Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

АВТОРЕФЕРАТ
випускної кваліфікаційної роботи бакалавра

«СИСТЕМА УПРАВЛІННЯ РЕЗЕРВУВАННЯМ РЕСУРСІВ (ЗА ПРИКЛАДОМ СЕРВІСУ BOOKING) (FRONTEND)»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту

№	П.І.Б.	Група
1	Подельнік Дмитро Ігорович	КН-П-181
2	Алькатеб Хайдер	КН-П-181
3	Сігал Анастасія Русланівна	КН-Д-182

Автор звіту

_____ Алькатеб Хайдер

РЕЗУЛЬТАТИ ЕКСПЕРТИЗИ

Експерт	П.І.Б.	Оцінка	Підпис
Ментор	Черкезян Крістіне Арутюнівна		
Рецензент	доц. Суліма Миколай Миколайович		
ДЕК (захист проекту)	доц. Голова Е. К.		

Одеса – 2022

АНОТАЦІЯ

Причиною проекту стало створення нової системи онлайн-бронювання. Ця система ґрунтується на існуючій системі booking.com, а функціональні можливості системи та інтерфейс користувача модернізовані.

Розробка нової системи розпочалася з нового переліку умов від керівників департаментів та з урахуванням вступної ідеї чинної системи щодо booking.com.

Робота присвячена дослідженню, розробці та тестуванню системи бронювання, основною метою якої є легка торгівля між користувачами та компаніями. Функції системи включають в себе: реєстрацію, авторизацію, аутентифікацію, оренду, отримання інформації про певні регіони / країни, обмеження доступу до контенту певних ролей користувачів, а також ведення журналів діяльності системи.

Основними задачами були поліпшення зручності використання інтерфейсу системи. Об'єктом дослідження в рамках даної роботи є алгоритм збору комплексної оцінки результатів численних критеріїв з точки зору недостатньої визначеності початкових умов. Предметом дослідження є фактор, що дозволяє користувачеві подорожувати і об'єднувати об'єкти і польоти найпростішим способом, який тільки можна собі уявити.

У якості цілей дослідження виступають:

- переглянути системи бронювання, які дозволяють користувачам взаємодіяти з іншими користувачами та компаніями без необхідності йти через будь-які непрості кроки.
- розробити програмне забезпечення, реалізувавши результати, досягнуті в попередніх завданнях, провести тестування в різних умовах і для різних пристроїв.

- впровадити програмне забезпечення для захисту даних у систему для підвищення показників інформаційної безпеки створеної системи.
- методи дослідження, що використовуються для досягнення цілей, формуються суперпозицією методів системного аналізу, моделювання та когнітивної методології.

Для розробки на стороні сервера використовувалися Microsoft SQL Server і мова програмування C#, а для розробки на клієнтській стороні використовувався Angular Web Framework з HTML, JavaScript і CSS. Ця дослідницька робота зосереджена на тому, як створити систему бронювання для певного фільтру пошуку.

ЗМІСТ

ВСТУП

Все більше і більше людей використовують комп'ютери та мобільні пристрої як частину свого повсякденного життя з моменту швидкого зростання Інтернету за останні рока. Бронювання послуг є одним з цих завдань.

На ринку систем бронювання домінують кілька розробників програмного забезпечення та компаній, і є багато ситуацій, коли вони корисні. Систему бронювання можна знайти скрізь, в ресторанах, лікарнях, перукарнях, школах та в інших містах. Системи бронювання можуть бути дуже складними, оскільки існує так багато варіацій пристроїв та уподобань користувачів, що тягне за собою складність розробки.

Дуже гнучка система бронювання часто невиправдано складна для користувача та адміністрації. Результатом є непотрібні кліки на стороні адміністрування, при цьому вмикаються функції, які не потрібні. Системи бронювання повинні бути логічними і простими у використанні як для ринку, так і для користувача.

