Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему «РОЗРОБКА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГУРТКІВ, СЕКЦІЙ, КЛУБІВ»

Виконав студент 2 курсу, групи ЕК.м-01а
(номер курсу) (шифр групи)

Спеціальності 051 «Економіка»
(«Економічна кібернетика»)

Нечепоренко І.Д.
(прізвище, ініціали студента)

Керівник доцент, к.т.н, Яценко В.В.
(посада, науковий ступінь, прізвище, ініціали)

Суми – 2021 рік

РЕФЕРАТ

кваліфікаційної магістерської роботи на тему

«РОЗРОБКА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГУРТКІВ, СЕКЦІЙ, КЛУБІВ»

студента Нечепоренка Іллі Дмитровича

Актуальність теми, обраної для дослідження, визначається тим, що система гуртків, секцій, клубів позашкільної діяльності має велике значення у становленні підлітків у нашій країні. Тому розробка сучасної веборієнтованої інформаційної системи допоможе соціальному та освітньому розвитку учнів.

Мета кваліфікаційної роботи полягає у розробці веборієнтованої інформаційної системи гуртків, секцій, клубів.

Об'єктом дослідження є діяльність провідної IT-компанії «SoftServe».

Предметом дослідження є інформаційні системи та засоби розробки веборієнтованих інформаційних систем.

Задачі, які виконувались для досягнення мети роботи:

– дослідити суть поставленої задачі та предметної області;

– дослідити діяльність IT-підприємства, загальну характеристика організації;

– проаналізувати стан організаційної структури управління компанією;

– сформулювати основні вимоги до веборієнтованої інформаційної системи;

– описати основні моделі процесів інформаційної системи;

– розробити архітектуру інформаційної системи та вибрати технології вирішення поставлених завдань;

– дослідити структуру та особливості реалізації інформаційного забезпечення;

– розробити прототип додатку з урахуванням усіх обмежень та вимог накладених на нього;

– створити інструкцію з використання інформаційної системи;

– оцінити очікуваний ефект від впровадження інформаційної системи.

Для досягнення поставленої мети та задач дослідження були використані такі методи дослідження: аналітичний; експериментальний.

Інформаційною базою кваліфікаційної роботи є діяльність підприємства «SoftServe».

Основний науковий результат кваліфікаційної роботи полягає у створенні прототипу веборієнтованої інформаційної системи, що містить у собі всі необхідні функціональні модулі та ергономічний інтерфейс.

Одержані результати можуть бути використані користувачами із доступом до мережі інтернет.

Апробація результатів дослідження – опубліковані тези на Міжнародній науковій інтернет-конференції "Topical tendencies of science and practice" (10 грудня 2021 р., Едмонтон, Канада).

Ключові слова: веборієнтована інформаційна система, гурток, секція, клуб, база даних, портал, ASP.Net Core 3, C#, MySQL.

Зміст кваліфікаційної роботи викладено на 58 сторінках. Список використаних джерел із 70 найменувань, розміщений на 9 сторінках. Робота містить 24 рисунків, а також 3 додатків, розміщених на 10 сторінках.

Рік виконання кваліфікаційної роботи – 2021рік.

Рік захисту роботи – 2021 рік.

Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.е.н., професор
_____ О.В. Кузьменко
“__” _____ 2021 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ спеціальність 051 «Економіка (Економічна кібернетика)
студенту 2 курсу, групи ЕК.м-01а

Нечепоренко Іллі Дмитровичу
(прізвище, ім’я, по батькові студента)

1. Тема роботи Розробка веборієнтованої інформаційної системи гуртків, секцій, клубів
затверджена наказом по університету від «__»_____ 2021 року №_____
2. Термін подання студентом закінченої роботи «13» грудня 2021 року
3. Мета кваліфікаційної роботи розробка веборієнтованої інформаційної системи гуртків, секцій, клубів
4. Об’єкт дослідження діяльність провідної ІТ-компанії «SoftServe»
5.Предмет дослідження інформаційні системи та засоби розробки веборієнтованих інформаційних систем
6. Кваліфікаційна робота виконується на матеріалах «SoftServe»
7. Орієнтовний план кваліфікаційної роботи, терміни подання розділів керівникові та зміст завдань для виконання поставленої мети
Розділ 1 Дослідження організаційної структури управління компанією та формування вимог до інформаційної системи

