Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Калмикової Анастасії Володимирівни*
(ПІБ)

академічної групи *121-18ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для інформаційної підтримки надання послуг студії організації свят*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Гуліна І.Г.*			
розділів:				
спеціальний	*доц. Гуліна І.Г.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентки *121-18ск-1* *Калмикової Анастасії Володимирівни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інформаційної системи підтримки надання послуг студії організацій свят*

затверджена наказом ректора НТУ «ДП» від « 07 » 06 . 2021 р. № 317-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Калмикова А.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка 75 с., 21 рис., 3 табл., 3 дод., 21 джерело.

Об'єкт розробки: програмне забезпечення веб-орієнтованої інформаційної системи організації надання послуг організації свят.

Мета кваліфікаційної роботи: підвищення зручності та ефективності надання послуг організації свят.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування застосунку, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленого програмного продукту, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає у створенні застосунку, що підвищує зручность та ефективність організації бізнесу надання послуг організації свят.

Актуальність програмного продукту визначається великим попитом на подібні інструменти.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ДОДАТОК, ОРГАНІЗАЦІЯ СВЯТ.

ABSTRACT

Explanatory note 75 pp., 21 figs., 3 tabls., 3 apps, 21 sources.

Object of development: software of the web-oriented information system of the organization of rendering of services of the organization of holidays.

The purpose of the qualification work: to increase the convenience and efficiency of providing services for the organization of holidays.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the application, defines the input and output data, describes the characteristics of the parameters of hardware, describes the call and download the application, describes the program .

In the economic section, the complexity of the developed software product is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The practical significance lies in the creation of an application that increases the convenience and efficiency of the business organization of providing holiday services.

The relevance of the software product is determined by the high demand for such tools.

Keywords: INFORMATION SYSTEM, APPENDIX, HOLIDAY ORGANIZATION.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ІС – інформаційна система;

ООП – об'єктно-орієнтоване програмування;

ОС – операційна система;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології;

IDE – Integrated Development Environment (Інтегроване середовище розробки);

XAML – Extensible Application Markup Language (Розширювана мова розмітки для додатків);

VS – Visual Studio;

SQL – Structured Query Language (Мова програмування структурованих запитів, яка використовується в якості ефективного способу збереження даних).

ВСТУП

На сучасному етапі розвитку людства Інтернет виступає в ролі потужного інструменту з пошуку та надання інформації. За статистикою, більше половини жителів планети мають доступ до мережі Інтернет. Як наслідок, розробка Web-сайту в мережі Інтернет дозволяє використовувати сучасні технології для розвитку інформаційної підтримки та реклами.

Процес розробки Web-сайту та інформаційних систем в мережі Інтернет забезпечить приплив нових користувачів, оскільки добре зроблений сайт буде достатньо просто знайти за допомогою пошукових систем. За допомогою сайту можна домогтися швидкого реагування на побажання користувачів, а також вносити відповідні зміни в роботу Web-сайту.

Тому метою даного проекту стала розробка веб-орієнтованої інформаційної системи для залучення нових клієнтів та автоматизації процесів. Основне призначення спроектованої системи – залучити нових клієнтів та автоматизувати деякі процеси, а саме поповнення бази клієнтів, надання інформації про наявні послуги, ведення обліку заказів.

Завдання кваліфікаційної роботи та об'єкт її діяльності безпосередньо пов'язані зі спеціальностю 121 «Інженерія програмного забезпечення» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених компетенцій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Предметною галузю даної кваліфікаційної роботи є проектування веб-орієнтованої інформаційної системи для залучення нових клієнтів та автоматизація роботи студії організацій свят. Сайт повинен забезпечувати користувальницький інтерфейс для роботи з базою даних. Студія надає послуги клієнтам, що звертаються з метою організувати певне свято, після уточнення всіх деталей замовлення затверджується і оформлюється договір. Сумарна вартість замовлення підраховується згідно прайс-листу.

Користуватися автоматизованою програмною системою будуть співробітники студії для доповнення бази даних та сформування документації замовлень і клієнти для замовлення послуг та отримання різноманітної інформації про компанію.

Система студії організацій свят крім інформації про продажі включає в себе докладні відомості про послуги, їх характеристики, цінову політику, а також клієнтів і їх замовлень.

Вхідними даними будуть заповнення таблиць бази даних.

Вихідними даними будуть заповненні таблиці та запити до бази даних.

Таким чином слід вивчити та проаналізувати питання розробки веб-орієнтованої інформаційної системи та баз даних.

У теорії та практиці створення інформаційних систем виділяють три підходи: локальний, глобальний та системний.

Суть локального підходу полягає в тому, що інформаційні системи створюють послідовним нарощуванням задач, які розв’язуються на ЕОМ. Він передбачає необмежений розвиток інформаційних систем, а тому кожну із них неможливо пізнати в цілому. Крім того, проект на предмет його повноти взагалі не розглядається та втрачається можливість науково обґрунтувати вибір і

оцінити напрямки розвитку інформаційної системи, комплекс технічних засобів, а також побудувати її модель. До позитивних сторін цього підходу віднесемо: відносно швидку віддачу, наочність задач, можливість розробки невеликими “замкненими” групами, простоту керування створенням систем. Недоліки: неможливість забезпечення раціональної організації комплексів задач, дублювання, постійна перебудова програм та організації задач, що призводить до дискредитації самої ідеї створення інформаційної системи.

При глобальному підході спочатку розробляють проект немовби повної, завершеної системи, а потім її впроваджують. Як правило, цей підхід призводить до морального старіння проекту ще до його впровадження, оскільки час його розробки може перевищувати період оновлення технічних, програмних та інших засобів, використаних у ньому.

Системний підхід до створення інформаційної системи – це комплексне вивчення економічного об’єкта як одного цілого з представленням частин його як цілеспрямованих систем і вивчення цих систем та взаємовідносин між ними. При системному підході економічний об’єкт розглядається як сукупність взаємопов`язаних елементів однієї складної динамічної системи, яка перебуває в стані постійних змін під впливом багатьох внутрішніх і зовнішніх факторів, пов’язаних процесами перетворення вхідного набору ресурсів в інші вихідні ресурси.

Системний підхід має такі принципи:

– кінцевої мети – абсолютний пріоритет кінцевої (глобальної) мети;

– єдності –розгляд системи як цілого, так і сукупності частин (елементів);

– зв'язності – розгляд будь якої частини разом з її зв'язками з оточенням;

– модульної побудови – корисно виділяти модулі в системі та розглядати її як сукупність модулів;

– ієрархії - корисно вводити ієрархію частин (елементів) і (чи) їх ранжування;

– функціональності – спільний розгляд структури і функцій з пріоритетом функцій над структурою;

– розвитку – врахування змін системи, її здатність до розвитку, розширення, заміни частин, нагромадження інформації ;

– децентралізації - поєднання рішень, які приймаються, та керування централізацією і децентралізацією;

– невизначеність – врахування невизначеностей та випадковостей у системі.

Задачею системного підходу до створення інформаційної системи є розробка всієї сукупності методологічних і соціально – наукових засобів обстеження (опис, аналіз, синтез, реалізація) систем різного типу.

Створення бази даних слід починати з її проектування (розробки). У результаті проектування має бути визначена структура бази, тобто склад таблиць, їхня структура та логічні зв'язки. Структура реляційної таблиці визначається складом стовпців, їхньою послідовністю, типом даних кожного стовпця та їхнім розміром, а також ключем таблиці. Процес проектування можна здійснювати двома підходами. За першого підходу спочатку визначають основні задачі, для розв'язання яких створюється база, та потреби цих задач у даних. За другого підходу визначають предметну область (сферу), здійснюють аналіз її даних і встановлюють типові об'єкти предметної області. Найбільш раціональним підходом проектування бази даних є поєднання обох підходів.

Зазвичай з базами даних працюють дві категорії користувачів. Перша категорія - проектувальники. Процес проектування бази даних поділяється на етапи, кожний з яких передбачає виконання певних дій. Перший етап-розробка інформаційно-логічної моделі даних предметної області, який базується на описі предметної області, отриманому в результаті її обстеження. На цьому етапі спочатку визначають склад і структуру даних предметної області, які мають міститись у базі даних та забезпечувати виконання запитів, задач і застосувань користувача. Ці дані мають форму реквізитів, що містяться в різних документах - джерелах завантаження бази даних. Аналіз виявлених

даних дозволить визначити функціональні залежності реквізитів, які використовують для виділення інформаційних об'єктів, що відповідають вимогам нормалізації даних. Подальше визначення структурних зв'язків між об'єктами дозволяє побудувати інформаційно-логічну модель.

Другий етап - визначення логічної структури бази даних. Для реляційної бази даних цей етап є значною мірою формальним, оскільки інформаційно-логічна модель відображається в структуру реляційної бази даних адекватно.

Наступний етап - конструювання таблиць бази даних, який здійснюється засобами СУБД, та узгодження їх із замовником. Структура таблиць бази даних задається за допомогою засобів опису (конструювання) таблиць у СУБД із цілковитою відповідністю інформаційним об'єктам. Крім таблиць, проектувальники розробляють й інші об'єкти бази даних, які призначені, з одного боку, для автоматизації работи з базою, а з іншого - для обмеження функціональних можливостей работи з базою (безпека бази даних). Після формування структури таблиць база даних може наповнюватись даними з документів-джерел. Проектувальники, як правило, не наповнюють базу конкретними даними (оскільки замовник може вважати дані конфіденційними і не надавати стороннім особам). Винятком є експериментальне наповнення модельними даними на етапі відлагодження об'єктів бази.

Друга категорія - користувачі, які працюють із базами даних. Вони отримують вхідну базу даних від проектувальників, наповнюють її та обслуговують. Користувачі не мають засобів доступу до управління структурою бази, вони мають доступ тільки до даних, при цьому тільки до тих, робота з якими передбачена на їхньому конкретному робочому місці.

Проектування бази даних починається з вивчення технічного завдання на проектування бази даних, яке повинен надати замовник. Отже, бажано, щоб замовник володів відповідною термінологією і знав, принаймні в загальних рисах, технічні можливості основних СУБД. На жаль, на практиці ці побажання виконуються не завжди. Тому зазвичай розробники використовують такі підходи: демонструють замовникові роботу аналогічної бази даних, після чого

узгоджують специфікацію відмінностей; якщо аналога немає, з'ясовують коло задач і вимог замовника, після чого допомагають йому підготувати технічне завдання. Під час підготовки технічного завдання складають: перелік вхідних даних, з якими працює замовник; перелік вихідних даних, потрібних замовникові для управління структурою свого підприємства; перелік вихідних даних, які не є необхідними для замовника, але які він повинен надати іншим організаціям (у вищестоящі структури, в органи статистики, інші адміністративні і контрольні організації).

Визначивши основну частину даних, які замовник використовує, розпочинають розробку структури бази, тобто структури її основних таблиць.

1. Робота починається з визначення генерального переліку полів, який може нараховувати десятки і сотні позицій.

2. Відповідно до типу даних, що розміщуються в кожному полі, визначають тип кожного поля.

3. Розподіляють поля генерального списку по базових таблицях. На першому етапі розподіл здійснюють за функціональною ознакою. Мета - забезпечити одноразове введення даних в одну таблицю по можливості в рамках одного підрозділу, або (ще краще) - на одному робочому місці. На другому етапі розподілу полів здійснюють нормалізацію даних з метою вилучення повторів даних у таблицях бази даних.

4. Для кожної таблиці визначають ключове поле. Ключовим вибирають поле, дані в якому повторюватись не можуть. Наприклад, для таблиці даних про студентів таким полем може бути індивідуальний шифр студента. Для таблиць, у яких міститься розклад занять, такого поля можна і не знайти, але його можна створити штучно комбінуванням полів "Час заняття" і "Номер аудиторії". Ця комбінація унікальна, оскільки в певній аудиторії в певний час назагал не проводять двох різних занять. Якщо ж у таблиці взагалі немає полів, які можна було 6 використовувати як ключові, завжди можна ввести додаткове поле типу лічильник - воно за визначенням не може містити дані, що повторюються.

5. На наступному етапі визначають зв'язки між таблицями (схему даних). Зв'язки між таблицями організуються на основі спільного поля, причому в одній із таблиць воно обов'язково має бути ключовим. Тобто на стороні "один" має бути ключове поле, яке не повторюється, значення на стороні "багато" можуть повторюватися.

6. "Паперовий" етап роботи над технічними пропозиціями закінчується розробкою схеми даних. Цю схему слід узгодити із замовником, після чого розпочати безпосереднє створення бази даних. Слід пам'ятати, що в ході розробки проекту замовникові неодмінно будуть надходити нові ідеї. Можливість гнучкого використання його побажань суттєво залежить від кваліфікації розробника бази даних. Якщо схема даних складена правильно, підключити до бази нові таблиці неважко. Якщо структура бази нераціональна, розробник може наштовхнутись на суттєві труднощі і дійти суперечності із замовником. Суперечка виконавця із замовником завжди свідчить про недостатню кваліфікацію виконавця. На цьому етапі завершується попереднє проектування бази даних, і на наступному етапі починається її безпосередня розробка (впровадження).

1.2. Призначення розробки та галузь застосування

Метою кваліфікаційної роботи є розробка веб-орієнтованого додатку для залучення нових клієнтів та автоматизації процесів ведення бізнесу. Основне призначення спроектованої системи – залучити нових клієнтів та автоматизувати деякі процеси, а саме поповнення бази клієнтів, надання інформації про наявні послуги, ведення обліку заказів.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Отже, підставами для розробки та виконання кваліфікаційної роботи є:

- освітня програма спеціальності 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;
- завдання на кваліфікаційну роботу на тему: Розробка програмного забезпечення для інформаційної підтримки надання послуг студії організації свят.

1.4. Постановка завдання

При розробці даної інформаційної системи необхідно реалізувати наступні завдання:

Функціональні:

– Формування даних про клієнтів та послуги.
– Формування даних прайс-листу для визначення вартості послуг.
– Формування довідки про наявні послуги.
– Формування довідки про отримані замовлення.
– Пошук послуг за вказаними характеристиками.
– Здійснювати підрахунок сумарної вартості обраних послуг.
– Здійснювати підрахунок прибутку студії за вказаний період часу.
– Формування довідки про замовлення певного клієнта або всіх клієнтів за вказаний період часу.
– Мати можливість перегляду та редагування даних студії.
– Мати можливість зміни налаштувань з’єднання з сервером.
– Мати можливість збереження налаштувань для з’єднання з сервером у файл.
– Введення дат, числових та інших даних реалізувати по можливості у вигляді випадаючих візуальних елементів (календарів, калькуляторів, списків тощо).
– При запуску програми автоматично завантажити останні вдалі налаштування та виконати з’єднання з сервером.