У документі розглядаються проблеми, з якими ми зіткнулися при розробці програмного забезпечення, а також описується повний цикл цієї розробки.

Звіт складається з трьох розділів, в яких описуються загальні характеристики системи, деталі створення клієнтської частини, а також результати її розгортання і тестування. Висновки відображають результати, досягнуті в цій частині проекту, загальний висновок - сукупність звітів всіх учасників проекту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

СИСТЕМА УПРАВЛІННЯ РЕЗЕРВУВАННЯМ РЕСУРСІВ
(за прикладом сервісу Booking.com)

Загальний опис.

Програмне забезпечення (ПЗ) призначене для створення розподіленої системи керування ресурсами, яке передбачає можливість проактивного управління, зокрема, резервування. Функції системи мають забезпечити повний життєвий цикл управління ресурсами: створення, актуалізація, використання та видалення (вживання). Головна особливість системи полягає у тому, що пріоритет використання обмежених ресурсів знаходиться у залежності від терміну його резервування.

Призначення:

ПЗ орієнтується на створення контейнеру управління обмеженими ресурсами на основі аналізу їх резервування. Оброблення конкурентних запитів на обмежені ресурси має враховувати показники пріоритетності, одним з яких є часовий показник моменту резервування. ПЗ створюється за прикладом сервісу бронювання обмеженої кількості місць, у той же час має складатись за принципом універсальності, мати змогу змінити цільове призначення.

Вимоги до функціоналу. Обов'язкова частина

Головна сторінка

Стартовий інтерфейс ПЗ (головна сторінка або активність) має за мету популяризацію системи. На ньому має бути забезпечено відображення популярних показників як-то загальна кількість користувачів та інформаційних одиниць, вибірка з найбільш популярного контенту та нових

дописів, зовнішня реклама. Також головна сторінка має містити засоби авторизації та реєстрації або посилання на них.

Реєстрація користувачів:

Користувач ПЗ має змогу зареєструватись як за допомогою авторизаційних даних (логіну/паролю), так і за допомогою облікових записів популярних соціальних мереж (Google, Facebook, тощо).

При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - телефон та електронна пошта. На тестовому етапі, за умови ускладнення замовлення послуг телефонії чи СМС, можливе використання лише електронної пошти. Засіб оперативного контактування при реєстрації підлягає верифікації через відповідь на тестовий виклик чи надісланий лист, або введення коду, пересланого через повідомлення довільного типу. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

За умови що користувач забув пароль, ЗОК має бути використаний для його відновлення. Схема відновлення паролю жорстко не регламентується.

Також передбачається наявність у системі користувачів з додатковими правами: модераторів та адміністраторів. Реєстрація кореневого адміністратора системи має бути реалізоване при інсталяції системи, наступним користувачам змінюють права наявні адміністратори.

Авторизація та особистий кабінет користувача:

За фактом успішної авторизації користувачу надається доступ до особистого кабінету. Це окремий інформаційний ресурс (сторінка), що являє собою основне робоче місце користувача ПЗ. Повинен забезпечити усі необхідні засоби для

- Пошук ресурсів за метаданими, групування та фільтрування результатів за різними ознаками, наявними у знайденому контенті. Для об'ємних результатів має бути забезпечена пагінація.
- Створення замовлень на вільні ресурси
- Реєстрація (підписування) на повідомлення щодо зміни статусу ресурсів
- Перегляд поточної активності: історії пошуку, перелік відзначеного контенту (вибране), контактні особи (друзі), тощо
- Управління власними замовленнями: перегляд, пошук, підтвердження, відмова

Також засобами особистого кабінету має бути передбачено можливість зміни персональних даних.

Зворотній зв'язок

Користувач має змогу встановити власну оцінку (лайк або бал) для спожитого ресурсу чи його власника (надавача). Засоби пошуку, фільтрування та групування на інших сторінках мають забезпечити врахування рейтингу контенту.