15 листопада 2021 року
(назва – термін подання)

У розділі 1 дати загальну характеристику ІТ-компанії, проаналізувати стан організаційної структури управління, сформувати вимоги до веборієнтованої інформаційної системи
(зміст конкретних завдань до розділу, які має виконати студент)

Розділ 2 Проектування веборієнтованої інформаційної системи

22 листопада 2021 року
(назва – термін подання)

У розділі 2 зробити опис моделі процесів інформаційної системи, представити архітектуру веборієнтованої інформаційної системи, розглянути основні технології створення та склад функціональної частини web-додатку

(зміст конкретних завдань до розділу, які повинен виконати студент)

Розділ 3 Реалізація прототипу веборієнтованої інформаційної системи

29 листопада 2021 року
(назва – термін подання)

У розділі 3 описати структуру та особливості реалізації інформаційного забезпечення, створити інструкцію щодо використання, оцінити очікуваний ефект від впровадження інформаційної системи

(зміст конкретних завдань до розділу, які повинен виконати студент)

8. Консультації з роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

9. Дата видачі завдання: «18» жовтня 2021 року

Керівник кваліфікаційної роботи _____ В.В. Яценко
(підпис) (ініціали, прізвище)

Завдання до виконання одержав _____ І.Д. Нечепоренко
(підпис) (ініціали, прізвище)

ЗМІСТ

ВСТУП

Важливий і прогресивний напрям в системі освіти, спрямований на якісне досягнення високих результатів у сфері основної освітньої програми. На відміну від навчальної діяльності, позакласна робота спрямована, перш за все, на формування інтересу учня до різних видів діяльності та поширення індивідуальних здібностей та обдарованості учнів. Позакласні гуртки, секції, клуби допоможуть знайти своє соціальне місце та розкрити наявні таланти, а також покращити здоров'я та підвищити загальний рівень культурного розвитку юнаків [1].

В Україні зареєстровано 1 тис. 351 заклад позашкільної освіти, де діють 7,5 тис. гуртків, в яких навчаються 1 млн 138 тис. дітей та працюють 33,4 тис. педагогічних працівників. Позашкільні заклади підготували 150 майстрів спорту міжнародного класу, 725 майстрів спорту України, понад 3 тис. кандидатів у майстри спорту України та понад 10 тис. спортсменів-розрядників [2].

Актуальність теми, обраної для дослідження, визначається тим, що система гуртків, секцій, клубів позашкільної діяльності має велике значення у становленні підлітків у нашій країні. Тому розробка сучасної веборієнтовної інформаційної системи допоможе соціальному та освітньому розвитку учнів.

Мета кваліфікаційної роботи полягає у розробці веборієнтованої інформаційної системи гуртків, секцій, клубів.

Об'єктом дослідження є діяльність провідної IT-компанії «SoftServe».

Предметом дослідження є інформаційні системи та засоби розробки веборієнтованих інформаційних систем.

Задачі, які виконувались для досягнення мети роботи:

– дослідження суті поставленої задачі та предметної області;

– дослідження діяльності IT-підприємства, загальна характеристика організації;

– аналіз стану організаційної структури управління компанією;

– формулювання основних вимоги до веборієнтованої інформаційної системи;

– опис основних моделей процесів інформаційної системи;

– розробка архітектури інформаційної системи та вибір технологій вирішення поставлених завдань;

– дослідження структури та особливостей реалізації інформаційного забезпечення;

– розробка прототипу додатку з урахуванням усіх обмежень та вимог накладених на нього;

– створення інструкції з використання інформаційної системи;

– оцінка очікуваного ефекту від впровадження інформаційної системи.

Для досягнення поставленої мети та задач дослідження були використані такі методи дослідження: аналітичний, експериментальний.

1 ДОСЛІДЖЕННЯ ОРГАНІЗАЦІЙНОЇ СТРУКТУРИ УПРАВЛІННЯ КОМПАНІЄЮ ТА ФОРМУВАНЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Характеристика об’єкта дослідження

Місце практики магістратури – ІТ-компанія «SoftServe». В дослідженні використано матеріали бази практики для написання кваліфікаційної роботи. SoftServe – провідна ІТ-компанія, що пропонує консалтинг та послуги у галузі цифрових технологій [3].

