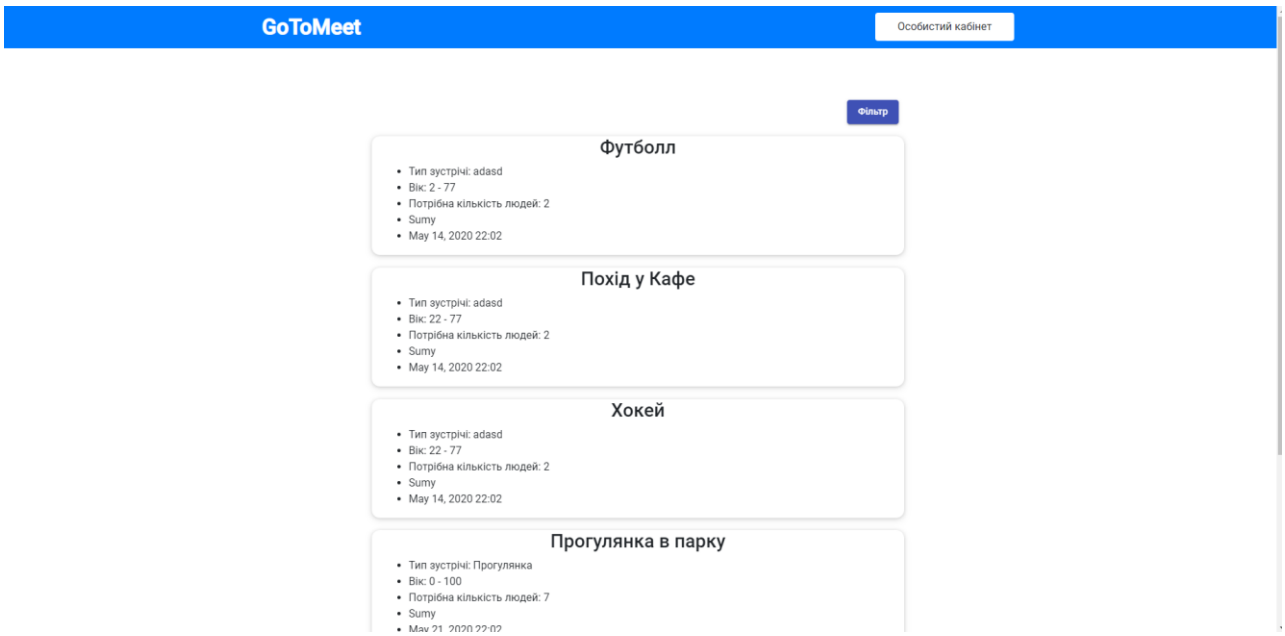

Рисунок 3.11 – Перегляд варіантів для зустрічі

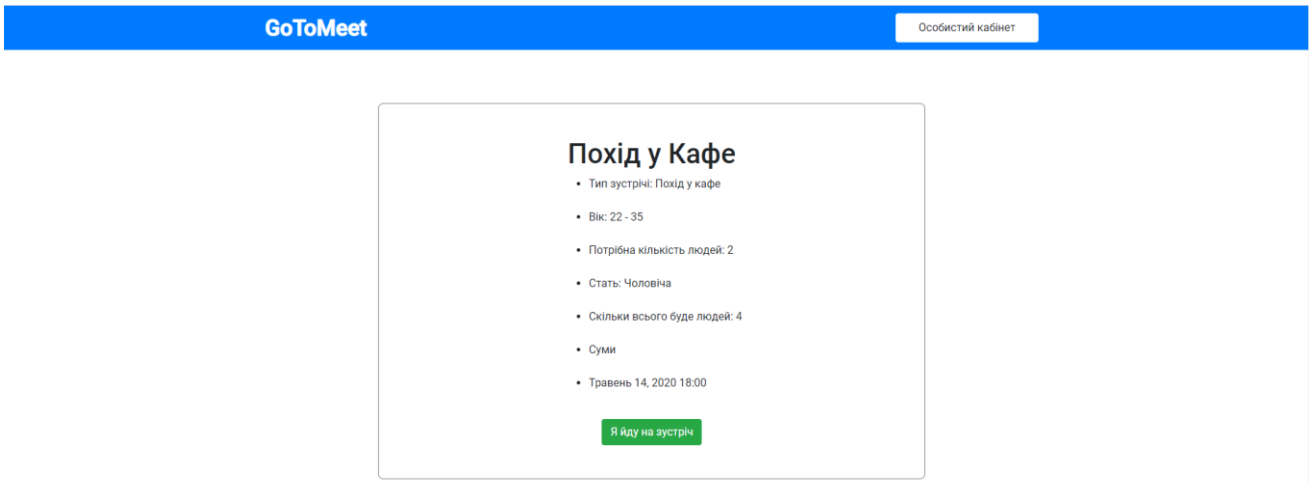

Рисунок 3.12 – Деталі зустрічі

Якщо користувач не має акаунт, він повинен натиснути кнопку «Не маєте аккаунту?» після чого його перенаправить на сторінку реєстрації, де буде потрібно ввести ім’я, прізвище, електронну адресу, пароль, номер телефону, стать та свій вік. Якщо хоча б одне поле буде пустим, то система не дасть йому зареєструватися. При вдалій реєстрації, користувач перейде на сторінку авторизації, де необхідно ввести свої данні, якщо вони будуть не вірними, то користувач не зможе увійти до системи (рис. 3.14, 3.15, 3.16).