Вимоги до функціоналу. Опціональна частина

Привілейовані функції

З метою покращення економічної ефективності проекту передбачається наявність додаткових функцій системи, що доступні лише за умови платної підписки (або внутрішніх досягнень) користувачів. До таких функцій можна віднести відображення ЗОК інших користувачів (для прямого контактування), розширені засоби фільтрації та групування. Також можливе встановлення обмеження щодо кількості опублікованих одиниць контенту.

Модерація

Для забезпечення дотримання загальноприйнятих норм спілкування, а також з метою недопущення публікування контенту, який містить заборонені компоненти (ненормативні висловлювання, заклики до протиправних дії, приниження чи дискримінація, тощо), усі інформаційні одиниці, а також створені користувачами метадані (маркери) мають в обов'язковому порядку пройти погодження (модерацію). Право надати погодження належить кореневому адміністратору системи, а також модераторам. При погодженні інформаційна одиниця має зберігати ідентифікатор користувача, який надав погодження.

Архітектура

ПЗ має бути спроектоване за розподіленою клієнт-серверною архітектурою з чітко відокремленими розподіленими вузлами системи:

Mockup - візуальна модель інтерфейсу користувача із зазначенням зв'язків і переходів між різними частинами інтерфейсу.

Backend - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами.

Web App - додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів усіх категорій

Mobile App (опціонально) - додаток, що встановлюється на мобільних пристроях (смартфон, планшет). Дана частина виконується за наявності у команді проекту трьох і більше програмістів.

.

Усі модулі ПЗ (окрім візуальної моделі Mockup) повинні обов'язково включати до свого складу засоби

- тестування функціональності та модульні тести,
- розгортання на новому пристрої (інсталяції) із встановленням початкових таблиць у БД та кореневого користувача-адміністратора

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:

- root
    - Backend
    - Web
    - Mobile
    - Mockup (вихідні файли або посилання на зовнішній проект)

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

Backend - .NET технології із хмарним розміщенням (Azure),

Web - JS, HTML, CSS,або JS фреймворки, зокрема React, Angular

Mobile - Java, або Kotlin, або Flutter

Mockup - визначається ментором з дизайну

Команда проєкту

Для виконання поставлених завдань передбачається робота команди розробників окремо за кожним з модулів проекту.

Мінімальна команда проекту - два учасники, у такому складі один учасник виконує модуль Backend та програмну частину Web, другий - модель Mockup та розміткову частину Web.

Рекомендована команда проекту - три учасники, у такій команді кожен з учасників відповідає за одну з частин: Mockup, Backend, Web.

Дозволяється включення до команди більшої кількості учасників, у такому разі модуль Mobile App набуває обов'язкової реалізації. Також модуль Mockup може бути поділений на моделі Web та Mobile частин.

Звітність

За результатами виконання завдань із створення ПЗ кожен з учасників команди проекту складає власний (індивідуальний) звіт, що складається з практичної частини та автореферату. Практична частина є прямим результатом роботи і подається у вигляді посилання на репозиторій (чи інший URL) або у вигляді архіву на довільному носієві даних. Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

Автореферат містить описову частину, у якій розкриваються основні концепти проекту, деталізуються завдання до виконавця, встановлюється актуальність роботи, її об'єкт та предмет, обґрунтовується вибір технологій та відзначається шлях виконання та ступінь досягнення поставлених задач, посилання на джерела інформації.

РОЗДІЛ 1
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Архітектура платформи складається з таких частин:

«Frontend».

«Design»,

«Backend»,

«FRONTEND»

Імперативним аспектом веб-архітектури є те, що користувальницький досвід і інтерфейс залишаються послідовними протягом усієї платформи. Кожна функціональність повинна бути розроблена з урахуванням цього фундаментального принципу.

Платформа відображає односторінковий додаток, що відповідає максимальної зручності для користувача. Односторінковий додаток - це, по суті, те, що потрібно для того, щоб уникнути оновлення браузера при перезавантаженні, або переході на іншу сторінку.