Нефункціональні системні завдання:

– Реалізувати графічний інтерфейс.
– Веб-сервер, наприклад Open Server.
– ОС Windows XP,7,8,10.
– СУБД MongoDB.
– Джерелом введення інформації є клавіатура.
– Результати роботи виводяться на екран та заносяться у базу даних.
– Обов’язкове збереження даних перед закінченням роботи.
– Надійність - в програмі відсутні помилки. Програма організована так, що помилкові ситуації, що виникають, не ведуть до фатальних наслідків.
– Простота використання - чітко визначені умови і межі використання даної програми, ясно оговорені пояснення, які полегшують і прискорюють використання, але виключають невірне застосування.
– Зручність експлуатації - передбачена можливість для подальшого покращення і модифікації, без великих переробок для того, щоб експлуатація програми здійснювалась з мінімальними затратами.
– Мобільність - програма пристосована до переносу з одного комп’ютера на інший, або до змін операційного середовища.
– Сумісність.
– Програма відповідає існуючим стандартам по кодам, форматам, структурам даних.
– Чітко визначені вхідні та вихідні дані з вказанням їх функціонального призначення.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Вимоги до програмного забезпечення — набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде розроблено, або знаходиться у розробці. Вимоги визначаються в процесі аналізу вимог та фіксуються в специфікації вимог, діаграмах прецедентів та інших артефактах процесу аналізу та розробки вимог.

При розробці даної програмної системи було виділено наступні вимоги:

Функціональні:

– Формування даних про клієнтів та послуги.
– Формування даних прайс-листу для визначення вартості послуг.
– Формування довідки про наявні послуги.
– Формування довідки про отримані замовлення.
– Пошук послуг за вказаними характеристиками.
– Здійснювати підрахунок сумарної вартості обраних послуг.
– Здійснювати підрахунок прибутку студії за вказаний період часу.
– Формування довідки про замовлення певного клієнта або всіх клієнтів за вказаний період часу.
– Мати можливість перегляду та редагування даних студії.
– Мати можливість зміни налаштувань з'єднання з сервером.
– Мати можливість збереження налаштувань для з'єднання з сервером у файл.
– Введення дат, числових та інших даних реалізувати по можливості у вигляді випадаючих візуальних елементів (календарів, калькуляторів, списків тощо).
– При запуску програми автоматично завантажити останні вдалі налаштування та виконати з'єднання з сервером.

Нефункціональні системні вимоги:

– Реалізувати графічний інтерфейс.
– Веб-сервер, наприклад Open Server.
– ОС Windows XP,7,8,10.
– СУБД MongoDB.
– Джерелом введення інформації є клавіатура.
– Результати роботи виводяться на екран та заносяться у базу даних.
– Обов'язкове збереження даних перед закінченням роботи.
– Надійність - в програмі відсутні помилки. Програма організована так, що помилкові ситуації, що виникають, не ведуть до фатальних наслідків.
– Простота використання - чітко визначені умови і межі використання даної програми, ясно оговорені пояснення, які полегшують і прискорюють використання, але виключають невірне застосування.
– Зручність експлуатації - передбачена можливість для подальшого покращення і модифікації, без великих переробок для того, щоб експлуатація програми здійснювалась з мінімальними затратами.

– Мобільність - програма пристосована до переносу з одного комп’ютера на інший, або до змін операційного середовища.

– Сумісність.

– Програма відповідає існуючим стандартам по кодам, форматам, структурам даних.

– Чітко визначені вхідні та вихідні дані з вказанням їх функціонального призначення.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

– контроль вхідних даних;

– обробку виняткових ситуацій;

– виведення повідомлень про помилки;

– можливість повторного введення даних;

– можливість безперервної роботи протягом не менше 120 годин (5 діб);

– забезпечення збереження та неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови застосунку.

Особливих чи додаткових вимог до інформаційної безпеки додатку не висувається.

1.5.3 Вимоги до складу та параметрів технічних засобів

Програма не є вимогливою до складу та параметрів технічних засобів та може завантажуватись ПК чи ноутбуках під управлінням ОС Windows.

Системні вимоги:

– Графічний інтерфейс (наявність монітора).

– Веб-сервер, наприклад Open Server.

– ОС Windows XP,7,8,10.

– СУБД MongoDB.

– Джерелом введення інформації є клавіатура.

– Результати роботи виводяться на екран та заносяться у базу даних. (достатне місце для збереження інформації).

– Обов’язкове збереження даних перед закінченням роботи (достатне місце для збереження інформації).

1.5.4 Вимоги до інформаційної та програмної сумісності

Програма не є вимогливою до інформаційної та програмної сумісності.

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати веб-орієнтована система, відповідало наступним вимогам:

- ✓ операційна система сімейства Windows (XP, Vista, 7, 8, 10), Linux;
- ✓ веб-браузер Firefox / Google Chrome / Opera / Internet Explorer.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою кваліфікаційної роботи є розробка веб-орієнтованого додатку для залучення нових клієнтів та автоматизації процесів ведення бізнесу. Основне призначення спроектованої системи – залучити нових клієнтів та автоматизувати деякі процеси, а саме поповнення бази клієнтів, надання інформації про наявні послуги, ведення обліку заказів.

Розроблена в даній роботі інформаційна система забезпечує виконання наступних функцій та властивостей:

- Формування даних про клієнтів та послуги.
- Формування даних прайс-листу для визначення вартості послуг.
- Формування довідки про наявні послуги.
- Формування довідки про отримані замовлення.
- Пошук послуг за вказаними характеристиками.
- Здійснювати підрахунок сумарної вартості обраних послуг.
- Здійснювати підрахунок прибутку студії за вказаний період часу.
- Формування довідки про замовлення певного клієнта або всіх клієнтів за вказаний період часу.
- Мати можливість перегляду та редагування даних студії.
- Мати можливість зміни налаштувань з’єднання з сервером.
- Мати можливість збереження налаштувань для з’єднання з сервером у файл.
- Введення дат, числових та інших даних реалізувати по можливості у вигляді випадаючих візуальних елементів (календарів, калькуляторів, списків тощо).
- При запуску програми автоматично завантажити останні вдалі налаштування та виконати з’єднання з сервером.
- Реалізуваний графічний інтерфейс.
- Веб-сервер, наприклад Open Server.
- ОС Windows XP,7,8,10.

– СУБД MongoDB.
– Джерелом введення інформації є клавіатура.
– Результати роботи виводяться на екран та заносяться у базу даних.
– Обов’язкове збереження даних перед закінченням роботи.
– Надійність - в програмі відсутні помилки. Програма організована так, що помилкові ситуації, що виникають, не ведуть до фатальних наслідків.
– Простота використання - чітко визначені умови і межі використання даної програми, ясно оговорені пояснення, які полегшують і прискорюють використання, але виключають невірне застосування.
– Зручність експлуатації - передбачена можливість для подальшого покращення і модифікації, без великих переробок для того, щоб експлуатація програми здійснювалась з мінімальними затратами.
– Мобільність - програма пристосована до переносу з одного комп’ютера на інший, або до змін операційного середовища.
– Сумісність.
– Програма відповідає існуючим стандартам по кодам, форматам, структурам даних.
– Чітко визначені вхідні та вихідні дані з вказанням їх функціонального призначення.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці системи математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

При розробці системи було використано шаблон проектування MVC. Моде́ль–вигляд–контро́лер (MVC, Модель–представлення–контролер, англ. Model-view-controller) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

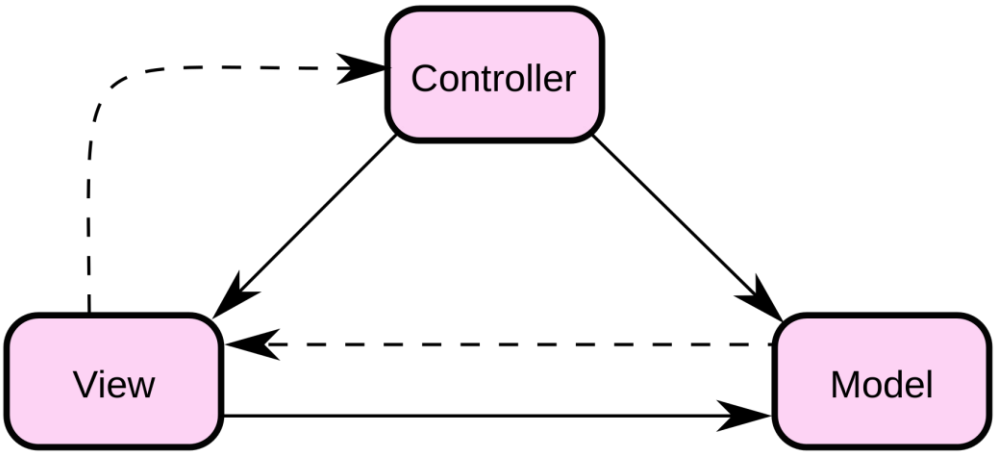

Рис. 2.1. Діаграма взаємодії між компонентами шаблону

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону - гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

2.4. Опис використаних технологій та мов програмування

Необхідною частиною для налаштування свого робочого середовища є таск-менеджери. Вони використовуються для автоматизації однотипних дій програміста під час розробки та збирання проектів.

Це значить, що, задавши необхідну конфігурацію, процеси мінімізації коду, компресії, препроцесорна обробка можуть запускатись всі разом однією командою.

Пряме визначення - це інструмент для автоматизації роботи (конкатенація, мінімізація файлів, робота з препроцесорами, відстеження помилок в коді, робота з зображеннями і багато іншого) розробників. Однак постає питання вибору між двома найбільш відомими, а саме - gulp чи grunt. З огляду на те, що gulp стабільніший і містить більшу кількість різного роду додатків, очевидно, що вибір повинен схилятись саме в його сторону.

Одним головних сегментів, необхідних для запуску менеджера - наявність gulpfile.js файлу, що містить всю конфігураційну інформацію.

Зазвичай ці файли для різних проектів не сильно відрізняються, адже в будь-якому разі існує певний стек завдань, необхідних в кожному проекті, серед них такі: мініфікація, конкатинація, використання препроцесори. 38 Використання таких засобів значно пришвидшують розробку та оптимізують. Однак, окрім того, що було сказано вище про схожість налаштувань для всіх проектів, постає питання також структуризації свого проекту. Зазвичай кожний js-фреймворк несе з собою певні домовленості щодо організації проекту. Саме ці домовленості переважно однакові всюди і немає сенсу щоразу створювати одну і таку ж структуру для різних проектів.

Для пришвидшення роботи в таких випадках використовується генератор коду початкового скелету Yeoman. Отже, для початку варто визначитися, який програмний код він генерує.

В програмування існує таке поняття — скаффолдинг. Воно означає автоматичну побудову структури проекту. За деякими даними воно вперше було використано в Ruby on Rails. Там була спеціальна команда, яка генерувала код головних компонентів проекту, а саме - контролерів і моделей. Зазвичай робота програміста досить однотипна.

Вона найчастіше полягає в тому, щоб вже заздалегідь установлені межі бібліотеки, фреймворку або движка внести власні корективи для реалізації потрібного функціоналу. Таким чином, перший крок роботи по створенню чогось нового у будь-якій системі відтворення базового шаблону тієї або іншої конструкції, яка закладена в її фундамент.

Генератори пропонують позбутися трати часу на початкові кроки створення проекту. Як приклад - позбутися від рутинної операції тиражування вихідних текстів. Однак, така можливість генерації каркаса частин програми вбудована далеко не в кожну систему.

Отже, Yeoman - якісне рішення для побудови скелету. Він надає велику кількість різних генераторів для різних фреймворків. І саме для таких можна знайти готові рішення:

1. AngularJS це фреймворк JavaScript з відкритим програмним кодом, який розробляє Google. Він призначений для розробки 39 односторінкових додатків, що складаються з однієї HTML сторінки з CSS і JavaScript. Його мета - розширення браузерних додатків на основі шаблону модель-вид-контролер (MVC), а також спрощення їх тестування та розробки.

2. React.js - JavaScript бібліотека з відкритим програмним кодом для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових додатків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

3. Ember.js - вільний JavaScript каркас веб-додатків, який реалізує MVC шаблон, призначений для спрощення створення масштабованих односторінкових веб-додатків. Фреймворк використовується такими компаніями як TED, Yahoo!, Twitch.tv і Groupon.

Для реалізації поставленої задачі було обрано React, бо він:

1. Легкий у вивченні. React набагато легше вчиться зважаючи на простоту його синтаксису. Інженери просто повинні згадати свої навички написання HTML і все на цьому. Немає потреби в глибокому вивченні TypeScript, як у випадку з Angular.

2. Високий рівень гнучкості і максимальна чуйність.

3. Віртуальна DOM (document object model), яка дозволяє впорядковувати документи форматів HTML, XHTML або XML в дерево, яке найкраще підходить веб-браузерів для аналізу різних елементів веб-додатки.

4. У поєднанні з ES6 / 7 ReactJS може легко працювати при високих навантаженнях.

5. Зв'язування даних від великих до менших. Це означає такий потік даних, при якому дочірні елементи не можуть впливати на батьківські дані.

6. 100% -а JavaScript-бібліотека з відкритим вихідним кодом, яка отримує безліч щоденних оновлень і поліпшень відповідно до відгуками розробників по всьому світу.

7. Неймовірно легка вага, так як дані, які виконуються на стороні користувача, можуть легко бути представлені на стороні сервера в той же самий час.

8. Міграція між версіями, як правило, дуже проста. Також Facebook надає «codemods» для автоматизації більшої частини цього процесу.

2.5. Опис структури програми та алгоритмів її функціонування

На етапі проектування програмного забезпечення визначається його структура дані які, які є частиною системи інтерфейси взаємодії системних компонентів і алгоритми що використовуються.

Кінцевими результатами процесу проектування є точні специфікації на алгоритми і структури даних.

Етапи процесу проектування:

– Архітектурне проектування. Визначаються і документуються підсистеми і взаємозв'язки між ними.

– Узагальнена специфікація. Для кожної підсистеми розробляється узагальнена специфікація на її сервіси і обмеження.

– Проектування інтерфейсів. Для кожної підсистеми визначається і документується її інтерфейс.

– Компонентне проектування. Проводиться розподіл системних функцій (сервісів) по різних компонентах і інтерфейсах.

– Проектування структур даних. Детально розробляються структури даних, необхідні для реалізації програмної системи.

– Проектування алгоритмів. Детально розробляються алгоритми, призначені для системних сервісів.

Описана схема є загальною і на практиці може пристосовуватися до розробки конкретного програмного забезпечення.

Для процесу архітектурного проектування можна виділити декілька загальних етапів.

Для розробки програмного забезпечення обрана модель “клієнт-сервер”, яка використовується для багатокористуваць ских систем: на сервері зберігаються дані й програми їхньої обробки; інші комп'ютери мережі (клієнтські станції) відправляють серверу запит на обробку даних; сервер обробляє запит і клієнтові відправляє тільки результати запиту.

При використанні технології клієнт-сервер додаток розділяється на дві частини. Клієнтська частина забезпечує зручний графічний інтерфейс і розміщується на комп'ютері користувача. Серверна частина здійснює управління даними, поділ інформації, адміністрування і забезпечує безпеку інформації. Клієнтський додаток формує запити до сервера бази даних, на якому виконуються відповідні команди. Результати виконання запитів пересилаються клієнту.

При розробці розподілених інформаційних систем в організації взаємодії клієнтської і серверної частини виділяються наступні важливі в практичному сенсі завдання:

– Перенесення персональної бази даних на сервер для подальшого її колективного використання як корпоративної бази даних.

– Організація запитів до корпоративної базі даних, розміщеної на сервері, з боку комп'ютера-клієнта.

– Розробка клієнтського додатка для віддаленого доступу до корпоративної бази даних з боку комп'ютера-клієнта.