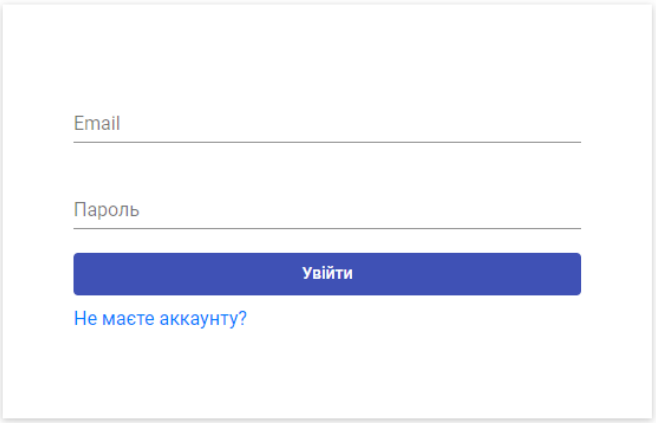

Рисунок 3.13 – сторінка авторизації користувача

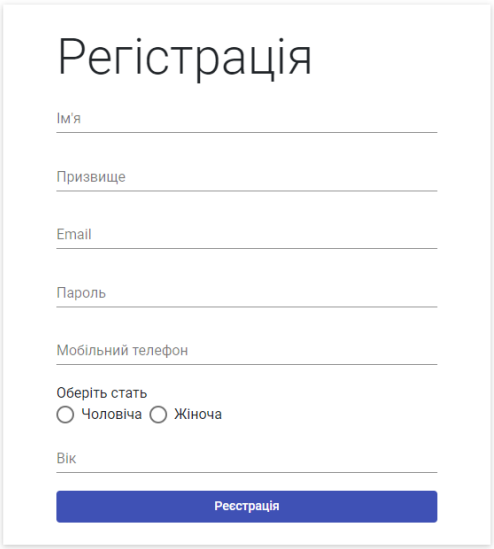

Рисунок 3.14 – Сторінка реєстрації користувача

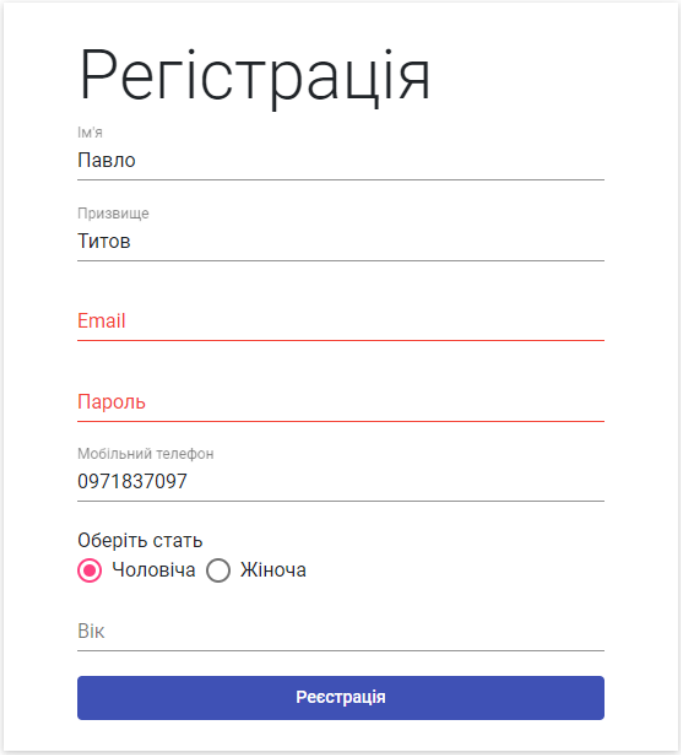

Рисунок 3.15 – Перевірка на коректність введення даних при реєстрації

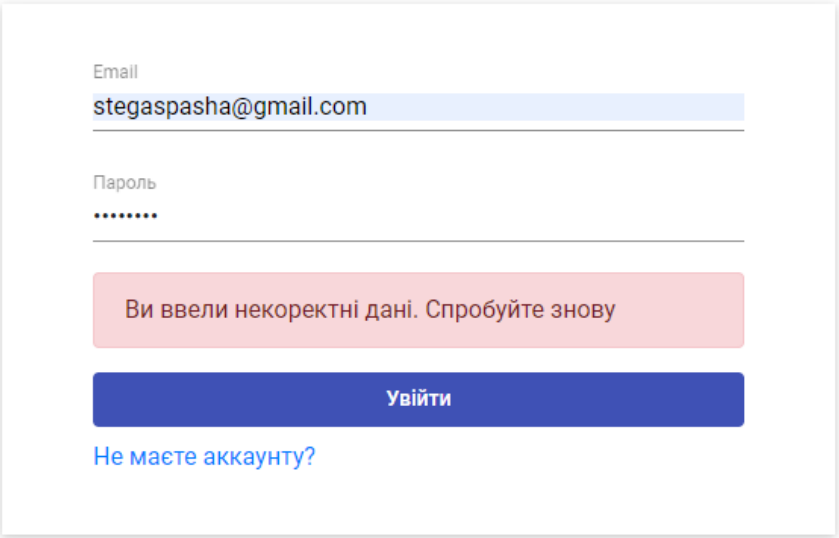

Рисунок 3.16 – Перевірка на коректність введення даних при авторизації

Після введення коректних даних, користувач потрапить до особистого кабінету, де буде інформація про нього та його зустрічі (рис. 3.17).