Перший офіс софтверної компанії SoftServe було відкрито у Львові 1993 року [4]. Головний офіс компанії знаходиться у Львові та Остіні (Техас, США), а в європейських та американських офісах компанії працює понад 10 000 співробітників.

SoftServe у Центральній та Східній Європі є однією з найбільших компаній з розробки програмного забезпечення та є у трійці найбільших аутсорсингових компаній в Україні. SoftServe надає супровід компаніям на шляху до інновацій – від розробки та впровадження технологічних послуг і продуктів, до створення сучасних привабливих ідей. SoftServe надає допомогу компаніям у ідентифікації точок диференціації на ринку, допомагають клієнтам ефективно конкурувати, допомагають прискорити розробку інноваційних цифрових рішень, що створюють додаткову перевагу для бізнесу. Компанія займається трансформацією, прискоренням та оптимізацією способу ведення крупного бізнесу та технологічних компаній. Маючи за плечима досвід у сфері охорони здоров’я, фінансових послуг, розробки програмного забезпечення, роздрібної торгівлі, медіа, SoftServe створює комплексні цифрові рішення, які ефективно допомагають клієнтам мати кращі бізнеспоказники та впроваджувати інновації [5].

SoftServe відіграв важливу роль для створення Microsoft Bird's Eye у 2004 році. Пізніше Google використовував таку саму концепцію розробки Google Street

View. За свій внесок у розвиток проекту SoftServe його запросили виступити на щорічній конференції Microsoft, де компанія показала приклад створення комерційних додатків технологічними корпораціями.

SoftServe має великий досвід у розробці програмного забезпечення, від хмарних обчислень, безпеки та проектування інтерфейсу користувача до Big Analytics та Інтернету речей. Основні галузі, для яких компанія розробляє програмні рішення - охорона здоров'я, роздрібна торгівля, фінансові послуги та програмне забезпечення.

У 2012 році SoftServe запустив щорічний конкурс Ukraine IT Awards, який визначає найкращих професіоналів IT-індустрії України. Компанія і досі залишається конкурентоспроможним партнером.

«SoftServe» є прем’єр-партнером Google Cloud, та офіційним торговим представником Google Cloud у Ірландії та Великобританії.

У статусі прем’єр-партнера SoftServe підтримує суцільну імплементацію та управління Google Cloud, від початку розробки додатків до контролювання витрат, надаючи супровід клієнтам з Великобританії та Ірландії за допомогою ландшафту управління хмарними ресурсами. У подальшому SoftServe планує розширення роботи як офіційного торгівельного представника Google Cloud у інших регіонах.. Також SoftServe спеціалізується на Cloud Migration, IoT, Data Analytics, Infrastructure, Machine Learning, та Security. Компанія успішно завершила вже більше 150 проектів у цьому напрямку, маючи доступ до широкого інструментарію та бази знань Google Cloud, який складає більше 430 ресурсів. У SoftServe працює понад 330 сертифікованих експертів, а тільки у 2020 році близько 100 розробників SoftServe приєдналися до сертифікованих експертів у Google Cloud [6].

Будучи однією з найбільших технологічних компаній у регіоні, SoftServe активно займається участю у позитивній трансформації IT-освіти в школах та університетах, саме для того, щоб відповідати сучасним технологічним тенденціям і бути конкурентоспроможними на світовому ринку. SoftServe послідовно працює над вирішенням цієї проблеми на національному, місцевому

та корпоративному рівнях [7].

1.2 Аналіз організаційної структури управління компанією

Одна з основних концепцій менеджменту – організаційна структура управління. Ця концепція пов'язана з процесом управління, роботою менеджерів цілями, функціями та поділом повноважень між ними. У рамках цієї структури є потоки інформації та прийняття уряду.

Складання документів, в яке втручаються менеджери всіх рівнів, категорій та професійних спеціалізацій. Структура – це своєрідний каркас побудови адміністративної системи, побудований для своєчасного та ефективного виконання цих процесів. Отже, керівництво IT-компаній особливу увагу приділяє принципам і методам побудови методів управління, вивчаючи тенденції зміни, зміни. Тип управління – це впорядкований набір взаємозалежних елементів організації, які забезпечують їхнє функціонування та розвиток в цілому.