Переваги систем, базованих на клієнт-серверній архітектурі:

– мінімум затрат на обслуговування бізнес-процесів;

– максимальна оперативність при оперуванні даними;

– зручність в обслуговуванні, більшість операцій може виконуватись автоматично;

– один працівник може легко обслуговувати кілька процесів одночасно без особливих зусиль;

– мінімум затрат на комунікації між підрозділами компанії;

– оперативне і гнучке отримання звітів про діяльність компанії;

– веб-сервер, СКБД і програмні модулі що забезпечують функціонування бізнес-логіки як правило розміщуються на одному комп’ютері;

– працівникові для роботи з системою потрібний лише звичайний веб-браузер;

– роботу з системою можна здійснювати з будь якого комп'ютера що приєднаний до інтернету.

Вся система працює під управлінням програмного забезпечення, яке забезпечує реалізацію основних функцій системи. Загальна структура програмного забезпечення представлена на рис. 2.2.

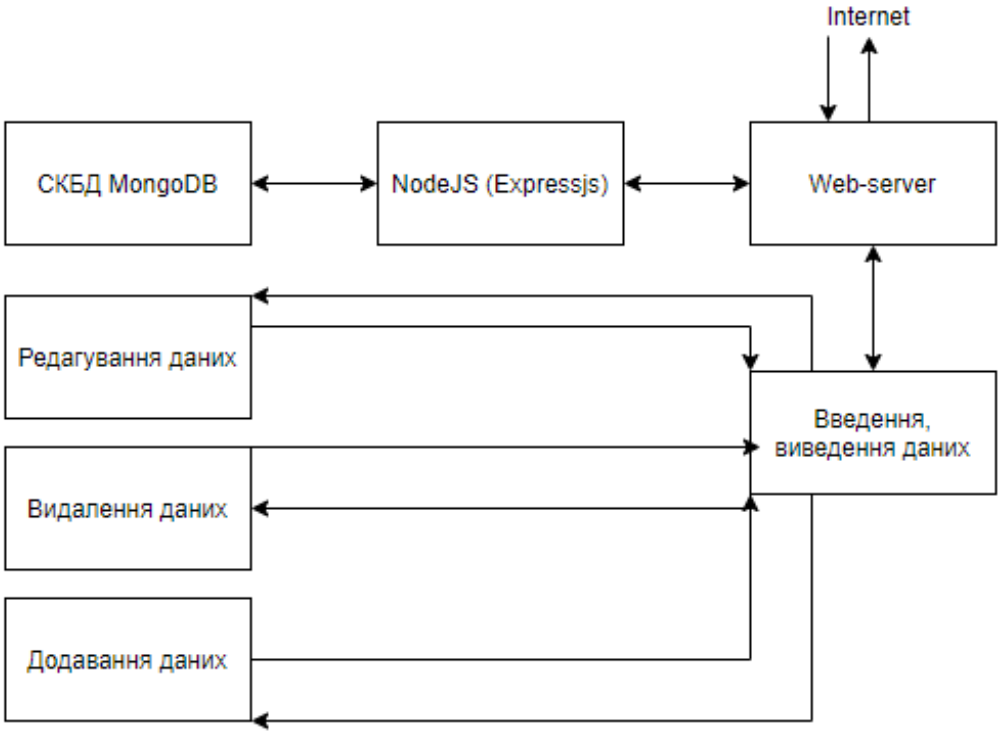

Рис. 2.2. Архітектура програмного забезпечення системи

Інтерфейс включає в себе два компоненти: обладнання і програмне забезпечення. Існують інтерфейси для різних систем. UI є засобом для виконання двох типів взаємодії:

1. Введення (Input), що дозволяє користувачам управляти системою.
2. Висновок (Output), що дозволяє системі демонструвати ефект від вироблених користувачем маніпуляцій.

Основною метою при розробці інтерфейсу для взаємодії людини з машиною є створення UI, який дозволить легко (інтуїтивно), ефективно, і з задоволенням (user friendly) домагатися бажаного результату під час роботи з машиною. Ідеальним результатом вважається той, при якому оператор (або користувач) виробляє мінімальні зусилля для введення, отримуючи від машини бажаний висновок, при цьому машина мінімізує у висновку зайві дані. Існує безліч типів призначеного для користувача інтерфейсу, однак до тих, що відносяться до інформаційних технологій варто віднести такі:

– Графічний інтерфейс користувача (GUI). Користувач здійснює операцію введення через пристрої, такі як комп'ютерна клавіатура і миша, а машина і забезпечує графічний висновок на моніторі комп'ютера.

– Веб-інтерфейс (WUI). Це сукупність засобів, за допомогою яких користувач взаємодіє з веб-сайтом або будь-яким іншим додатком через 30 браузер. Нові реалізації використовують Java, JavaScript, AJAX, Adobe Flex, Microsoft. NET, або аналогічні технології. З їх допомогою можливий контроль в режимі реального часу, що виключає необхідність оновлення, що складається в основі HTML браузерів.

– Адміністративні веб-інтерфейси. Використовуються для роботи з серверами і віддаленими комп'ютерами. Часто називаються панеллю керування.

– Сенсорні екрани. Це дисплеї, які беруть введення при торканні пальцями або стилусом. Використовується в мобільних пристроях, вуличних автоматах різних типів і т. д.

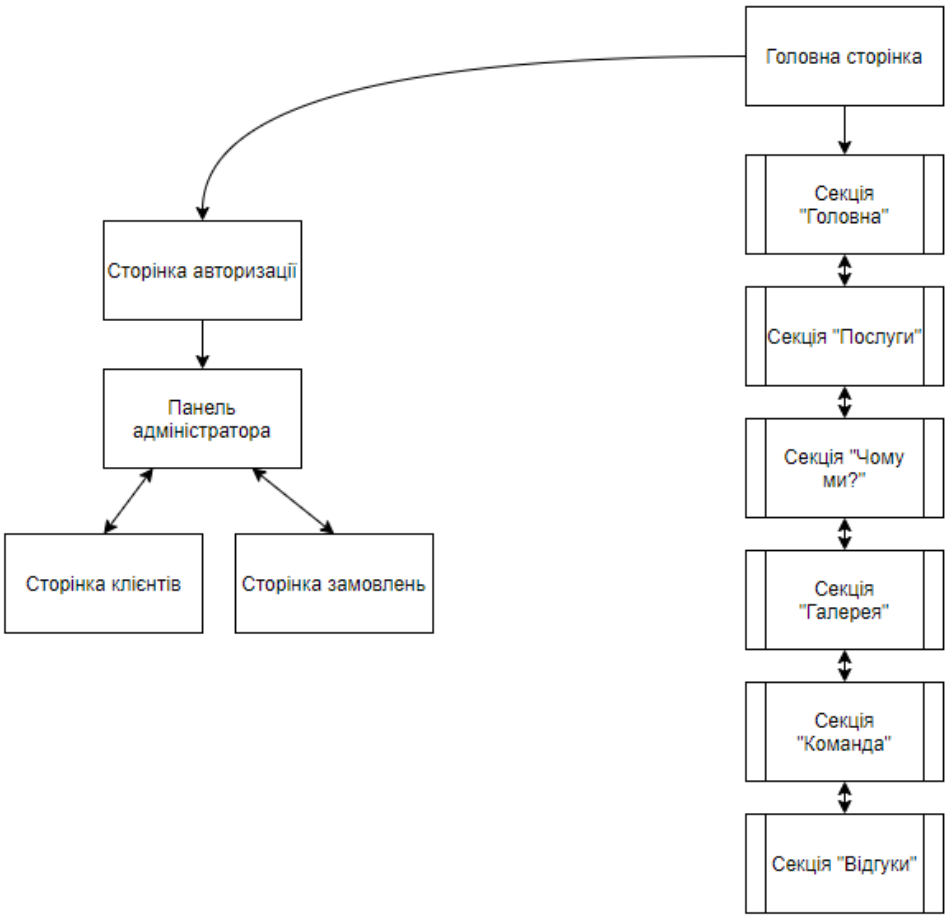

Рис. 2.3. Логічна схема сайту

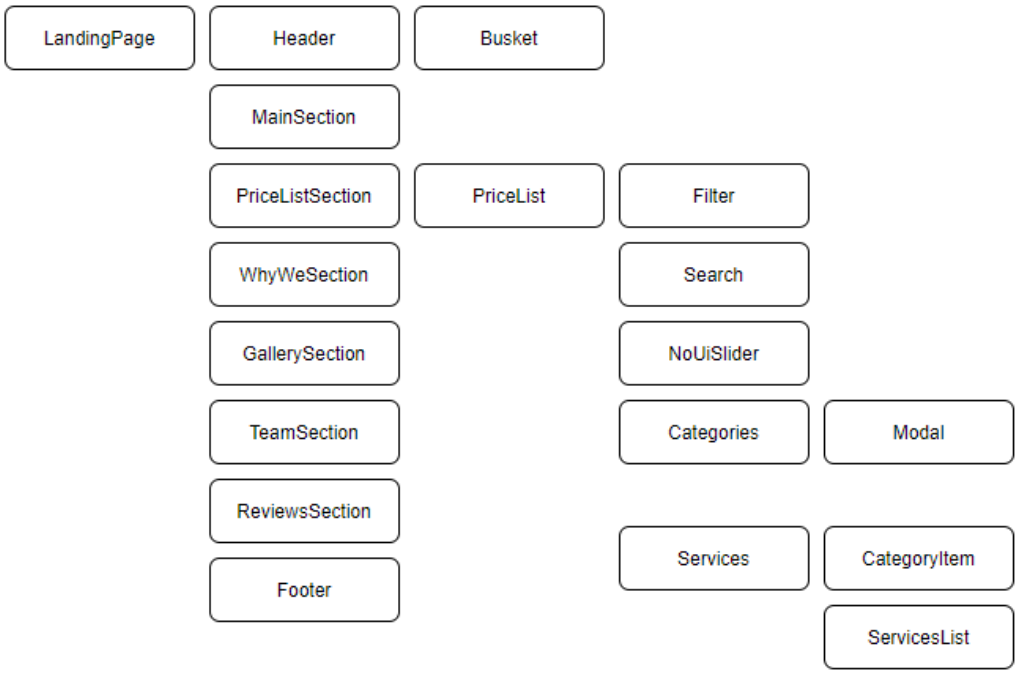

Рис. 2.4. Фізична схема Головної сторінки

Логічне (див.рис.2.3.) і фізичне уявлення (див.рис.2.4.) не треба змішувати і краще всього робити їх незалежними один від одного. Логічна структура сайту може мінятися, тоді доведеться перейменовувати файли і каталоги, відповідно міняти посилання усередині готових сторінок - це збільшує вірогідність внесення помилок у вже перевірені сторінки.

Практично любий сайт є сукупністю сторінок, які містять тексти, картинки, мультимедійні чи інші об’єкти. Для комфортного перебування відвідувача на сайті і для полегшення пошуку потрібної інформації, сайт повинен мати чітку і продуману структуру.

Структура сайту - це внутрішній устрій сайту, його «кістяк», розташування сторінок, розділів, підрозділів, додаткових матеріалів. І першочерговим завданням дизайнера є створення стрункого порядку з хаотичного скупчення інформації.

UI - це різновид інтерфейсів, в якому одна сторона представлена людиною (користувачем), інша - машиною та пристроєм. Являє собою сукупність засобів і методів, за допомогою яких користувач взаємодіє з

різними, найчастіше складними, машинами, пристроями та апаратурою. Іншими словами, призначений для користувача інтерфейс є системою, за допомогою якої люди (користувачі) взаємодіють з машиною.

Розробка UI - це створення інтерфейсу, який забезпечує найкращий, простий, приємний і не обтяжує спосіб взаємодії користувача з продуктом.

Більшу частину роботи під час створення інтерфейсу становить спостереження за поведінкою користувача, що дозволяє приймати рішення, засновані на зібраних даних.

Враховуючи дані вимоги, можна стверджувати, що в якості інтерфейсу варто розробити односторінковий додаток (SPA), також відомий як односторінковий інтерфейс (SPI) - це веб-застосунок чи веб-сайт, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою. За даних умов реалізація серверного рендерингу буде не оптимальним рішенням.

В односторінковому додатку весь необхідний код (HTML, JavaScript, та CSS) завантажується разом із сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача на іншу сторінку в процесі роботи з нею. Взаємодія з односторінковим додатком часто включає в себе динамічний зв'язок з веб-сервером.

Розробка такого типу сторінок досить класичне завдання для front-end розробника. Інструментами для розробки в такому випадку будуть:

1. JavaScript (JS) - динамічна, об'єктно орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також 35 використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java і C#), розробки ігор,

стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ, всередині PDF-документів тощо. JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну, декларативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу. Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманний «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

2. HTML5 - наступна версія мови HTML. До складу робочої групи з HTML5 AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera та кілька сотень інших виробників. 28 жовтня 2014 консорціум W3C оголосив про надання набору специфікацій HTML5 статусу рекомендованого стандарту. Цікаво, що у цьому вигляді специфікації HTML 5.0 були сформовані ще два роки до того, після чого робота була зосереджена на проведенні тестування та оцінки сумісності доступних реалізацій. На час стандартизації HTML5 вже давно став стандартом де-факто і активно використовується у веб-36 застосунках. Фактичне затвердження стандарту лише формально поставило крапку в просуванні HTML5 і підтвердило повсюдність і коректність його реалізації. Специфікації HTML5 не обмежуються тільки розміткою і включають в себе низку веб-технологій, котрі у сукупності формують відкриту веб-платформу - програмне оточення для роботи крос-платформенних застосунків, здатних взаємодіяти з обладнанням, і які підтримують засоби для роботи з відео, графікою і анімацією, що надає розширені мережеві можливості.

3. CSS - спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних. CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність

або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі - сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

Саме цей стек мов зазвичай дозволяє розробляти веб-інтрейфейси. Однак за останній час з‘явилась велика кількість додаткових засобів, що прискорюють та полегшують розробку.

Якщо говорити про такі інструменти, то спершу варто зазначити, що більшість з них неможливо використовувати без наявного Node.js інтерпритатора. Node.js - платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.

Платформа Node.js перетворила мову JavaScript, що здебільшого використовувалась в браузерах на мову загального використання з великою спільнотою розробників.

Використання цієї платформи необхідне впершу чергу через пакетний менеджер npm. npm - це пакетний менеджер Node.js. З його допомогою можна керувати модулями і залежностями. Саме він дозволяє налаштувати робоче середовище відповідно до поставлених вимог.

Відомо, що Node.js використовується здебільшого для back-end розробки, однак npm містить велику кількість пакетів, призначених і для розробки інтерфейсів.

Всіх користувачів інформаційної системи, умовно можно поділити на 2 великих класи: користувач та адміністратор. Кожний з класів має свою права та повноваження (рис.2.5.)

Рис. 2.5. Діаграма User Case

Згідно з завданням, мета даної роботи полягає у проектуванні та розробці інформаційної системи для студії організацій свят. Для реалізації поставленої було вирішено скористатися двома методами для проектування БД. Першим підходом до проектування є метод сутність - зв'язок, другим же є метод нормалізації відносин.

Модель сутність - зв'язок або по іншому ER-модель - це модель даних, що дозволяє описувати концептуальні схеми предметної області. Дана модель використовується при високорівневої проектуванні баз даних і дозволяє з її допомогою виділити ключові сутності і позначити зв'язки, які встановлюються між цими сутностями. Нормалізація відносин - це формальний апарат обмежень на формування відносин, який дозволяє усунути дублювання і потенційну суперечливість збережених даних, таким чином зменшує трудовитрати на ведення БД.