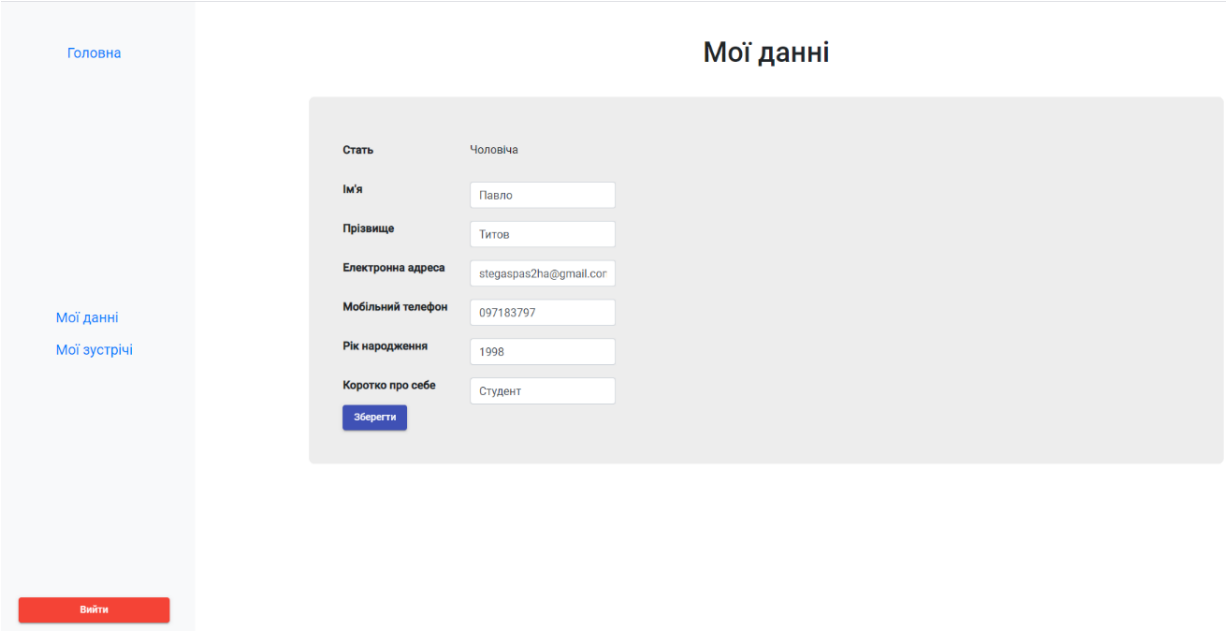

Рисунок 3.17 – Особистий кабінет користувача

Якщо користувач є власником зустрічі, то у нього є можливість редагувати зустрічі (рис. 3.18).

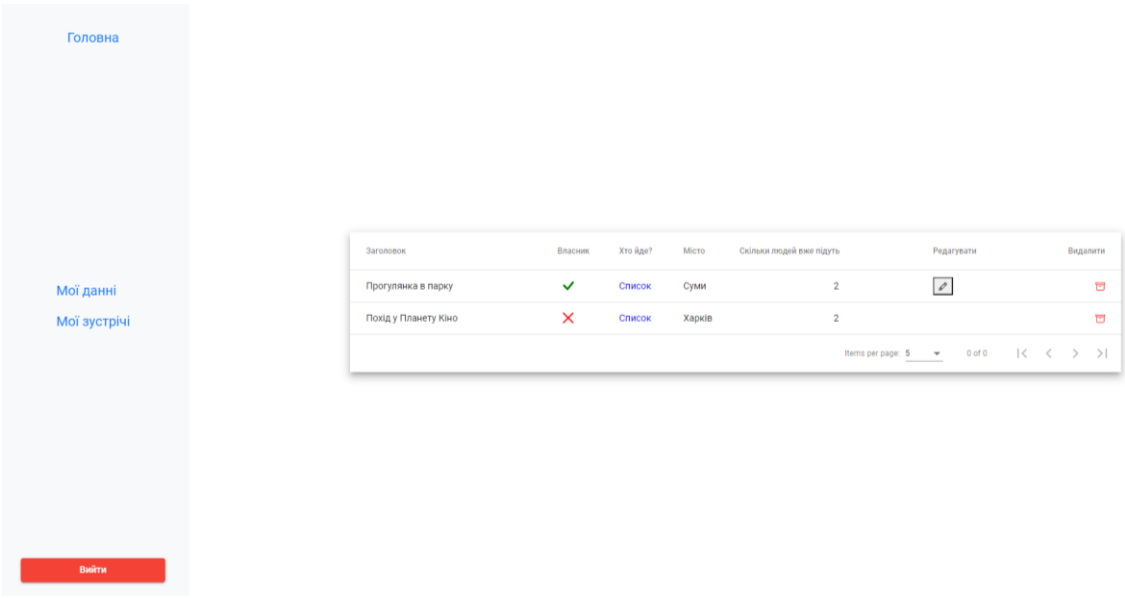

Рисунок 3.18 – Особистий кабінет користувача

Коли користувач натисне на кнопку «список», то побачить список людей хто погодився, а також матиме можливість приймати людей для зустрічі або відхиляти (рис. 3.19).