У групі управління організації є лінійні та функціональні зв'язки. Лінійні зв’язки пов'язані з прийняттям та реалізацією адміністративних рішень, інформацією між лінійними керівниками, тобто людьми, які несуть за це повну відповідальність. Функціональні – пов'язані з певними специфічними функціями управління. Отже, формуються повноваження: лінійний персонал, заводський персонал та функціональний персонал. Повноваження лінійних начальників дають право записувати всі питання розвитку довірених їм організацій та підрозділів, а також давати попередження та накази. Повноваження персоналу обмежуються правом планувати, рекомендувати, падати чи допомагати, але з наставляти інших членів організації.

Популярною структурою управління великою IT-компанією – є дивізіональна організаційна структура. Створюється у випадках, коли підприємство зростає, ускладнюються технологічні процеси, відбувається диверсифікація підрозділів. Тобто навколо певного виробництва формується організаційний підрозділ з автономією у здійсненні своєї повсякденної

операційної діяльності (рис 1.1).

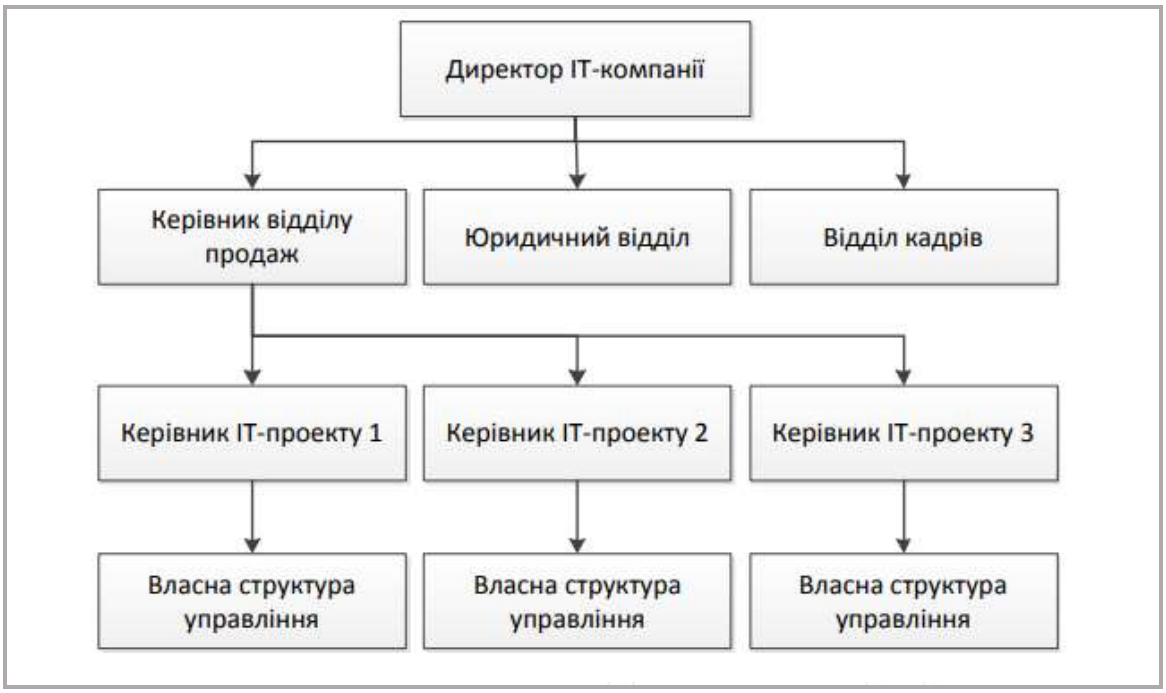

Рисунок 1.1 – Схема дивізіональної структури підприємства

Переваги дивізіональної оргструктури:

– забезпечення управління багатопрофільним підприємством із загальною кількістю працівників до 100 тисяч і територіально віддаленими підрозділами;

– без втручання в оперативну діяльність виробничих підрозділів;

– можливість вищому керівництву зосередитись на вирішенні стратегічних проблем;

– підвищення якості рішень (завдяки наближенню до місця виникнення проблеми);

– гнучкість до змін у зовнішньому середовищі;

– міцний взаємозв‘язок виробництва зі службовцями;

– конкуренція всередині фірми.

Недоліки дивізіональної оргструктури:

– повторення функцій управління на рівні підрозділів;

– розбіжність інтересів дивізіонів і центру;

– складність контролю з верхівки за порушеннями на місцях;

– підвищення витрат на утримання апарату управління [8].