Процес нормалізації полягає в декомпозиції вихідних відносин на більш прості відносини, до тих пір, поки не вийде такий проект БД, в якому «кожен факт з'являється лише в одному місці».

Концептуальна модель - абстрактна модель, що визначає структуру модельованої системи, властивості її елементів та причинно-наслідкові зв’язки, властиві системі і суттєві для досягнення мети моделювання.

Сутності в базі даних - це будь-які об’єкти в базі даних, які можна виділити виходячи з суті предметної області для якої розробляється база даних.

Основними сутностями розробляємої бази даних є:

- Клієнт.
- Послуга.
- Замовлення.
- Одиниця.
- Категорія.
- Авторизація.

Опис типів сутностей приведені у таблиці 2.1.

Таблиця 2.1.

Опис типів сутностей

Ім‘я типу сутності	**Опис**
Клієнт	Данні про клієнта
Поуслуга	Данні про послугу
Замовлення	Додаткові послуги
Одиниця	Характеристика послуги
Категорія	Характеристика послуги
Авторизація	Данні співробітника для входу в систему

Атрибути в базі даних - це іменовані характеристики, що є деякими властивостями сутності (табл. 2.2.).

Таблиця 2.2.

Визначення атрибутів сутностей

Ім‘я сутності	Атрибути	Опис атрибута	Тип	Обмеження
Клієнт	ID	ID	Ціле	>0
	Ім’я	Ім’я клієнта	Рядок	<25
	Телефон	№ телефону	Рядок	>0
	Замовлення	ID замовлення	Ціле	>0
Замовлення	ID	ID	Ціле	>0
	Номер	№ замовлення	Ціле	>1000
	Клієнт	ID клієнта	Ціле	>0
	Послуги	ID послуги	Ціле	>0
	Ціна	Ціна замовлення	Ціле	>0
Послуга	ID	ID	Ціле	>0
	Назва	Назва послуги	Рядок	<50
	Ціна	Ціна послуги	Ціле	>0
	Одиниця	ID одиниці	Ціле	>0
	Категорія	ID категорії	Ціле	>0
Авторизація	ID	ID	Ціле	>0
	Логін	Логін співробітника	Рядок	<16
	Пароль	Пароль співробітника	Рядок	<16
Одиниця	ID	ID	Ціле	>0
	Назва	Назва одиниці	Рядок	<30
Категорія	ID	ID	Ціле	>0
	Назва	Назва категорії	Рядок	<50

Зв’язки між сутностями – це деякі відношення між двома типами сутностей.

Розрізняють 3 типи зв’язків між сутностями:

а) “Один до одного” або 1:1. Це означає, що одному екземпляру сутності може відповідати тільки один екземпляр іншої сутності;

б) “Один до багатьох” або 1:N. Це означає, що одному екземпляру сутності може відповідати будь-яка кількість екземплярів іншої сутності;

в) “Багато до багатьох” або N:N. Це означає, що декільком екземплярам однієї сутності може відповідати декілька екземплярів іншої сутності.

Таблиця 2.3.

Ввизначення зв’язків між сутностями

Ім‘я сутності	Зв’зок	Ім‘я сутності	Тип Зв’язку
Клієнт	Робить	Замовлення	1:n
Замовлення	Містить	Послуга	n:n
Послуга	Містить	Одиниця	n:1
Клієнт	Містить	Авторизація	1:1
Послуга	Містить	Категорія	n:1

В результаті буде отримана наступна концептуальна схема, що приведена на рис. 2.6.

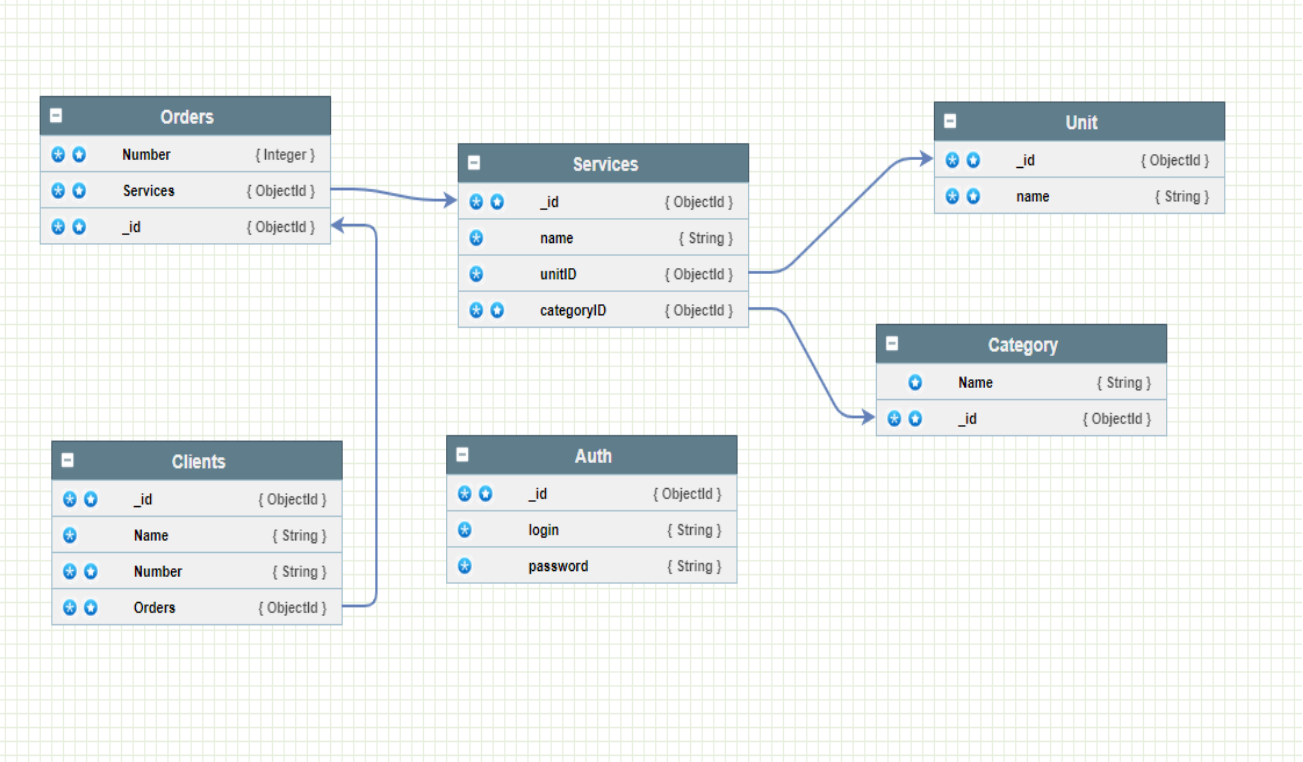

Рис. 2.6. Концептуальна схема моделі

Для даного проекту було обрано документоорієнтовану базу даних MongoDB (NoSQL).

Документоорієнтована СУБД, спеціально призначена для зберігання ієрархічних структур даних (документів) і зазвичай реалізована за допомогою

підходу NoSQL. В основі документоорієнтованих СУБД лежать документні сховища, які мають структуру дерева.

Структура дерева починається з кореневого вузла і може містити кілька внутрішніх і листових вузлів. Листові вузли містять дані, які при додаванні документа заносяться в індекси, що дозволяє навіть при досить складну структуру знаходити місце (шлях) шуканих даних. API для пошуку дозволяє знаходити за запитом документи і частини документів. На відміну від сховищ типу ключ-значення, вибірка за запитом до документному сховища може містити частини великої кількості документів без повного завантаження цих документів в оперативну пам'ять.

Документи можуть бути організовані (згруповані) в колекції. Їх можна вважати віддаленим аналогом таблиць реляційних СУБД, але колекції можуть містити інші колекції. Хоча документи колекції можуть бути довільними, для більш ефективного індексування краще об'єднувати в колекцію документи зі схожою структурою.NoSQL - термін, що позначає ряд підходів, спрямованих на реалізацію систем управління базами даних, що мають суттєві відмінності від моделей, що використовуються в традиційних реляційних СУБД з доступом до даних засобами мови SQL. Застосовується до баз даних, в яких робиться спроба вирішити проблеми масштабованості та доступності за рахунок атомарности і узгодженості даних.

MongoDB - це NoSQL сховище даних, вкрай зручне для зберігання інформації, яка не може бути нормально структурована в рамках реляційних баз даних.

MongoDB - це СУБД з відкритим вихідним кодом, яка не потребує опису схеми таблиць і написана на мові C ++. Завдяки спеціалізації бази даних вдалося відійти від принципу «один розмір під все», а за рахунок мінімізації методів роботи з транзакціями з'явилася можливість вирішення цілої низки проблем, пов'язаних з нестачею продуктивності, що також сприяло поліпшенню горизонтального масштабування.

Документи в MongoDB зберігаються в JSON або BSON, робота з такою моделлю простіше кодується і простіше управляється, а внутрішня угруповання релевантних даних забезпечує додатковий виграш у швидкодії. MongoDB, на думку розробників, повинна заповнити розрив між найпростішими NoSQL-СУБД, що зберігають дані у вигляді «ключ – значення і великими реляційними СУБД.

Що є в MongoDB:

- гнучка мова для формування запитів;
- динамічні запити;
- індексація колекцій;
- профілювання запитів;
- журнал операцій записів;
- відмовостійкість і масштабованість;
- асинхронна реплікація і шардінг;
- повнотекстовий пошук з підтримкою морфології.

Робота з MongoDB реалізована на мовах програмування Java, C ++, C #, PHP, Python, Perl, Ruby та інших, а відповідні компоненти є в багатьох фреймворками (в Ruby on Rails і в Yii, наприклад).

Було вирішено обрати MongoDB для даного проекту, бо вона мала декілька переваг:

- Вкрай зрозуміла структура кожного об'єкта.
- Легко масштабується.
- Відсутність складних JOIN запитів.

Також на моє рішення вплинуло те, що MongoDB входить як стандарт до стеку технологій, на якому я збиралася писати.

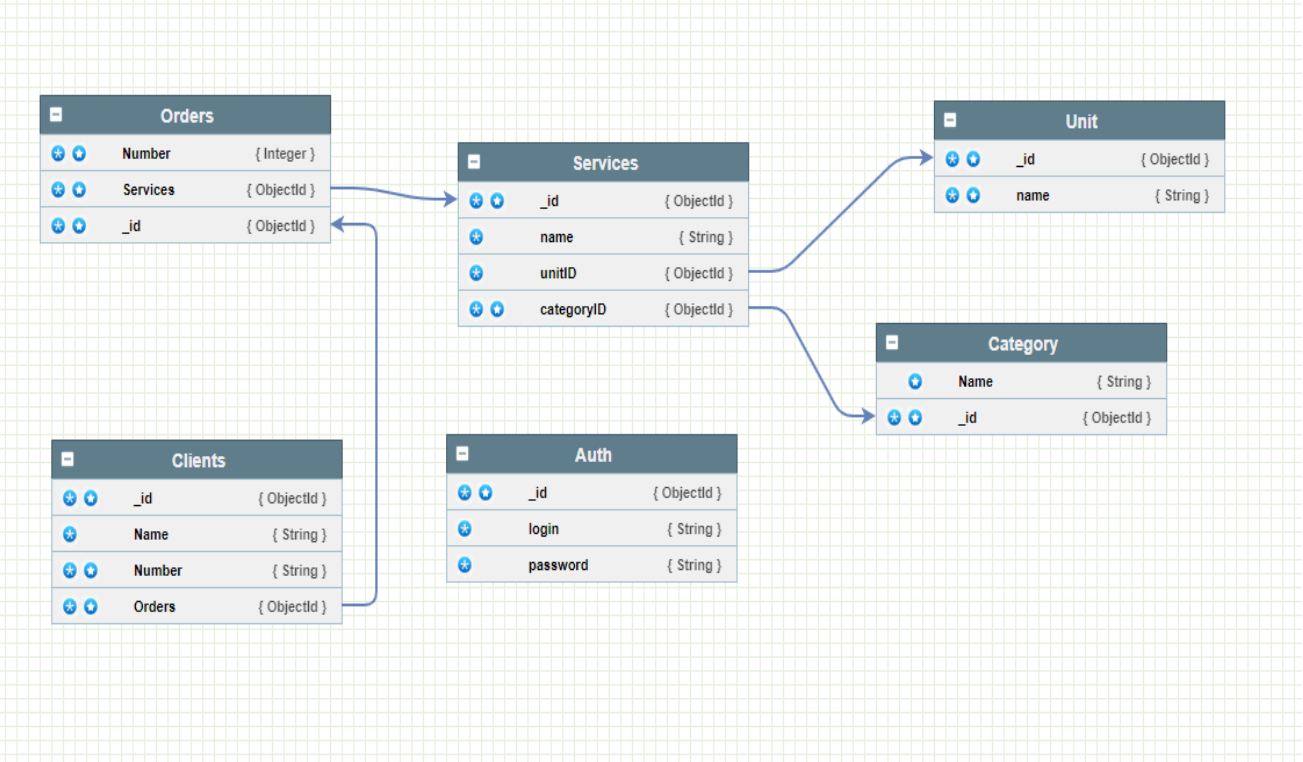

Рис. 2.7. Логічна схема моделі

2.6. Обгрунтування та організація вхідних та вихідних даних програми

Вхідними даними є введені користувачем дані або обрані управляючі команди. Вихідними даними є дані, що отримує користувач від системи.

Вхідними даними будуть заповнення таблиць бази даних.

Вихідними даними будуть заповненні таблиці та запити до бази даних

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

При розробці системи було використано Ноутбук Dell Vostro 15 3501 з наступними характеристиками: екран 15.6" WVA (1920x1080) Full HD, глянсовий з антивідблисковим покриттям / Intel Core i3-1005G1 (1.2 — 3.4 ГГц) / RAM 8 ГБ / SSD 256 ГБ / Intel UHD Graphics / без ОД / LAN / Wi-Fi / Bluetooth / вебкамера

2.7.2. Використані програмні засоби

Розроблена в рамках кваліфікаційної роботи інформаційна система була реалізована на мові JavaScript в середовищі Visual Studio Code з системою керування базами даних MongoDB.

2.7.3. Виклик та завантаження програми

Виклик та завантаження розробленої інформаційнох системи виконується за допомогою веб-браузера з підтримкою JavaScript.

2.7.4. Опис інтерфейсу користувача

На приведених нижче рисунках (рис.2.8. – 2.21.) описаний приклад роботи програми. Інтерфейс є зручний та інтуітивно зрозумілий.