Один із головних моментів - це те, що клієнтський інтерфейс повинен відповідати сучасності та зручності користування. Цей критерій вважається частиною вимоги узгодженості для програми веб-порталу:

1. Кожна частина платформи повинна використовувати однакові елементи взаємодії користувача.
2. Розташування кожного елемента взаємодії користувача має бути однаковим.
3. Розміри елементів інтерфейсу повинні бути однаковими для кожного випадку використання.
4. Усі елементи повинні відображатися у різних станах в одному стилі.

5. У будь-який час, коли це можливо, для реалізації всіх елементів користувача повинна бути використана заздалегідь визначена колірна схема.

Для виконання цих критеріїв необхідно, щоб розробник фронтенду зобов'язався і ретельно розумів принципи проектування. Однак документування обраних концепцій має вирішальне значення для допомоги адаптаційному процесу.

«DESIGN»

Існує кілька моделей та описів того, як користувачі взаємодіють з інформаційною системою в розділі “Дизайн”. Ці моделі описують рухові шляхи і переходи між окремими інтерфейсами, а також взаємодію між різними видами користувачів. Типовий інтерфейс користувача використовує методи UI/UX і концепції колірної схеми для інтелектуальної навігації.

«BACKEND»

Протягом життєвого циклу інформаційної системи у «backend» зберігаються та обробляються дані, створенні та редаговані у всій системі. Компоненти цього компонента включають проектну базу даних та API (Application Programming Interface), які дозволяють отримувати доступ до основних функцій системи. Незамінна частина «backend» створена на мові програмування C# і Microsoft SQL Server у якості бази даних.

Онлайн-платформа має пропрієтарну ліцензію програмного забезпечення. Через це тільки правовласник має доступ до вихідного коду програми.

Програмна частина платформи зберігається у приватному хмарному репозиторії веб-сервісу GitHub (через дотримання правил комерційної ліцензії) за посиланням: GitHub. Доступ до готової користувальницької частини онлайн-платформи знаходиться за адресою: посилання.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ FRONTEND

Відповідно до моделі «фази розробки», викладеної нижче, розробка системи була непростою. З часом ми покращили безпеку та функціональність, тому що це був перший масштабний онлайн-додаток, який ми розробили. Кожен великий етап розробки цього додатка був протестований.

1. Вимоги до збірки
2. Вибір інструменту розробки
3. Інтеграція сторінок користувача
4. Інтеграція функціональності входу

Вимоги до збору

Обов'язково потрібно пам'ятати, що деякі аспекти веб-сторінки можуть змінюватися протягом всієї розробки інтерфейсу. Ідея полягала в тому, щоб зібрати всі вимоги до початку розробки інтерфейсу. Незважаючи на те, що наш головний відділ надав більшість вимог, ми обговорювали, чи є вони розумними, здійсненими і чи вписуються у часові рамки цієї розробки.

Перелік вимог розробки:

- Сторінки користувача повинні бути оптимізовані для використання комп'ютерами, планшетами і смартфонами;
- Користувацькі сторінки повинні бути у першу чергу розроблені у вигляді, створенному дизайнером.
- Сторінки користувача повинні бути пов'язані з базою даних, створеною розробником “backend”, щоб використовувати всю необхідну інформацію.
- Найважливішим аспектом є те, щоб система дотримувалася загальних характеристик.

Вибір інструментів розробки

Створення веб-сайту з правильними інструментами має значний вплив на кількість часу та зусиль, необхідних для його створення. Це пов'язано зі складністю сайту, і необхідним рівнем налагодження. Визначення інструментів розробників має важливе значення для майбутнього розвитку. Використання широко відомих технологій може допомогти розробникам швидше вивчити нову систему.