Приклад використання центрів компетенцій компанії на одному з проектів:

– бізнес-аналітики;

– розробники;

– керівники проектів (PM);

– контроль якості;

Методологія: Scrum-ban, як засіб для отримання максимальної відповідності вимогам власників процесу.

SCRUM-Ban Task and Project Management – це рішення, що дозволяє уникнути детального планування і зосередитися на частому і раптовому появі термінових завдань. Але головне, за допомогою SCRUM-Ban можна візуалізувати та оптимізувати ланцюжок технологічного розвитку та користувальницькі історії. Причиною появи цього «гібрида» стало розчарування команд, які впровадили SCRUM, той факт, що ця методологія не враховує наступних характеристик:

– незаплановані завдання витісняють заплановані спринти та знецінюють ваші цілі. У цьому випадку клієнту потрібно все відразу: термінові завдання, і розвиток за планом. Але SCRUM спрямовано розвиток невеликими порціями за затвердженим сценарієм;

– складні чи недостатньо опрацьовані завдання вимагають надмірної кількості командного часу;

– якщо команда погано збалансована з погляду проектних ролей, деякі з її членів не діють, інші перевантажені [9].

1.3 Формування вимог до веборієнтованої інформаційної системи

Багато організацій працюють з великими обсягами даних. Дані є основними цінностями або фактами і організовані в базу даних. Багато людей вважають дані синонімом інформації; однак інформація насправді складається з даних, які були організовані, щоб допомогти відповісти на запитання та вирішити проблеми.

Інформаційна система визначається як програмне забезпечення, яке може допомогти організувати та аналізувати дані. Отже, мета інформаційної системи – перетворити вихідні дані в корисну інформацію, яку використовують для прийняття рішень в організації.

Хоча інформаційні системи можуть відрізнятися за тим, як вони використовуються в організації, вони зазвичай містять такі компоненти:

– апаратне забезпечення: комп'ютерні інформаційні системи використовують комп'ютерне обладнання, таке як процесори, монітори, клавіатура та принтери;

– програмне забезпечення: це програми, які використовуються для організації, обробки та аналізу даних;

– бази даних: інформаційні системи працюють з даними, організованими в таблиці та файли;

– мережа: різні елементи повинні бути підключені один до одного, особливо якщо багато різних людей в організації використовують одну інформаційну систему;

– процедури: вони описують, як конкретні дані обробляються та аналізуються, щоб отримати відповіді, для яких розроблена інформаційна система [10].

До аналізу вимог відноситься визначення потреб і умов, котрі висуваються для нового, або зміненого товару, беручи до уваги потенційно суперечливі вимоги клієнтів, користувачів чи бенефіціарів. Вирішальне значення для успішної розробки проекту має аналіз вимог.

Деякі вимоги є функціональними, а деякі - нефункціональними за своєю природою, вони також можуть бути відокремленими в той же час, що деякі вимоги можуть бути класифіковані як технологічно незалежні, а інші - технологічні. Таким чином, це зумовлює необхідність класифікації, яка дозволяє організаціям думати про різні аспекти вимог. FURPS – це методика перевірки пріоритетних вимог після розуміння потреб клієнтів. Абревіатура FURPS – це функціональність, зручність використання, надійність, продуктивність і

сумісність протягом певного періоду часу, і виникла велика потреба побачити рішення з більшої кількості вимірів, яке було надано для появи FURPS+. Ця техніка FURPS+ створила класифікацію вимог, щоб підкреслити розуміння різних типів нефункціональних вимог:

– функціональність: F в абревіатурі FURPS + представляє основні характеристики продукту, які знайомі в комерційній сфері розроблюваного рішення. Функціональні вимоги, які також можуть враховувати архітектурно значущі загальносистемні функціональні вимоги, можуть включати аудит, ліцензування, локалізацію, пошту, онлайн-довідку, друк, звітність, безпеку, адміністрування системи або робочий процес;

– зручність використання: зручність використання включає спостереження, фіксацію та встановлення вимог на основі проблем користувальницького інтерфейсу, таких як доступність, естетика інтерфейсу та узгодженість в інтерфейсі користувача;