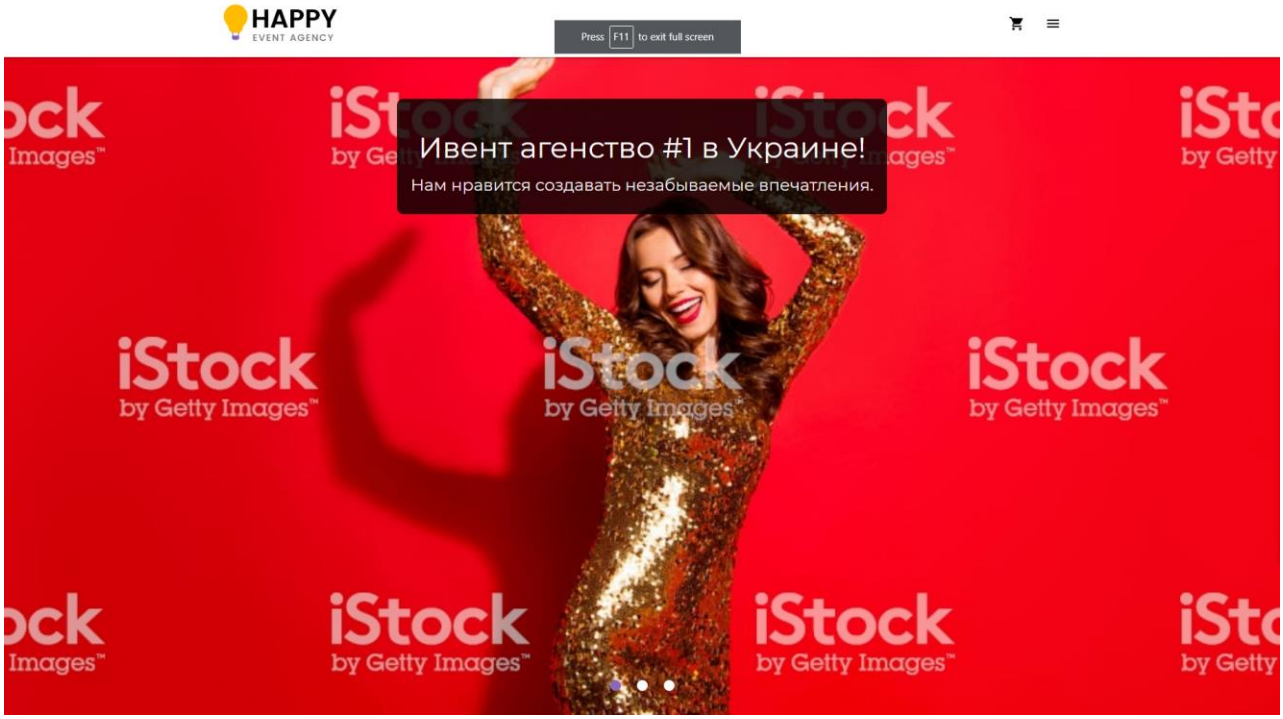

Рис. 2.8. Головна сторінка

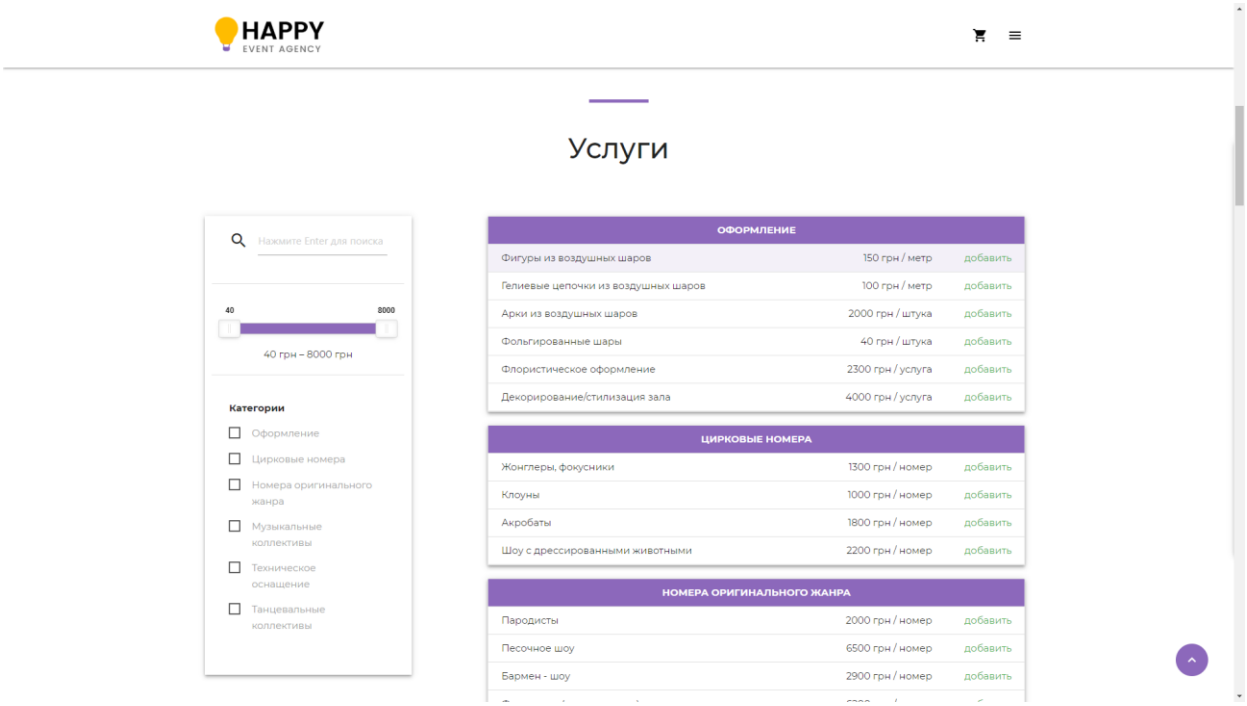

Рис. 2.9. Сторіка «Послуги»

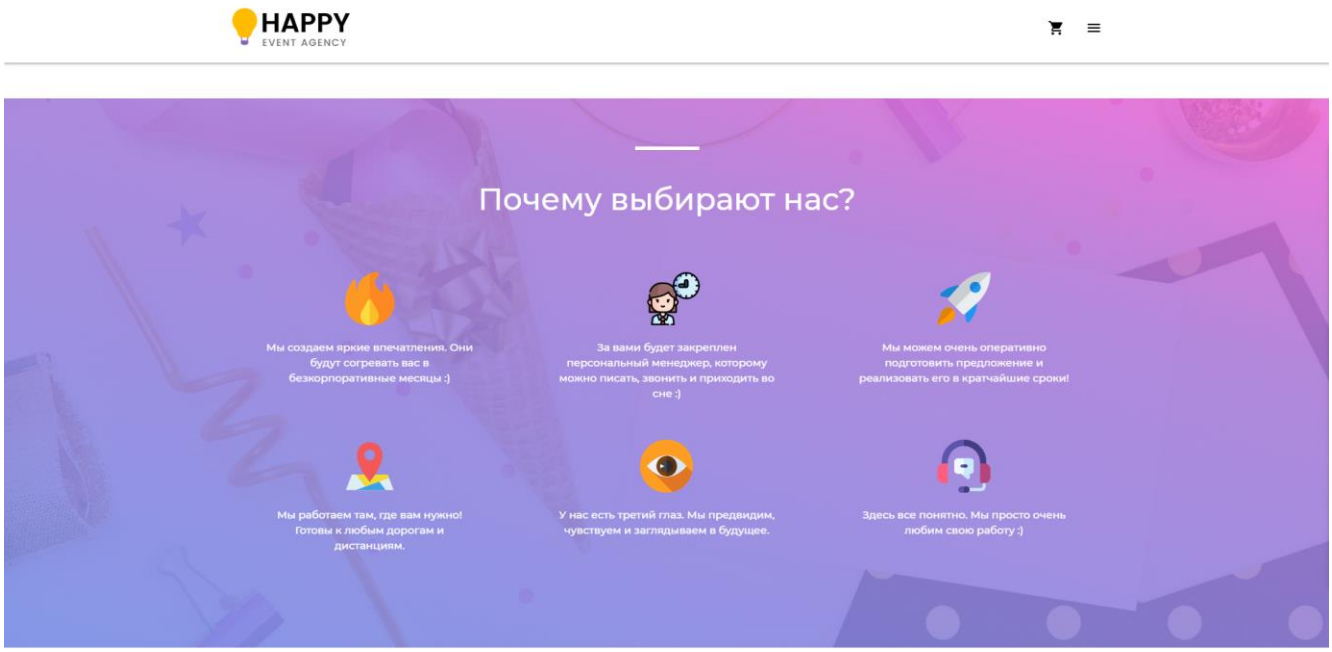

Рис. 2.10. Сторінка «Чому обирають нас?»

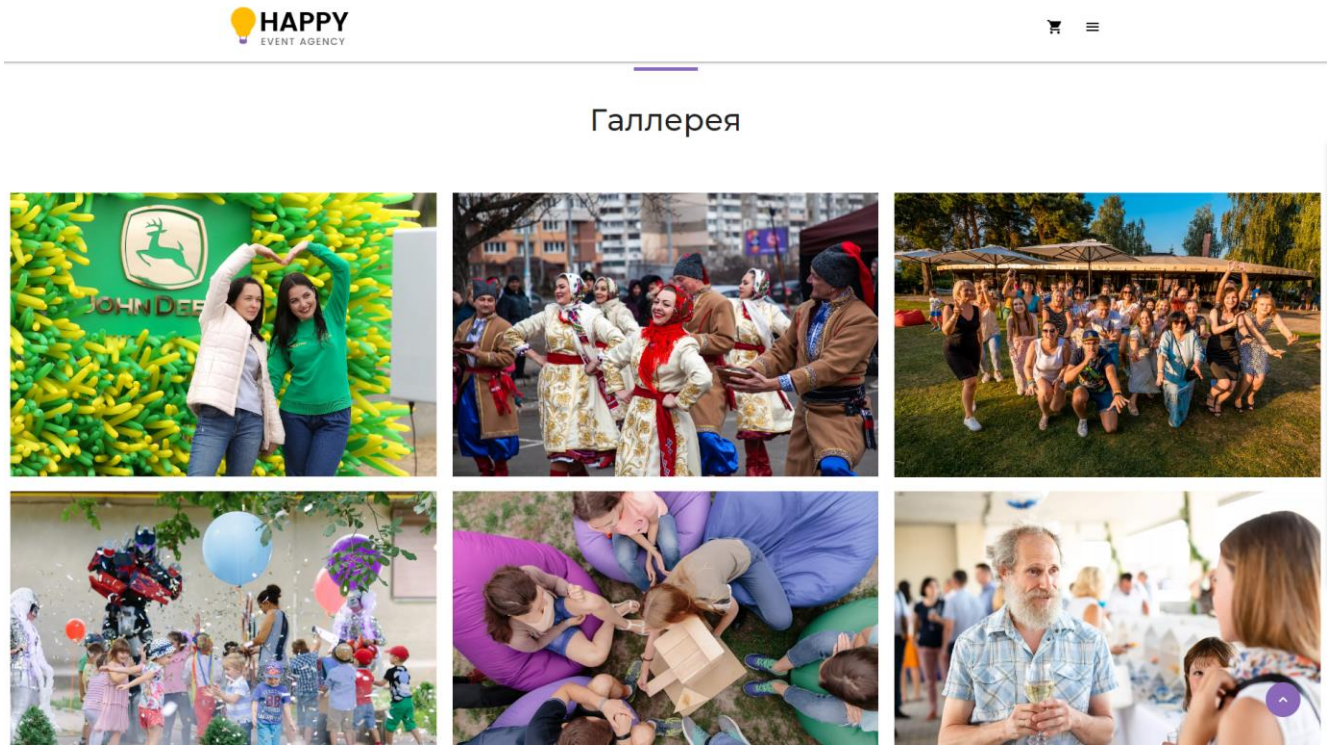

Рис. 2.11. Сторінка «Галерея»

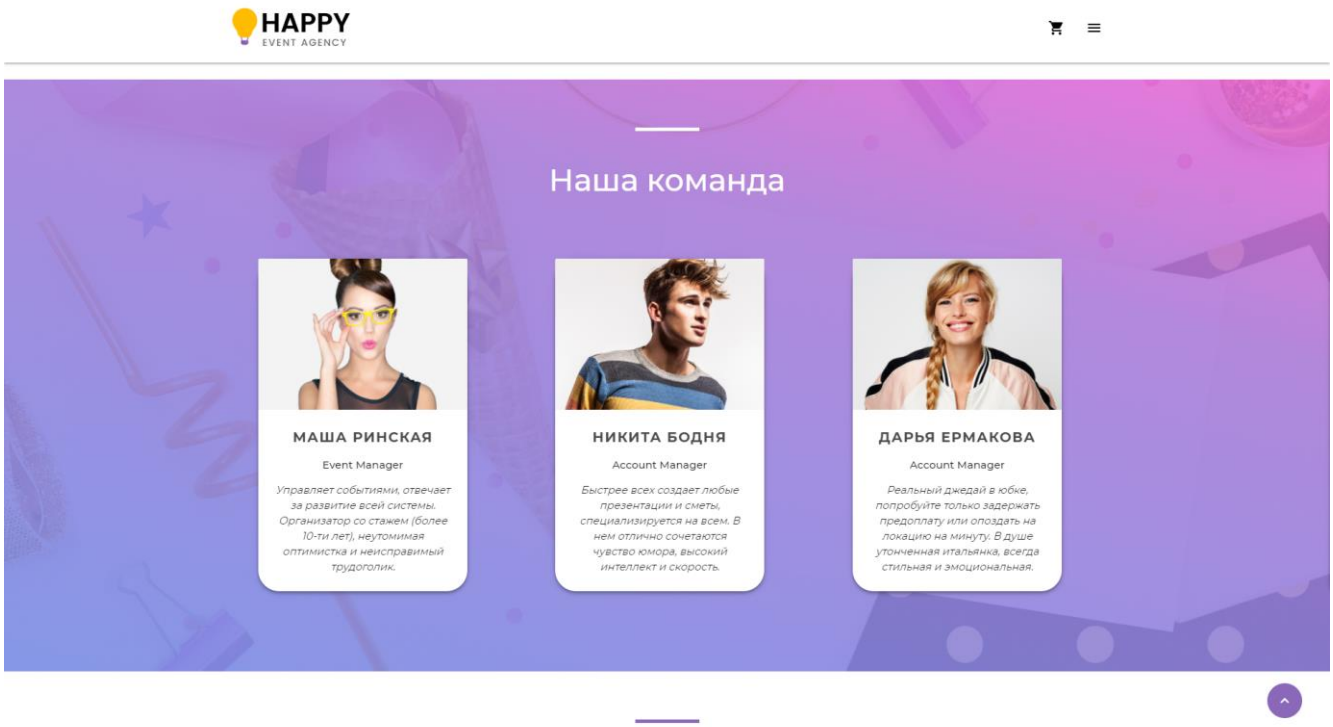

Рис. 2.12. Сторінка «Команда»