Вибір зовнішніх інструментів розробки, що беруть участь з урахуванням цих факторів:

- Розмір проекту
- Вид проекту
- Впізнаваність інструментів

Фреймворк bootstrap використовувався для задання стилів користувальницького інтерфейсу. Bootstrap — це інтерфейсний фреймворк з відкритим вихідним кодом, вихідний код якого доступний за ліцензією MIT. Він включає кнопки в стилі CSS, таблиці, форми та інші елементи, а також деякі розширення JavaScript. Bootstrap є найпопулярнішим фронтенд-фреймворком з великою документацією, що робить його придатним для використання у розробці системи.

Надаючи адаптивні функції та попередньо визначені стилі, bootstrap спрощує та прискорює розробку інтерфейсу. Завжди можна змінити CSS класи або додати свої власні до загального пакету.

Реалізація сторінок користувача

Спочатку ми створили головну веб-сторінку, яка є найважливішим фактором, оскільки це перше, що бачать користувачі, коли відвідують нашу платформу.

Інтерфейс користувача та процес бронювання були розроблені, щоб бути максимально безперебійними та простими. Звівши кількість кліків миші та /або дотиків до мінімуму, це стало можливим.

Варіанти наступні:

- Якщо користувачі хочуть шукати певні місця, вони можуть зробити це в полі пошуку.
- Якщо клієнти все ще не впевнені в тому, куди вони хочуть піти, ми рекомендуємо багато популярних напрямків та можливість перегляду за типом нерухомості в країні, з якої вони в даний час переглядають (геолокація є частиною майбутньої розробки)
- Якщо користувачі хочуть перейти в різні регіони, міста або країни, вони можуть зробити це, натиснувши на посилання.

Архітектурна основа складається з односторінкового додатка, який інкапсулює у собі дані у конкретних компонентах і посилається на основні односторінкові програми. У нашому проекті використовувався стандартний модуль маршрутизатора Angular, який дозволив нам створювати окремі маршрутизаційні зв'язки для кожного з компонентів. Це може бути використано для переходу між різними компонентами.

Після вирішення проблем маршрутизації та проблем побудови веб-сторінки, була реалізована база даних. Це дозволить завантажувати та зберігати інформацію, надану користувачами. Для цього використовувалися мережеві запити до API. Виклик API відбувається, коли клієнтський застосунок робить мережевий запит до нього. API бере запитувані дані та віддає їх клієнту.

Нижче наведено зразок зображення, яке показує метод отримання даних "міст" з бази даних, а потім використання його з метою використання інформації.

```
getCitiesData(): void {
  fetch(
    'https://localhost:44381/api/stayspage/getcitiesdata?country=' + this.mainDataService.getCurrentCountry(),
    {
      method: 'GET',
    }
  )
    .then((r) => r.json())
    .then((r) => {
      if (r.code === 200) {
        this.cities = r.cities;
        this.citySuggestionsLength = r.citySuggestionsLength;
        this.footerCities = r.footerCities;
        this.mainDataService.setSearchingCities(this.cities);
      } else {
        this.showAlert('Data fetching error!');
      }
    })
    .catch((err) => {
      this.mainDataService.alertContent = err;
      this.modalService.open(this.alert);
    });
}
```

Як засіб відображення різних елементів на кожній із веб-сторінок, ми адаптували шаблон дизайну сторінок. Це було реалізовано за допомогою bootstrap і численних користувацьких класів стилей CSS. Як ви можете бачити з наведеного нижче прикладу, існує ряд попередньо визначених класів Bootstrap (рядок m-0 p-0) і деякі власні класи (input-with-icon), які призначені для задоволення потреб макета дизайнера.

```
<div class="row m-0 p-0">
  <div class="col-md-10 p-0">
    <div class="row m-0 p-0">
      <div class="col-md-4 px-6">
        <div class="input-with-icon">
          <img src="assets/images/9.svg" alt="" />
          <input type="text" class="header-input pl-50" placeholder="Where are you going?"
            name="searchViewModel.place" [(ngModel)]="searchViewModel.place" />
          <div class="invalid_feedback" *ngIf="
              searchViewModel.place.replaceAll(reg, '').length == 0
            ">
            Enter destination to search
          </div>
        </div>
      </div>
```