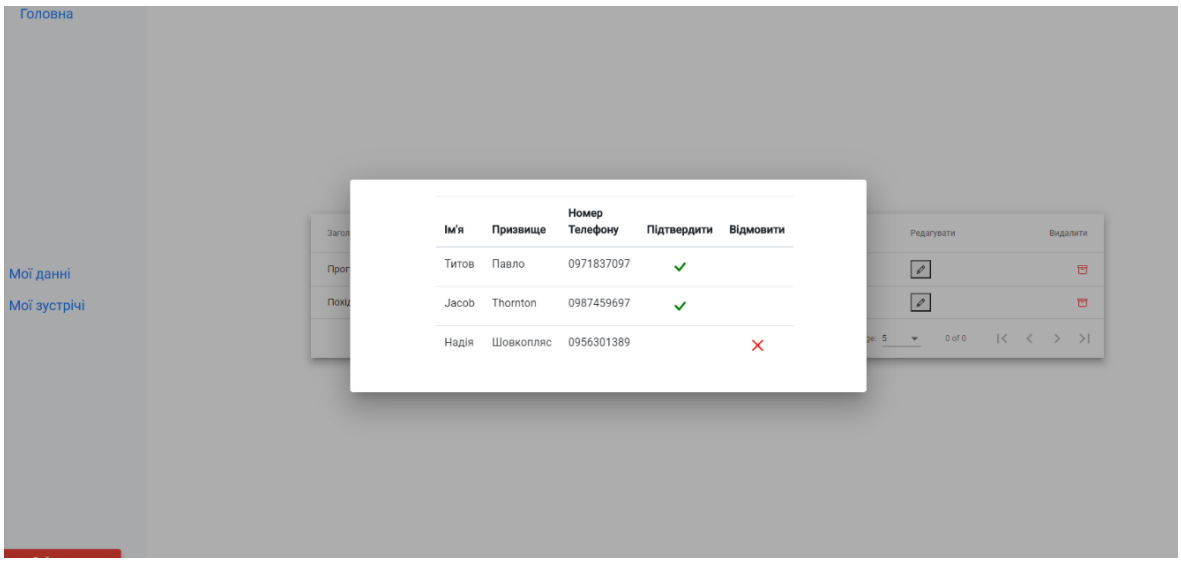

Рисунок 3.19 – Список учасників зустрічі власника зустрічі

Якщо користувач не є власником зустрічі, то у нього доступна інформація лиш ім’я, прізвище та телефон (рис. 3.20).