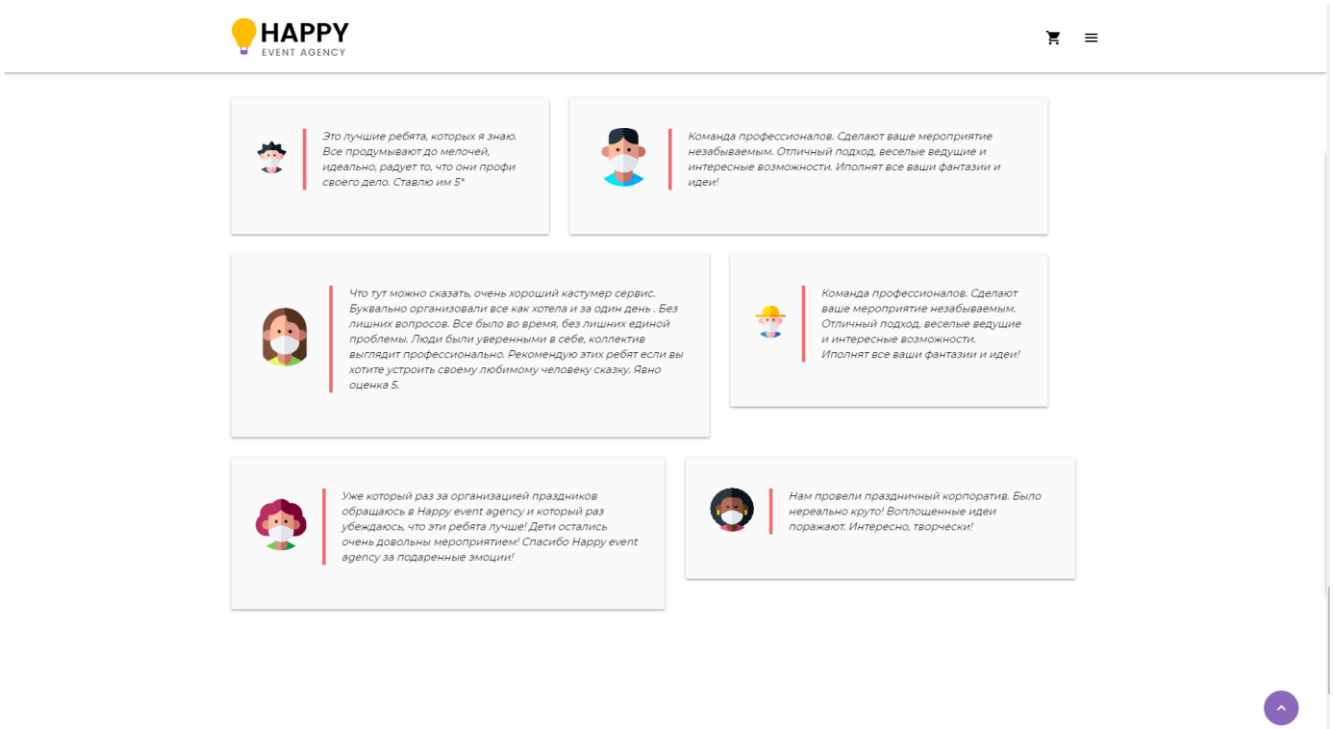

Рис. 2.13. Сторінка «Відгуки»