Додавання функцій входу

З метою ідентифікації електронної пошти ми надаємо серверу адресу електронної пошти користувача, пароль і код підтвердження, який запитує система. Паролі перевіряються двічі, щоб запобігти помилкам введення тексту. Ми передаємо встановлену інформацію до бази даних в її поточному стані, коли форма подається користувачем. Система перевіряє, чи були виконані вимоги до поля введення. Адреса електронної пошти, яка використовується як ім'я користувача, унікальна для кожного користувача, оскільки кожен з них має свою адресу електронної пошти. На додаток до URL-адреси сайту, додайте /auth до кінця, щоб перейти до сторінки входу. Користувачі будуть зобов'язані вказати свої адреси електронної пошти. Після введення імені користувача форма передається до функції перевірки

користувача. Це порівнює дані із даними у базі даних і надсилає електронний лист, що містить код підтвердження, який потім використовується для входу.

РОЗДІЛ 3
ОЦІНКА

Потреби веб-сайту повинні бути розглянуті після ретельного проектування архітектури програми та розробки її основних компонентів. Оцінка потреб користувачів може бути використана як основа для створення цілей на майбутнє.

У розділах «ЗАГАЛЬНІ ХАРАКТЕРИСТИКИ СИСТЕМИ» і «РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ» ми узагальнили початкові критерії розробки веб-додатка. У цих розділах ми розглядаємо вимоги як в контексті архітектури додатків, так і в контексті того, як вона реалізується. Адже продумане планування не обов'язково гарантує успіх.

Основи

Спочатку платформа booking.com була взята у приклад для створення платформи. Зокрема, дизайн і функціональність повинні бути схожі на booking.com, з кількома поліпшеннями макета і виправленнями. Беручи це до уваги в контексті архітектури додатків, є багато рішень, які слід розглянути.

Створення численних класів стилей, що реалізують бажану функціональність, може виконати цю вимогу.

Функціональність односторінкового додатку є ідеальним рішенням для нашого випадку.

Однак, коли програма розроблена з вибраним підходом, легше регулювати деталізацію доступу, оскільки користувачам може бути обмежено доступ до певного компонента.

Таким чином, веб-додаток може бути адаптований до кращої відповідності інформаційним вимогам конкретного користувача та потребам бізнесу. Фактично, архітектура на основі компонентів дозволяє розробляти кожен компонент самостійно. Тому наш процес розробки програмного забезпечення значно полегшується таким підходом.

Вимога може вважатися повністю виконаною в поточному контексті, принаймні на даний момент, оскільки вона була виконана в процесі фактичного впровадження. В результаті обраної архітектури один додаток можна розробити значно швидше, так як розміром проекту можна легше маніпулювати.

Масштабованість

Вимога до розширення веб-додатків йде рука об руку з першими критеріями, оскільки інформація з “Основи” легко дозволяє масштабувати існуючий додаток. Залежно від того, як будується фундамент, простота масштабування може істотно відрізнятися. Додавши нові компоненти в додаток, програма може стати розширеною і викликати проблеми.

У багатьох випадках ці проблеми виникають через проблеми залежності між класами або тому, що в класі все ще є помилки, які не були виявлені у процесі розробки. Цілком можливо, що ці помилки можуть призвести до краху всієї

системи, якщо вони з'являться у виробництві. Добре продумана архітектура спростить масштабування і уникне проблем, які можуть виникнути.

Компоненти веб-додатків є самозалежними, тому додавання нових не повинно впливати на будь-які існуючі компоненти. Це залежить від того, як веб-додаток керує взаємодією між компонентами. Сильна взаємодія між компонентами може викликати проблеми. Окремі компоненти веб-програми також можуть бути легко збережені завдяки створенній архітектурі. Уражена частина веб-програми може бути виправлена тільки в тому випадку, якщо помилки виявлені у виробничому середовищі.