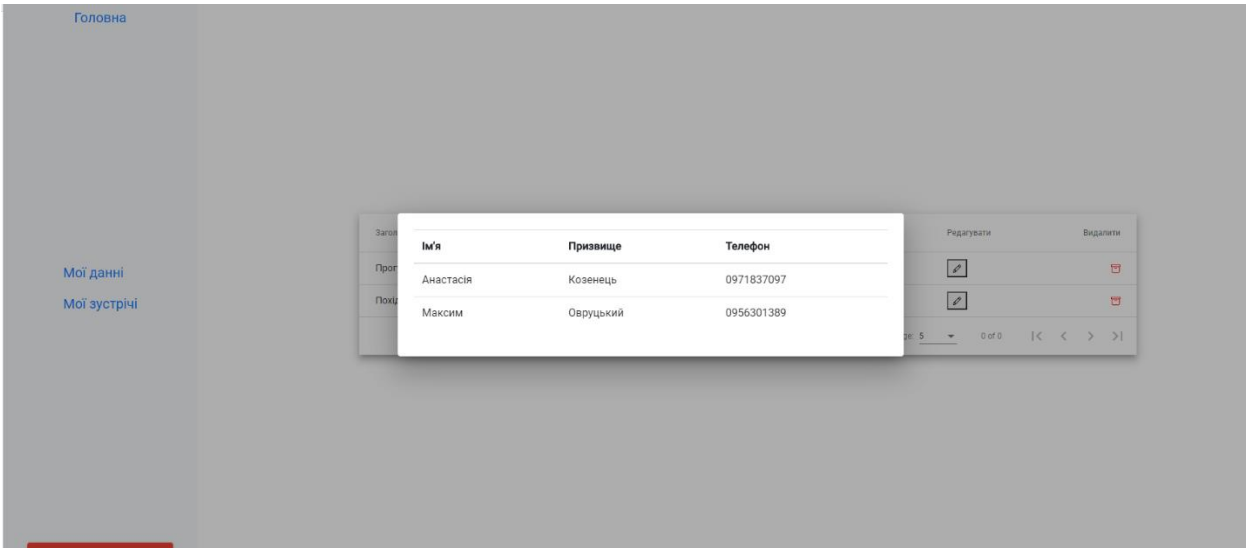

Рисунок 3.20 – Список учасників зустрічі

Якщо користувач захоче повернутися на головну сторінку, то потрібно натиснути кнопку «Головна», а щоб запропонувати зустріч – потрібно натиснути кнопку «Запропонувати зустріч» (рис. 3.21).