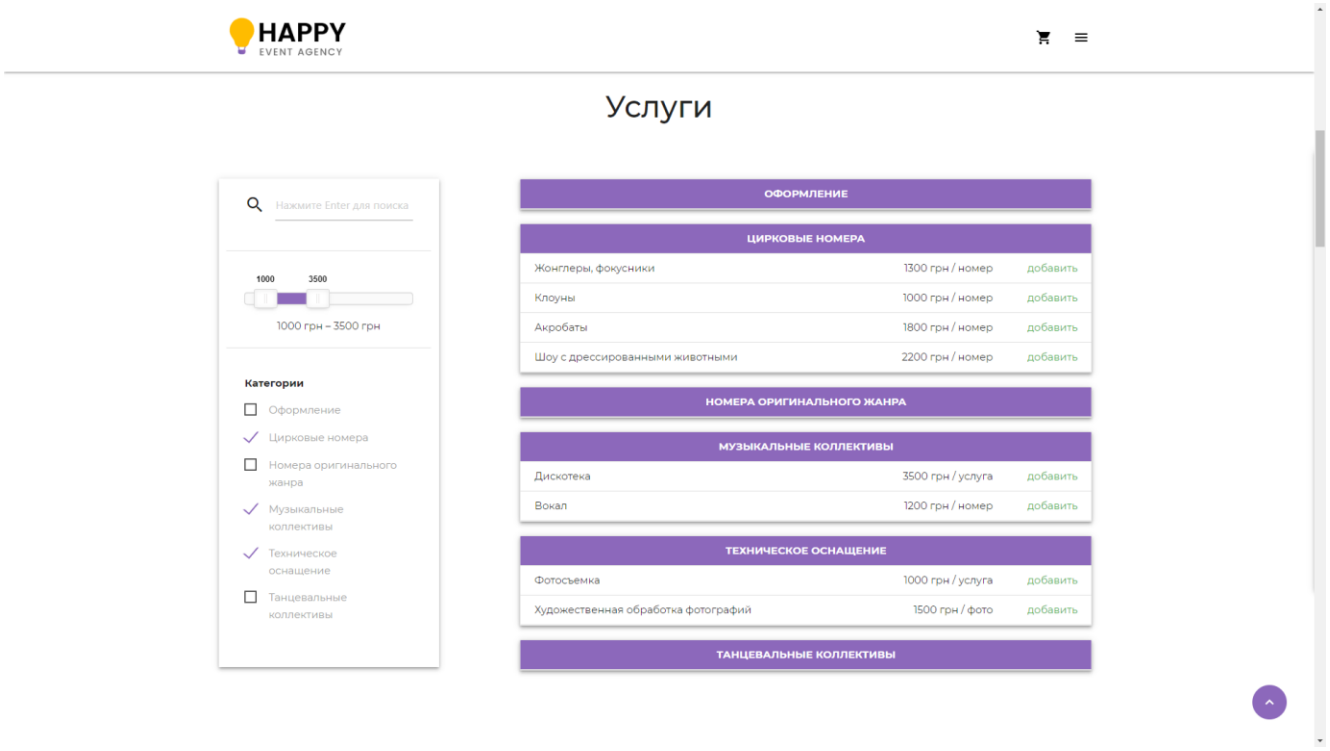

Рис. 2.14. Сторінка роботи фільтрів

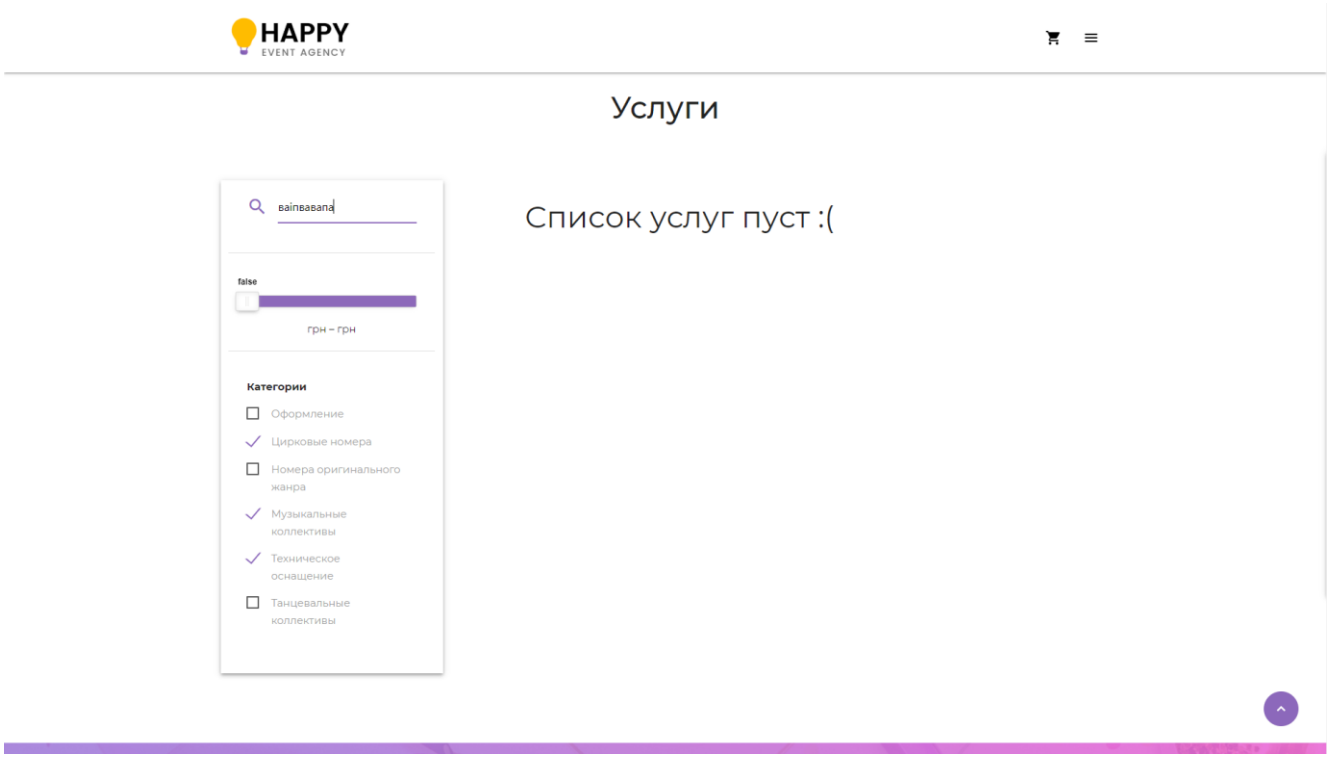

Рис. 2.15. Сторінка роботи пошуку

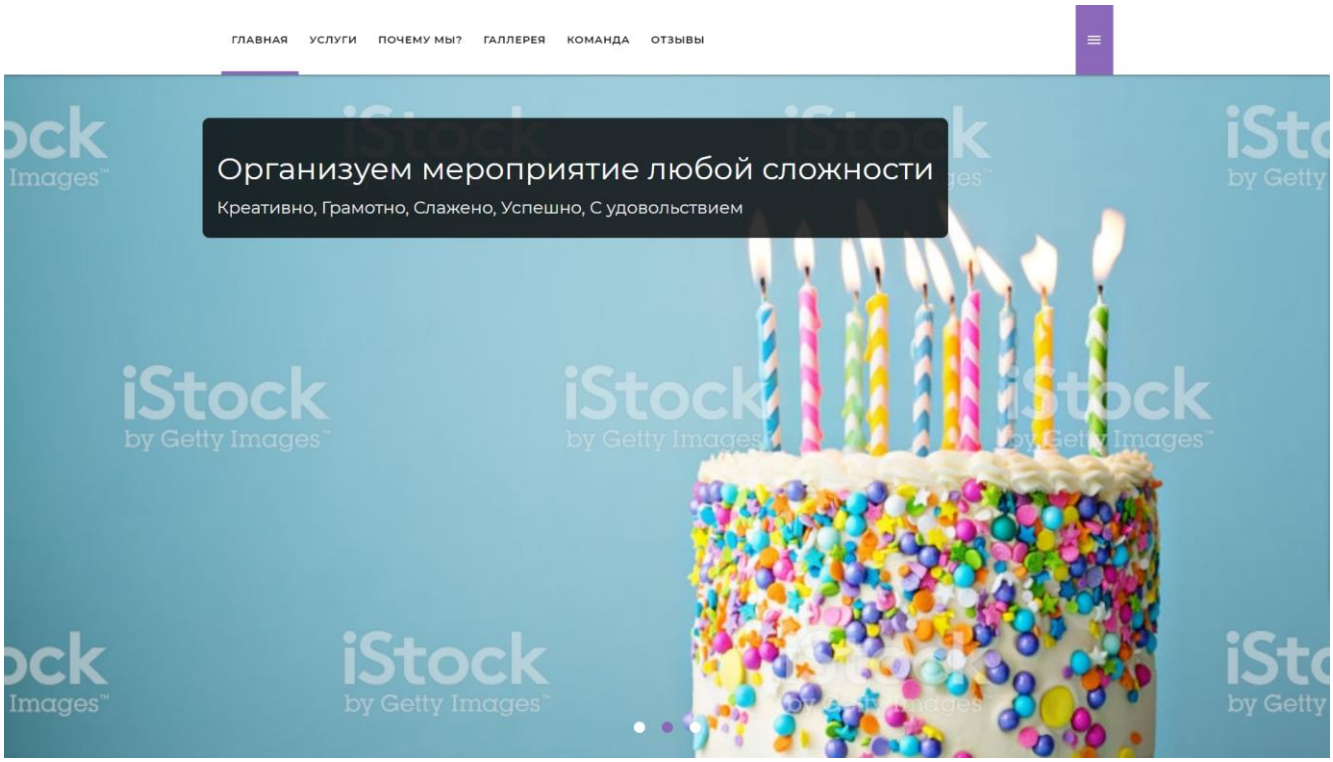

Рис. 2.16. Сторінка головного меню

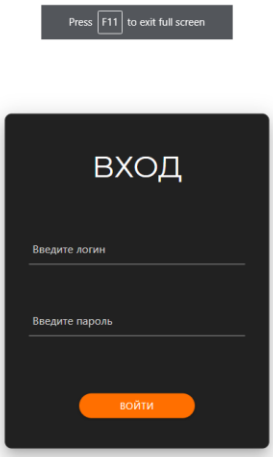

Рис. 2.17. Сторінка авторизації

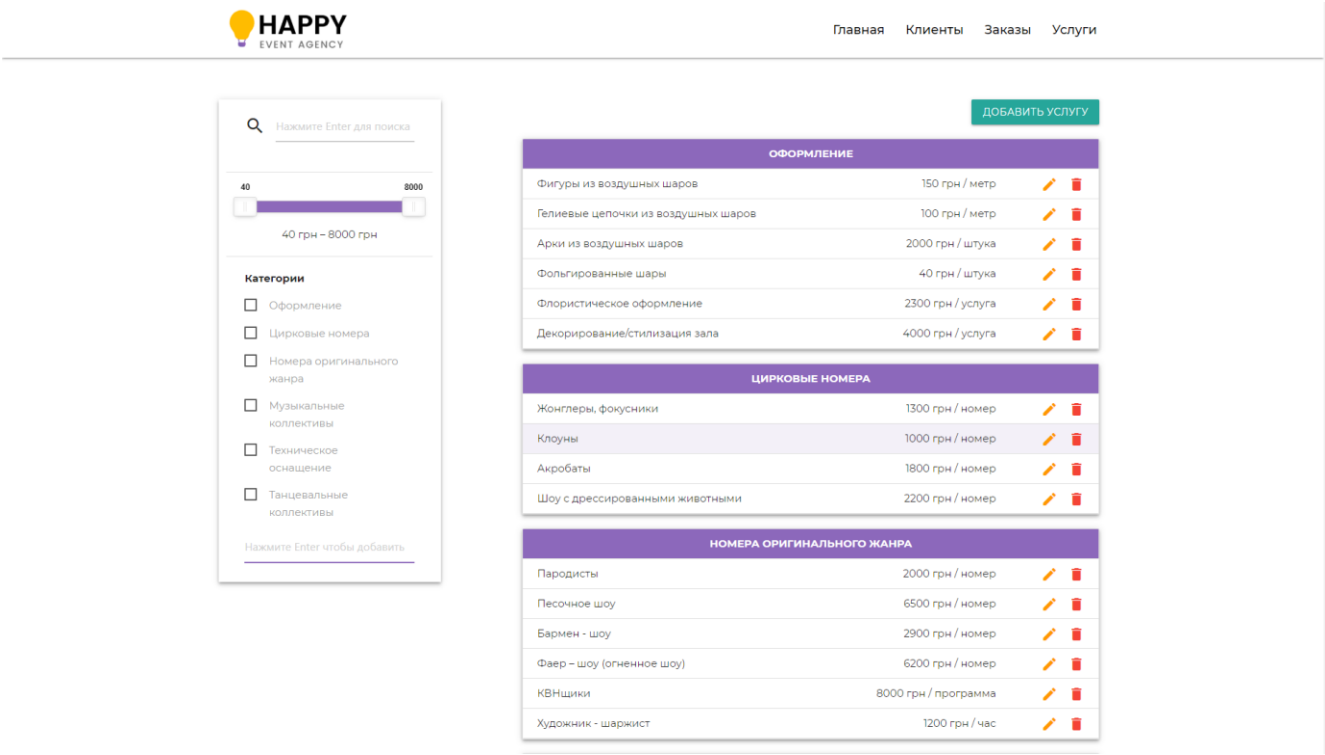

Рис. 2.18. Сторінка управління послугами

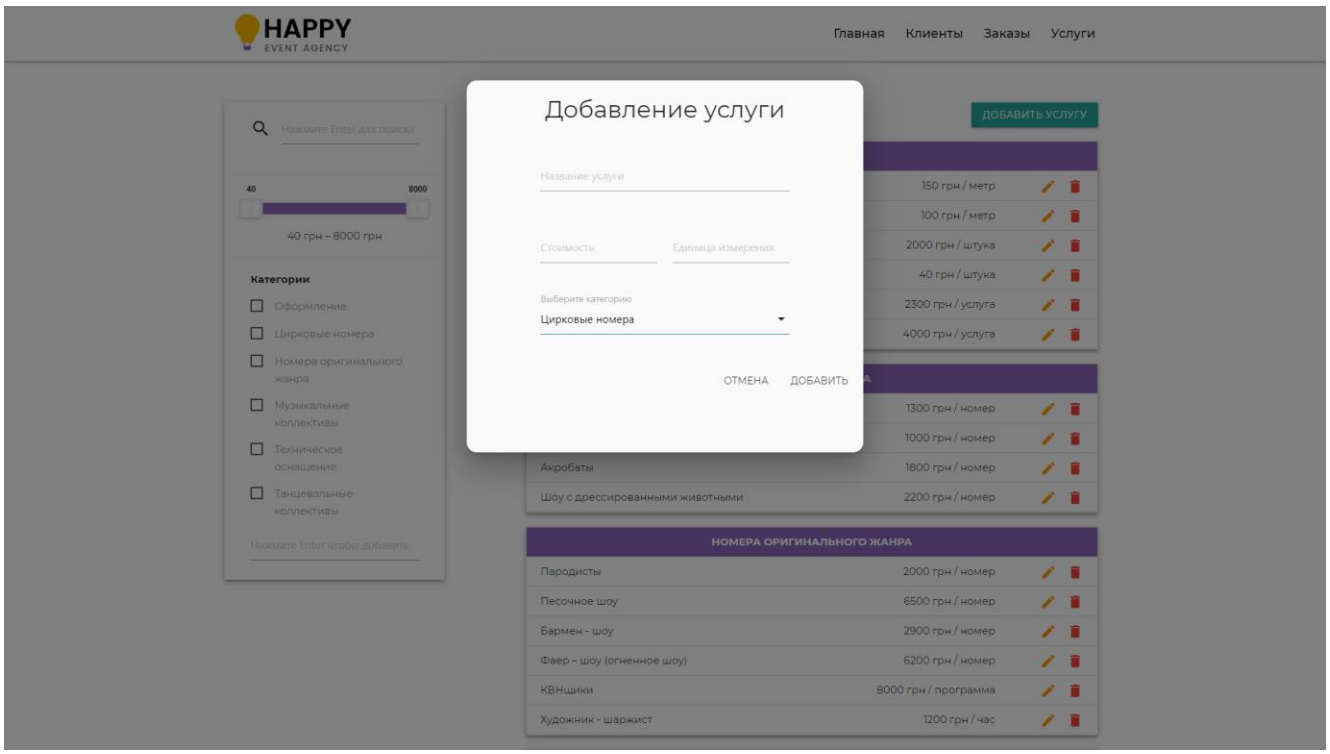

Рис. 2.19. Сторінка створення нової послуги

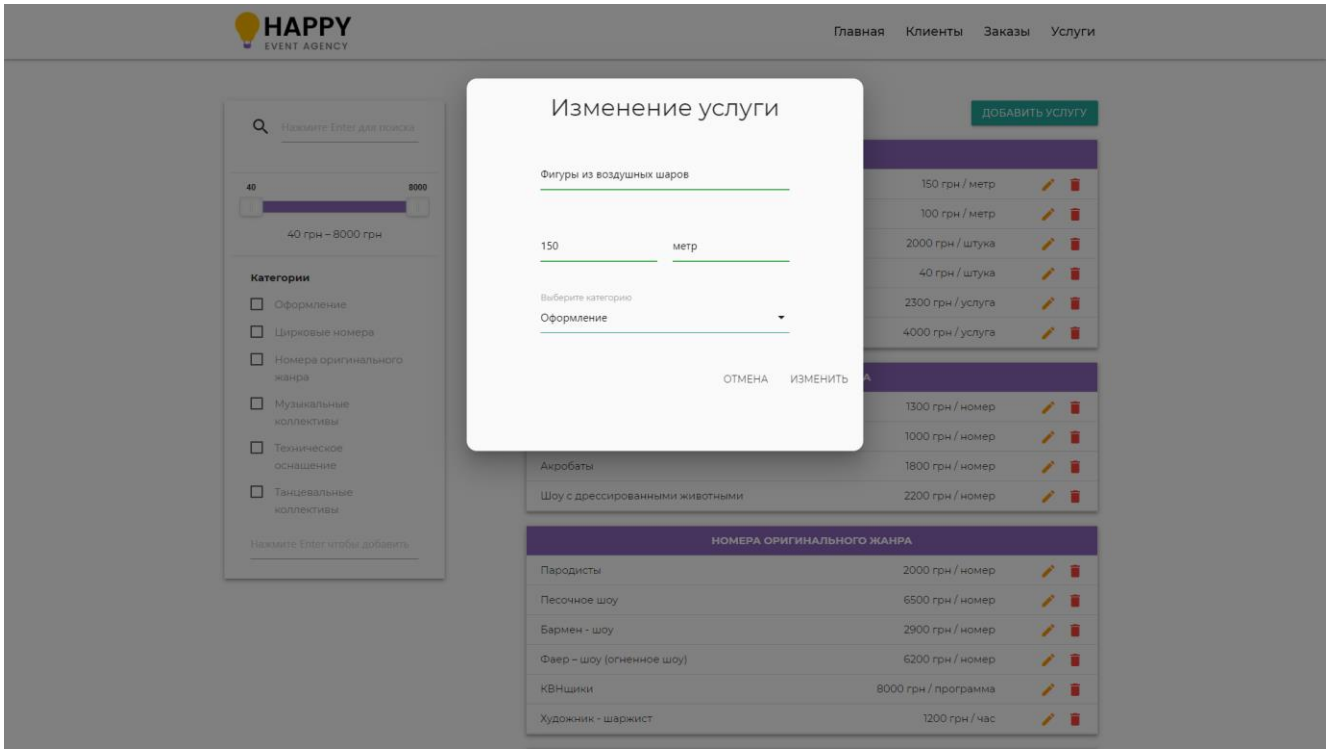

Рис. 2.20. Сторінка редагування послуги

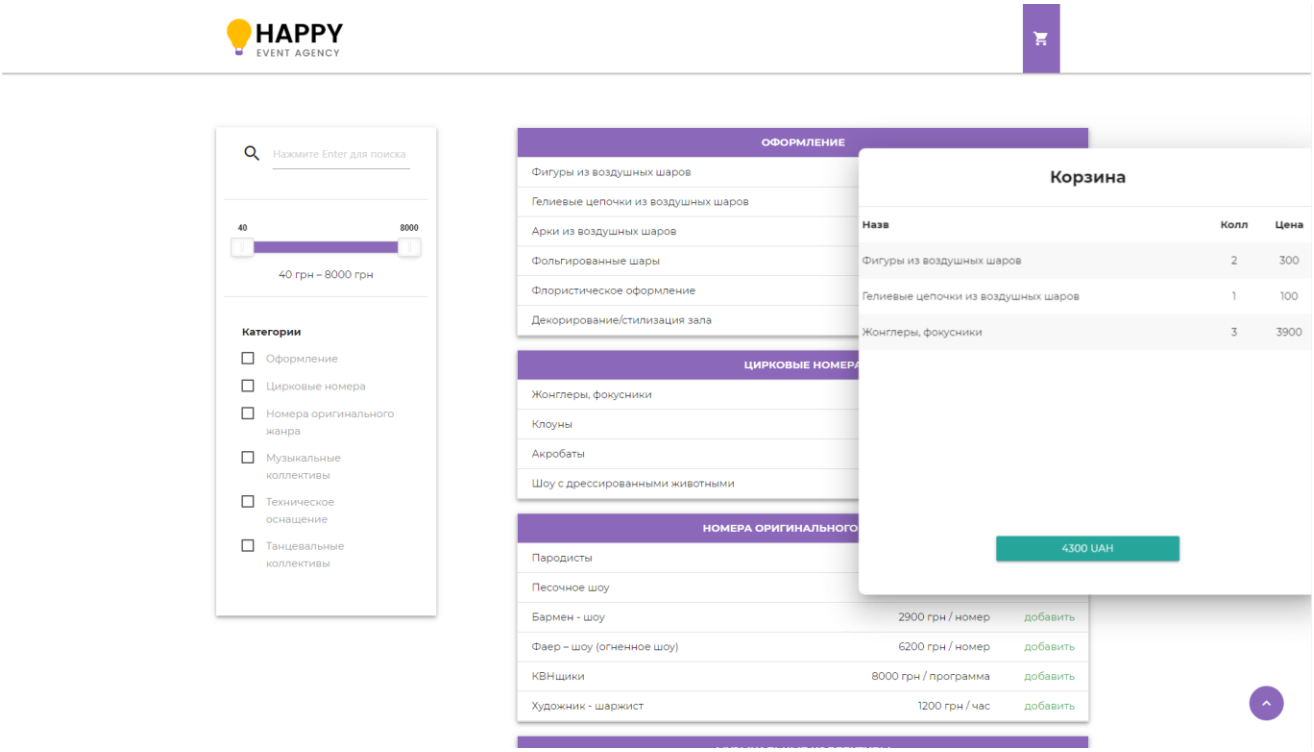

Рис. 2.21. Сторінка корзини

РОЗДІЛ 3
ЕКОНОМІКА

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1200;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста –165 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 14 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1200);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1200 \cdot (1 + 0{,}05) = 1680$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1680 \cdot 1{,}2) / (75 \cdot 1{,}2) = 22{,}4 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 1680 / (20 \cdot 1{,}2) = 70 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 1680 / (25 \cdot 1{,}2) = 56 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 1680 / (5 \cdot 1{,}2) = 280 \text{ чел.-ч.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^{k}_{отл} = 1{,}5 \cdot 280 = 420 \text{ людино-годин}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 1680 / (18 \cdot 1{,}2) = 77{,}78 \text{ людино-годин.}$$

$$t_{до} = 0{,}75 \cdot 77{,}78 = 58{,}33 \text{ людино-годин.}$$

$$t_{д} = 77{,}78 + 58{,}33 = 136{,}11 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 22{,}4 + 70 + 56 + 280 + 136{,}11 = 614{,}51 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 165 грн / год, отримуємо:

$$З_{ЗП} = 614{,}51 \cdot 165 = 39\,943{,}15 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (14 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 280 \cdot 14 = 3920 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 39943{,}15 + 3920 = 43\,863{,}15 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 614{,}51 / 1 \cdot 176 \approx 3{,}5 \text{ міс.}$$

Висновок. Програмне забезпечення призначене для організації роботи на підприємствах країни, які займаються організацією свят. Розроблене програмне забезпечення є мобільним додатком органайзером. Вартість даного програмного забезпечення становить майже 43,8 тис. грн. і не вимагає додаткових витрат як при розробці, так і при впровадженні програми. Очікуваний час розробки становить 3,5 місяців. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи було розроблено програмне забезпечення для «Студії організацій свят», яке було реалізовано на мові програмування JavaScript в середовище Visual Studio Code з системою управління базами даних MongoDB.

В ході виконання роботи було виконано наступні задачі:

- проведено аналіз предметної галузі. Зіставлено технічне завдання;
- побудована концептуальна модель бази даних;
- проведене логічне проектування бази даних;
- побудована та реалізована фізична модель бази даних;
- обґрунтовано вибір програмних засобів;
- спроектована та розроблена автоматизована програмна система.

Практичне значення системи полягає в розширенні впливу бізнесу на інтернет сегмент для збільшення клієнтської бази і відповідно доходів. Вона задовольняє усім сучасним вимогам, має високі показники якості та надійності, порівняно з аналогами.

Була проаналізована економічна ефективність розробки такої системи. Проведені розрахунки свідчать про економічну доцільність розробки та провадження нового програмного продукту. В подальшому, рекомендоване застосування цієї системи на підприємствах країни, які займаються організацією свят.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проектування баз даних - [Електронний ресурс]/IMF. – Режим доступу: https://ru.wikipedia.org/wiki/Проектирование_баз_данных

2. Логічне проектування - [Електронний ресурс]/IMF. – Режим доступу: http://shpora.me/Peter/12321

3. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2007. – 496 с.: ил. ISBN 5-94074-334-X.

4. О.М. Томашевський, Г.Г. Цетелик, М.Б. Вітер, В.І. Дубук. Інформаційні технології та моделювання бізнес-процесів [Текст]: навчальний посібник — К.: «Центр Учбової літератури», 2012. — 296 с. – ISBN 978-617-673-003-3.

5. Радченко, Г.И. Распределенные вычислительные системы / Г.И. Радченко. – Челябинск, 2012. – 184 с. ISBN 978-5-89879-198-8.

6. Раскин Д. Интерфейс: новые направления в проектировании компьютерных систем. — Пер. с англ. — СПб: Символ-Плюс, 2004. — 272 с., ил. ISBN 5-93286-030-8.

7. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. — 3-е изд. — М.: «Вильямс», 2003. — 1436 с. — ISBN 0-201-70857-4.

8. І.М. Бойчик, П.С. Харів, М.І. Хопчан, Ю.В. Піча. Економіка підприємства. Навчальний посібник для студентів економічних спеціальностей ВНЗ І-IV рівнів акредитації, друге видання, виправлене і доповнене – К., Каравела, Львів, Новий світ – 2000, 2001 рр. – 298 стор. ISBN 966-95-596-7-7.

9. Л.І. Шваб. Економіка підприємства, «Каравела», Навчальний посібник для студентів ВНЗ, 2004 р, 568 стор., ISBN 966-8019-2-6.

10. С.Ф. Покропивний. Економіка підприємства [Текст] : підручник для студ. екон. вузів і фак.: В 2. т. / ред. ; Київський економічний ун-т. - К. : Хвиля-Прес, 1995 . Т. 1. - [Б. м.] : [б.в.], 1995. - 393 с. – ISBN 5-87478-164-1.

11. Наказ Кабінету Міністрів України «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин» від 17 червня 1999 р.

12. М. Хавербеке – Выразительный JavaScript. Современное веб-программирование. - 2018. –С. 125-329.