Коли вимога розглядається в контексті реалізації, її можна вважати виконаною. Досить просто додати нові компоненти до веб-програми, використовуючи функціональні можливості, надані основним фреймворком. Обсяг реалізації, що вимагається основним фреймворком, аж ніяк не великий, але більшість робіт, необхідних для того, щоб дозволити додатку працювати як компонент веб-додатка, виконується безпосередньо до самого компонента.

Кожен компонент вимагає певного налаштування, а точка входу програми, наприклад, повинна бути визначена заздалегідь. Управління всіма основними конфігураціями може створити проблеми в майбутньому, якщо кількість компонентів значно зросте.

Узгоджений інтерфейс користувача та досвід

Вимога узгодженості інтерфейсу користувача і користувацького досвіду у веб-додатку була розбита на кілька розділів. Розглянуті компоненти стосуються таких проблем, як єдність окремих елементів інтерфейсу користувача та більш широкого дизайну інтерфейсу користувача.

Перший компонент критеріїв показує, що повинен бути обмежений набір загальних стилів макета, що всі види по всьому веб-додатку повинні

дотримуватися зазначених правил. Загальний дизайн доступних на даний момент додатків дотримується однакових правил.

Елементи інтерфейсу користувача, що використовуються для взаємодії з користувачем, повинні бути уніфіковані для кожного випадку використання відповідно до другої частини вимоги. Це означає, що на кожній сторінці всі текстові поля і кнопка, що використовуються для додавання нового елемента, повинні виглядати однаково. Розроблений дизайн просто допомагає розробникам правильно впроваджувати елементи інтерфейсу користувача, але вимагає відданості дизайнера в команді.

У третьому розділі критеріїв зазначено, що позиціонування елементів інтерфейсу користувача має бути узгодженим у всіх перспективах. Наприклад, якщо елемент рядка пошуку присутній поверх таблиці даних, він повинен бути присутнім у всіх містах. Коли розташування елементів довільно змінюються в різних містах, користувачеві стає складніше ефективно керувати додатком. Знову ж таки, система дизайну може допомогти розробникам правильно реалізувати користувальницький інтерфейс, але їх створення вимагає зобов'язань дизайнера і знання.

Четвертий розділ вимоги стосується питання масштабування елементів інтерфейсу користувача. Знову ж таки, розроблена система дизайну допомагає розробникам, пропонуючи класи корисних стилів, які можуть бути використані для опису розміру елемента. Цю умову можна вважати частково виконаною, з урахуванням поточних компонентів, а також самого веб-додатка. Ця проблема повинна бути вирішена в майбутньому, як і розширення нинішньої системи проектування з новими загальними компонентами для кращого задоволення цілей онлайн-програми.

Можливі майбутні вдосконалення веб-застосунку

Незважаючи на те, що кілька проблем були виявлені і згодом вирішені під час роботи над цим продуктом, є ще кілька, які повинні бути вирішені в майбутньому.

Їх проблеми з іншими альтернативними компонентами будуть обмежені самим компонентом. Оскільки наявні компоненти підтримують кешування та повторне отримання даних, інші можливі компоненти повинні реалізувати можливість забезпечити кращий та більш чуйний користувальницький досвід. Нові компоненти також повинні бути розділені на менші частини, щоб зробити їх більш зрозумілими та спростити подальшу розробку.

Однією з проблем, яка може виникнути в майбутньому, є розмір компонентів, які складають веб-додаток. Якщо розміри компонентів стають занадто великими, перехід від однієї частини програми до іншої може зайняти багато часу. Якщо один клієнт має велику кількість відкритих вікон і занадто багато з них завантажуються або підтримуються в пам'яті одночасно, це може викликати проблеми. Для вирішення цих проблем необхідно розробити механізм використання простору веб-застосунку.