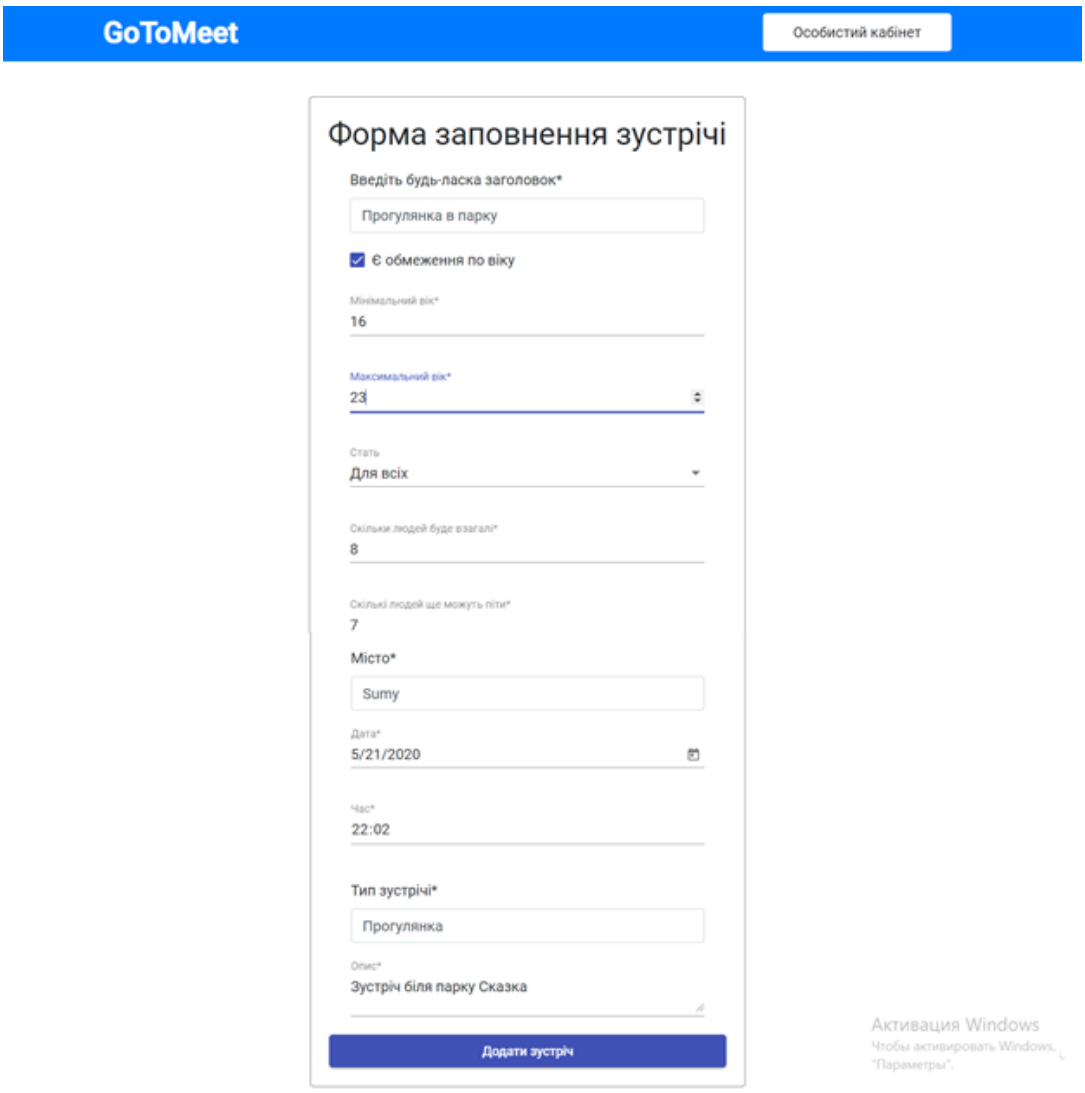

Рисунок 3.21 – Форма заповнення зустрічі

Тестування розробленої інформаційної системи показало, що програмний продукт працює корректно.

ВИСНОВКИ

Під час досліджень було розробено веб-сервіс для проводження вільного часу, а саме:

1) було проведено аналітичний огляд області розробки і визначена актуальність веб-сервісу

2) розроблено структуру веб-сервісу;

3) створенний інтерфейс веб-сервісу;

4) проведено тестування веб-сервісу.

Web-сервіс містить наступні функціональні можливості:

1) реєстрація користувачів;

2) налаштування облікових записів користувача;

3) перегляд списку зустрічей;

4) бронювання зустрічей;

5) можливість залишати відгуки про користувача;

6) фільтрація зустрічей: за типом зустрічі, віком, місцем, кількістю людей, тощо;

7) має адаптивність сайту для ПК, планшетів і мобільних телефонів;

Веб-сайт був розроблений на фреймворці Angular, який вважається найкращим фреймворком JavaScript для розробки сайтів, що можуть розширюватися або мають складну структуру. Використовували базу даних MongoDB яка є швидка, зручна та масштабована. Для бекенду ми використовували Express Node.js , що має обширний набір функцій для веб-сервісів, а також протокол HTTP для надійного API.

СПИСОК ЛІТЕРАТУРИ

1 Мови та технології програмування - https://www.metanit.com

2 Розробка web-сервісу для надання побутових послуг - https://core.ac.uk/download/pdf/156948657.pdf

3 Фреймворки JavaScript - https://medium.com/javascript-scene/top-javascript-frameworks-and-topics-to-learn-in-2020-and-the-new-decade-ced6e9d812f9