13. Вартість аренди ноутбуку почасово. [Електронний ресурс] - Режим доступу: https://notebooksbu.com/garantiya-3-goda/

14. Середня заробітна плата Junior FE developer в Україні станом на початок 2021 року. https://dou.ua/lenta/articles/salary-report-devs-june-2020/

15. Шеннон Брэдшоу, Йон Брэзил, Кристина Ходоров. Mongo DB. Полное руководство. ДМК Пресс, 2020. – 540 с. -ISBN - 978-5-97060-792-3.

16. Кайл Бэнкер. MongoDB в действии. ДМК Пресс, 2016. -394 с. ISBN978-5-94074-831-1, 978-5-97060-431-1.

17. Прамодкумар Дж. Садаладж, Мартин Фаулер. NoSQL: новая методология разработки нереляционных баз данных. Диалектика-Вильямс. 2017. -192 с. ISBN978-5-8459-1920-5.

18. Марк Тиленс Томас. React в действии. Питер, 2018. -368 с. ISBN - 5446109996, 9785446109999.

19. Стоян Стефанов. React.js. Быстрый старт. Питер, 2017. Бестселлеры O'Reilly. -304 с. ISBN - 978-5-496-03003-8.

20. Кайл Симпсон. Замыкания и объекты. Питер, 2016, Вы не знаете JS. Бестселлеры O'Reilly. -270 с. ISBN - 978-5-4461-1255-5, 978-1449335588.

21. Дакетт Джон. HTML и CSS. Разработка и создание веб-сайтов. Эксмо, 2016. -480 с. ISBN - 978-5-699-64193-2.

ДОДАТОК А

КОД ПРОГРАМИ

Файл servicesRoute.js

```
const router = require("express").Router();
const Services = require("../models/Services.model");

router.get("/", async (req, res, next) => {
  try {
    let services = await Services.find();
    let prices = [];
    let price = {};
    let attrs = [];

    services.forEach((item) => prices.push(item.price));

    services.forEach((item) => {
      attrs.push(item.category);
    });

    attrs = attrs.filter((item, idx, arr) => arr.indexOf(item) === idx);

    price.allMin = Math.min(...prices);
    price.allMax = Math.max(...prices);
    prices = [];

    if (req.query.search) {
      const searchValue = req.query.search;
      const services = await Services.find({
        name: { $regex: searchValue, $options: "$i" },
      });

      services.forEach((item) => prices.push(item.price));

      if (prices.length) {
        price.min = Math.min(...prices);
        price.max = Math.max(...prices);
      }

      return res.json({ price, services, attrs });
    }
```

```
    if (req.query.category) {
      const names = req.query.category.split(",").join("|");
      const rg = new RegExp("(^)(" + names + ")($)");

      services = await Services.find({
        category: { $regex: rg, $options: "i" },
      });
    }

    if (req.query.min && req.query.max) {
      const min = req.query.min;
      const max = req.query.max;
      services = await Services.find({
        price: { $gte: min, $lte: max },
      });
    }

    services.forEach((item) => prices.push(item.price));

    if (prices.length) {
      price.min = Math.min(...prices);
      price.max = Math.max(...prices);
    }

    res.json({ price, services, attrs });
  } catch (err) {
    next(err);
  }
});

router.post("/", async (req, res, next) => {
  try {
    const { name, price, category, unit } = req.body;

    const service = new Services({
      name,
      price,
      category,
      unit,
    });
```

```
    await service.save();

    res.json(service);
  } catch (err) {
    next(err);
  }
});

router.put("/:id", async (req, res, next) => {
  try {
    const service = await Services.findByIdAndUpdate(
      req.params.id,
      { $set: req.body },
      { new: true }
    );
    res.json(service);
  } catch (err) {
    next(err);
  }
});

router.delete("/:id", async (req, res, next) => {
  try {
    const service = await Services.findByIdAndDelete(req.params.id);
    res.json(service);
  } catch (err) {
    next(err);
  }
});

module.exports = router;
```

Файл authRoute.js

```
const router = require("express").Router();
const { createNewUser, getToken } = require("../lib/token");
const { body, validationResult } = require("express-validator");

router.post(
  "/",
  [
    body("login")
      .exists({ checkNull: true, checkFalsy: true })
```

```
      .isLength({ min: 2, max: 16 })
      .isString(),
    body("password")
      .exists({ checkNull: true, checkFalsy: true })
      .isLength({ min: 5, max: 16 })
      .isString(),
  ],
  async (req, res, next) => {
    try {
      const errors = validationResult(req);

      if (!errors.isEmpty()) {
        return res.status(400).json({
          errors: errors.array(),
          message: "Логин или пароль неверен",
        });
      }

      const { userId, token } = await getToken(
        req.body.login,
        req.body.password
      );

      res.json({ userId, token });
    } catch (err) {
      next(err);
    }
  }
);

router.post("/new", async (req, res, next) => {
  const { login, password } = req.body;
  const newUser = await createNewUser(login, password);
  res.json({ user: newUser });
});

module.exports = router;
```

Файл app.js

```
const express = require("express");
const app = express();
```

```
const errorHandler = require("./src/middleware/errorHandler");

app.use(express.json());
app.use("/api/services", require("./src/routes/servicesRoute"));
app.use("/api/clients", require("./src/routes/clientsRoute"));
app.use("/api/auth", require("./src/routes/auth.route"));

app.use(errorHandler);

module.exports = app;
```

Файл token.js

```
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const config = require("config");
const Auth = require("../models/auth.model");
const createError = require("http-errors");

module.exports.getToken = async (login, password) => {
  const user = await findUserByLogin(login);
  await verificationPasswords(password, user.password);
  const payload = { userId: user._id };

  const token = jwt.sign(payload, config.get("secret"), {
    expiresIn: "1h",
  });

  return { userId: user._id, token };
};

const findUserByLogin = async (login) => {
  const user = await Auth.findOne({ login });

  if (!user) {
    throw new createError(400, `Пользователь с логином: ${login} не найден`);
  }

  return user;
};
```

```
const verificationPasswords = async (receivedPassword, userPassword) => {
  const isMatch = await bcrypt.compare(receivedPassword, userPassword);

  if (!isMatch) {
    throw new createError(400, `Пароль не верный`);
  }

  return isMatch;
};

module.exports.createNewUser = async (login, password) => {
  const hashPassword = await bcrypt.hash(password, 10);

  const newUser = await Auth.create({
    login: login,
    password: hashPassword,
  });

  return newUser;
};
```

Файл mongo.js

```
const mongoose = require("mongoose");
const config = require("config");

const CONNECTION_URL = config.get("mongoURL");

async function connectToDb() {
  await mongoose.connect(CONNECTION_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
    useFindAndModify: true,
  });
  console.log("Database connected...");
}

module.exports = connectToDb;
```

Файл errorHandler.js

```
const errorHandler = async (err, req, res, next) => {
  res.status(err.status || 500);
```

```
  res.json({
    status: err.status,
    message: err.message,
    stack: err.stack,
  });
};

module.exports = errorHandler;
```

Файл PriceListSection.js

```
import React from "react";
import PriceList from "../../Components/PriceList";

const PriceListSection = () => {
  return (
          <section id="price-list-section" className="price-list-section scrollspy">
      <div className="container">
        <h3 className="title">Услуги</h3>
        <PriceList />
      </div>
    </section>
  );
};

export default PriceListSection;
```

Файл PriceList.js

```
import React, { useState, useEffect } from "react";
import Filter from "./Filter/Filter";
import Services from "./Services";
import { useHttp } from "../hooks/http.hook";
import Loader from "../Components/Loader";

import FormWrapper from "../Components/FormWrapper";

const PriceList = (props) => {
  const [services, setServices] = useState([]);
  const [price, setPrice] = useState(null);
  const [attrs, setAttrs] = useState([]);

  const { request, loading } = useHttp();
  const { isAuth } = props;
```

```
useEffect(() => {
  const res = request("/api/services", "GET");
  res.then((data) => {
    getData(data);
  });
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, [request]);

const getData = (data) => {
  setPrice(data.price);
  setServices(data.services);
  setAttrs(data.attrs);
};

if (loading) {
  return <Loader />;
}

return (
  <div className="row">
    <div className="col s3 z-depth-2">
      <Filter
        setAttrs={setAttrs}
        price={price}
        getData={getData}
        isAuth={isAuth}
        attrs={attrs}
        services={services}
      />
    </div>
    <div className="col s8 offset-s1">
      {isAuth && (
        <div className="row">
          <div className="col right">
            <a
              class="waves-effect waves-light btn modal-trigger"
              href="#add-modal"
            >
              Добавить услугу
            </a>
          </div>
```

```
        </div>
       )}

       <Services
        services={services}
        isAuth={isAuth}
        attrs={attrs}
        setServices={setServices}
       />
      </div>
      <FormWrapper
       title="Добавление услуги"
       id="add-modal"
       attrs={attrs}
       services={services}
       setServices={setServices}
       isAdd={true}
       agreeBtn="Добавить"
       cancelBtn="Отмена"
      />
    </div>
  );
};

export default PriceList;
```

Файл Filter.js

```
import React, { useState, useEffect } from "react";
import NoUiSlider from "../NoUiSlider/NoUiSlider";
import Search from "../Search";
import Categories from "./Categories";

const Filter = ({ price, attrs, setAttrs, getData, isAuth, services }) => {
  const [form, setForm] = useState("");
  const [createdAttrs, setCreatedResponce] = useState([]);

  const onkeypress = (e) => {
    if (e.key === "Enter") {
      let isUnique = false;

      attrs.forEach((item) => {
                if (item.toLowerCase() !== form.toLowerCase()) return (isUnique = true);
```

```
    });

    if (isUnique) {
      setAttrs(() => [...attrs, form]);
      setCreatedResponce(() => [...createdAttrs, form]);
      setForm("");
    }
  }
};

return (
  <div className="filter">
    <div className="row filter__item">
      <div className="col s10 offset-s1">
        <Search getData={getData} />
      </div>
    </div>

    <div className="divider"></div>

    <div className="row filter__item">
      <div className="col s10 offset-s1">
        {price && (
          <NoUiSlider
            getData={getData}
            price={price}
            start={[price.min, price.max]}
            step={1}
            range={{ min: price.allMin, max: price.allMax }}
            connect
          />
        )}
      </div>
    </div>

    <div className="divider"></div>

    <div className="row filter__item">
      <div className="col s10 offset-s1">
        <div className="category">
          <span className="category__title">Категории</span>
          <Categories
```

```
            attrs={attrs}
            getData={getData}
            setAttrs={setAttrs}
            services={services}
            createdAttrs={createdAttrs}
          />

          {isAuth && (
            <input
              id="attr"
              type="text"
              name="attr"
              className="validate"
              placeholder="Нажмите Enter чтобы добавить"
              value={form}
              onChange={(e) => setForm(e.target.value)}
              onKeyPress={onkeypress}
            />
          )}
        </div>
      </div>
    </div>
  </div>
 );
};

export default Filter;
```

Файл Filter.js

```
import React, { useState } from "react";
import { useHttp } from "../hooks/http.hook";

const Search = ({ getData }) => {
  const [search, setSearch] = useState("");
  const { request } = useHttp();

  const keyPressHandler = (event) => {
    if (event.key === "Enter") {
      request(`/api/services?search=${search}`, "GET").then((data) =>
        getData(data)
      );
    }
```

```
  };

  return (
    <div className="input-field">
      <i className="material-icons prefix">search</i>
      <input
        id="search"
        type="text"
        className="validate"
        placeholder="Нажмите Enter для поиска"
        name="search"
        value={search}
        onChange={(e) => setSearch(e.target.value)}
        onKeyPress={keyPressHandler}
      />
    </div>
  );
};

export default Search;
```

Файл NoUiSlider.js

```
import React, { useState, useEffect } from "react";
import noUiSlider from "materialize-css/extras/noUiSlider/nouislider.min.js";
import { useHttp } from "../../hooks/http.hook";
import wNumb from "wnumb";
import "./nouislider.css";

const NoUiSlider = (props) => {
  const { request } = useHttp();

  const [prices, setPrices] = useState([]);

  useEffect(() => {
    setPrices([props.price.min, props.price.max]);
    const slider = document.getElementById("ctg-price-slider");
    noUiSlider.create(slider, {
      ...props,
      format: wNumb({
        decimals: 0,
      }),
    });
```

```
    if (!props.price) {
      slider.setAttribute("disabled", true);
    }

    slider.noUiSlider.on("update", onUpdate);
    slider.noUiSlider.on("end", onEnd);

    return () => {
      slider.noUiSlider.destroy();
    };

    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [props]);

  const onEnd = (val) => {
        request(`/api/services?min=${val[0]}&max=${val[1]}`, "GET").then((data) => {
      props.getData(data);
    });
  };

  const onUpdate = (val) => {
    setPrices([val[0], val[1]]);
  };

  return (
    <div className="price-slider center-align">
      <div id="ctg-price-slider"></div>
      <span className="price-slider__layout">
        {prices[0]} грн – {prices[1]} грн
      </span>
    </div>
  );
};

export default NoUiSlider;
```

Файл Categories.js

```
import React, { useState, useEffect } from "react";
import { useHttp } from "../../hooks/http.hook";
import Modal from "../Modal";
```

```
const Categories = ({ getData, attrs, services, createdAttrs, setAttrs }) => {
  const [names, setNames] = useState([]);
  const { request } = useHttp();
  const [currentId, setCurrentId] = useState(null);

  useEffect(() => {
        request(`/api/services?category=${names.join(",")}`, "GET").then((data)  =>
    getData(data)
  );

  // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [request, names]);

  useEffect(() => {
   var modals = document.querySelectorAll(".modal");
   window.M.Modal.init(modals);
   var selects = document.querySelectorAll("select");
   window.M.FormSelect.init(selects);
  }, []);

  const deleteHandler = (item) => {
   setAttrs(() => attrs.filter((attr) => attr !== item));
  };

  const clickHandler = (event) => {
   const { name } = event.target;

   if (names.includes(name)) {
    return setNames(() => names.filter((i) => i !== name));
   } else {
    setNames(() => [...names, name]);
   }
  };

  return (
   <ul>
    {attrs &&
     attrs.map((item, idx) => {
      return (
       <li key={idx} className="category__item">
        <p>
         <label>
```

```
              <input
                type="checkbox"
                name={item}
                onClick={clickHandler}
                disabled={createdAttrs.includes(item)}
              />
              <span>{item}</span>
              <a className="modal-trigger" href="#delete-modal">
                <i
                  className="material-icons red-text"
                  onClick={() => setCurrentId(item)}
                >
                  delete
                </i>
              </a>
            </label>
          </p>
        </li>
      );
    })}
    <Modal
      id="delete-modal"
      title="Вы действительно хотите удалить эту категорию?"
      subtitle="Все услуги из категории будут также удалены."
      agreeBtn="Удалить"
      cancelBtn="Отмена"
      agreeFunc={() => deleteHandler(currentId)}
    ></Modal>
  </ul>
 );
};

export default Categories;
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка програмного забезпечення для інформаційної підтримки надання послуг студії організації свят»
студентки групи 121-18ск-1 Калмикової Анастасії Володимирівни

Керівник економічного розділу
Зав. каф. ПЕП та ПУ, д.е.н. **О. Г. Вагонова**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом Калмикова.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Калмикова.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
diplom.zip	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація Калмикова.ppt	Презентація кваліфікаційної роботи.