До технічних проблем, які залишаються у веб-додатку, як зазначалося раніше, ще належить їх виправити. Кількість спільних компонентів в системі проектування в даний час досить мінімальна. Цього слід дотримуватися, особливо з огляду на необхідність послідовного інтерфейсу користувача та досвіду по всій онлайн-програмі.

ВИСНОВКИ

Веб-додаток, побудований і реалізований під час написання цього документу, все ще знаходиться на ранній стадії, з багатьма компонентами, які можна поліпшити і додати. Робота, виконана для веб-додатку, була повністю пояснена у документі, і, як зазначається в розділі «ОЦІНКА», було зроблено багато роботи, але в майбутньому доведеться зробити ще більше. Сподіваємося, що недоліки онлайн-програми та відсутні елементи будуть вирішені в майбутньому.

З огляду на початкову точку процесу проектування архітектури і висновки, наведені в розділі «ОЦІНКА», можна з впевненістю стверджувати, що починати процес проектування після значного обсягу реалізації не бажано. Найскладніші моменти розробки, що виникли, були пов'язані з тим, що був значний обсяг відсутніх робіт для проектної системи. Наприклад, відсутність системи проектування і процес включення її в існуюче застосування потребували найбільших зусиль.

Можна навіть стверджувати, що це вимагало більших зусиль, ніж решта реалізації. В результаті ігнорування дизайну можна мати серйозні наслідки в майбутньому. Це питання слід вирішувати і краще враховувати в майбутніх подіях.

Дуже важливо розробити систему дизайну як частину загальної передньої архітектури, оскільки вона приносить користь як кінцевому користувачеві, так і розробнику. Це пов'язано з тим, що система проектування допомагає розробнику створити більш узгоджений користувальницький досвід з меншою роботою, використовуючи спільні компоненти системи проектування та інші утиліти. Оскільки елементи інтерфейсу користувача

відомі по всій системі, узгодженість інтерфейсу користувача дозволяє кінцевому користувачеві більш ефективно керувати системою.
Регулювання кількості деталізації веб-програми, використовуючи фреймворк bootstrap як основу для додавання стилів до сторінок веб-додатків, дозволило повністю використовувати існуючі компоненти, які раніше були реалізовані до початку архітектурного проектування.

На даний момент немає вагомих причин критикувати архітектуру веб-додатку. Однією з переваг використання фреймворку bootstrap в інтерфейсі, як зазначалося раніше, є те, що додатки технологічно незалежні. Це дозволяє легко оновлювати та змінювати технології, що використовуються в процесі розробки додатків. В результаті можлива найкраща технологія для кожної даної програми.

Під час підготовки цього документу, яка почалася в січні 2022 року, було багато злетів і падінь, оскільки було кілька разів, коли робота, пов'язана з продуктом, була поставлена на паузу. Це часто було викликано умовами в нашому житті і в нашій країні. Оскільки технології, що використовуються для реалізації, були настільки простими у використанні, архітектура веб-програми була спланована і побудована досить швидко після початку роботи.

Однак очікування системи проектування зайняло набагато більше часу, ніж передбачалося, і використовувати її було не простіше. Проблеми з системою дизайну та існуючими додатками зробили написання фактичної тези трохи нудним, і було важко знайти мотивацію, щоб продовжувати йти, але, здається, вона завершена зараз.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ КЛІЄНТСЬКА ЧАСТИНИ

№ п.п.	**Ресурс**
1.	Angular; Розробник: Google; Документація: посилання.
2.	Visual Studio Code; Розробник: Microsoft; Документація: посилання.
3.	JavaScript; Розробник: Brendan Eich of Netscape; Документація: посилання.
4.	TypeScript; Розробник: Microsoft; Документація: посилання.
5.	Bootstrap; Розробник: Bootstrap Core Team; Документація: посилання.

6.	HTML; Розробник: WHATWG; Документація: посилання.
7.	CSS; Розробник: World Wide Web Consortium; Документація: посилання.
8.	FETCH API; Документація: посилання.