4 Довідник Angular - https://www.angular.io/

5 Мови та технології програмування - https://www. infoworld.com

6 Бази данних - http://schoollib.com.ua/dovidnyk_shkolyara/informatyka

7 Бази данних - www.computerhope.com

8 Codd E. F. Relational completeness of data base sublanguages. - Ibid. 1972, p. 65—98.

9 Найкращі CSS фреймворки - https://geekbrains.ru/posts

10 Архітектура REST - https://habr.com/ru/post/38730/

11 HTML - https://www.w3.org/MarkUp/html-spec/html-essay.html

12 What is JavaScript? - https://www.infoworld.com/article/3441178

13 JavaScript https://www.123helpme.com/topics/javascript

14 Essay On HTML - https://www.bartleby.com/essay/Essay-On-HTML

15 Type of WebSite - https://www.hostgator.com/blog/popular-types-website

ДОДАТОК А

Код програмного додатка

```
const express = require('express');
const config = require('config');
const mongoose = require('mongoose');

const app = express();

app.use(express.json({extended: true}));

app.use('/api/auth', require('./routes/auth.routes'));
app.use('/api/meet', require('./routes/meet.route'));

const PORT = config.get('port') || 5000;

async function start() {
     try {
          await mongoose.connect(config.get('mongoUrl'), {
               useNewUrlParser: true,
               useUnifiedTopology: true,
               useCreateIndex: true
          });
          app.listen(PORT, () => console.log('App has been
started'));
     } catch (e) {
          console.log('Server Error', e.message)
          process.exit(1);
     }
}

start()


     const {Router} = require('express');
```

```
const Meet = require('../models/Meet');
const auth = require('../middleware/auth.middleware');
const shortid = require('shortid');
const router = Router();

router.post('/generate', auth, async (req, res) => {
try {
     const code = shortid.generate();
     const _id = new Date().valueOf();
     const {title, minAge, maxAge, meetingDate,
meetingTime, comment, countPeople, needCountPeople, sex,
meetingCity, meetingType} = req.body;
     if (minAge > maxAge) {
          return res.status(400).json({ message:
'Перевірте вікові обмеження' })
     }

     if (needCountPeople > countPeople) {
          return res.status(400).json({ message:
'Перевірте потрібну і загальну кількість людей' })
     }
     const meet = new Meet({
          _id,title, minAge, maxAge, meetingDate,
meetingTime, owner: req.user.userId, comment, countPeople,
needCountPeople, code, sex, meetingCity, meetingType
     });
     await meet.save();
     res.status(201).json({ meet })
} catch (e) {
     res.status(500).json({ message: 'Щось пішло не
так, спробуйте знову' })
}
});
router.get('/my', auth, async (req, res) => {
try {
     const meet = await Meet.find({ owner:
req.user.userId });
     res.json(meet)
} catch (e) {
     res.status(500).json({ message: 'Щось пішло не
так, спробуйте знову' })
}
});

router.get('/', async (req, res) => {
try {
     const meet = await Meet.find();
     res.json(meet)
} catch (e) {
     res.status(500).json({ message: 'Щось пішло не
так, спробуйте знову' })
}
```

```
});

router.get('/:id', auth, async (req, res) => {
try {
     const meet = await Meet.findById(req.params.id);
     res.json(meet)
} catch (e) {
     res.status(500).json({ message: 'Щось пішло не
так, спробуйте знову' })
}
});

module.exports = router;

import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { SexSelect } from '../models/sexSelect';
import { Router } from '@angular/router';
import { OfferMeetingService } from './offer-
meeting.service';

@Component({
 selector: 'app-offer-meeting',
 templateUrl: './offer-meeting.component.html',
 styleUrls: ['./offer-meeting.component.scss']
})
export class OfferMeetingComponent implements OnInit {
  public form: FormGroup;

  error: any;
    limitAge = false;

  sex: SexSelect[] = [
    { value: 0, viewValue: 'Для всіх'},
    { value: 1, viewValue: 'Тільки для чоловіків'},
    { value: 2, viewValue: 'Тільки для жінок'}
  ];

  constructor(
    private formBuilder: FormBuilder,
    private router: Router,
    private offerService: OfferMeetingService
  ) { }
  ngOnInit() {
    this.initForm();
  }
   initForm() {
    this.form = this.formBuilder.group({
      title: [ [], Validators.required],
      minAge: [0],
      maxAge: [100],
```

```
            sex: [0],
            countPeople: [[], Validators.required],
            needCountPeople: [[], Validators.required],
            meetingCity: ['Sumy', Validators.required],
            meetingDate: [[], Validators.required],
            meetingTime: [[], Validators.required],
            meetingType: [[], Validators.required],
            comment: [[], Validators.required]
          });
        }

        add() {
          if ( this.form.valid) {
            this.offerService.add(this.form.value).subscribe(
              // () => this.router.navigate(['/']),
              () => {},
              (error) => {
                this.error = error
              }
            )
          } else {
            this.error.error.message = 'Перевірте заповненість
всіх полів'
          }
        }
          changeLimitAge() {
          this.limitAge = !this.limitAge;
        }}
```