– надійність: надійність включає такі речі, як доступність, точність і можливість відновлення, наприклад, обчислення або здатність системи відновлюватися після збою в роботі;

– продуктивність: продуктивність включає такі речі, як пропускна здатність інформації через систему, час відповіді системи (який також пов'язаний з зручністю використання), час відновлення та час запуску;

– сумісність: нарешті, ми прагнемо включити розділ під назвою сумісність, де ми визначаємо ряд інших вимог, таких як тестованість, адаптивність, ремонтопридатність, сумісність, конфігураційність, можливість встановлення, масштабованість, локалізація тощо;

– знак «+» в абревіатурі FURPS + дозволяє нам вказати обмеження, включаючи дизайн, реалізацію, інтерфейс та фізичні обмеження;

– обмеження дизайну: обмеження дизайну, як випливає з назви, обмежує дизайн; наприклад, вимога до реляційної бази даних визначає підхід, який ми використовуємо для розробки системи;

– обмеження реалізації: обмеження реалізації накладає обмеження на

кодування чи конструкцію: стандарти, платформу чи мову реалізації;

– обмеження інтерфейсу: обмеження інтерфейсу є вимогою для взаємодії із зовнішнім елементом. Коли він розробляється всередині компанії, йому часто доводиться взаємодіяти із зовнішніми системами;

– фізичні обмеження: фізичні обмеження впливають на апаратне забезпечення, яке використовується для розміщення системи, наприклад форму, розмір і вагу [11, 12].

У випадку застосування до цієї класифікації поділ вимог саме на функціональні та нефункціональні, вони повинні включати всі перераховані вище групи, крім першої, тобто URPS +. Додаткова специфікація містить інші вимоги, інформацію, та обмеження, які нелегко врахувати у випадках використання або глосарію, включаючи загальносистемні атрибути або вимоги якості «URPS +». Вимоги специфічні для випадку використання можуть (і, ймовірно, повинні) бути написані спочатку з використанням у розділі Особливі вимоги, але деякі вважають за краще також об’єднати всі в додатковій специфікації [13].

Специфікація може включати:

– FURPS + вимоги: функціональність, зручність використання, надійність, продуктивність, і сумісність;

– звіти

– апаратні та програмні обмеження (операційні та мережеві системи);

– обмеження розробки (наприклад, інструменти або процеси розробки);

– інші обмеження щодо дизайну та реалізації;

– проблеми інтернаціоналізації;

– документація (користувач, установка, адміністрування) та допомога;

– ліцензії та інші юридичні питання;

– упаковка;

– стандарт (технічність, безпека, якість);

– проблеми з фізичним середовищем (наприклад, тепло або вібрація);

– експлуатаційні проблеми (наприклад, як обробляються помилки або як часто;

– правила домену або комерційні правила;

– інформація про сфери інтересів ;

– обробка кредитних платежів;

Надійність включає в себе такі характеристики системи, як:

– відмови;

– допустима частота / частота відмов;

– середній час відмов і їх серйозність;

– відновлення системи після збоїв, а також можливість попереднього резервного копіювання даних;

– прогнозованість поведінки;

– час роботи системи, час доступності системи або режим роботи (наприклад, «системі треба бути доступною 24 години на добу, 7 днів на тиждень»);

– точність розрахунків.

Продуктивність. Продуктивність системи складається з наступних характеристик:

– час відгуку системи, швидкість;

– ефективність ;

– пропускна здатність, включаючи загальну і допустиму кількість одночасних користувачів, кількість призначених для користувача запитів, кількість звернень системи до бази даних і кількість запитаних / переданих даних за одиницю часу;

– час, необхідний для відновлення - швидкість відновлення;

– час, необхідний для запуску та вимикання - швидкість запуску і виключення;

– витрата ресурсів.

Обслуговування системи. Підтримка включає в себе можливості:

– тестування;

– розширення - збільшення додаткової функціональності системи;

– масштабування - тиражування, наприклад, в філіях / підрозділах організації;

– адаптація / адаптація до використання в даному середовищі, в тому числі шляхом попередньої настройки;

– сумісність;

– технічне обслуговування, супровід: оновлення даних та виправлення помило, періодичність архівування та резервного копіювання;

– сервіс і ремонт;

– встановлення;

– локалізація;

– портативність;

– відповідність міжнародним стандартам [14-15].

Вхідною інформацією для інформаційної системи буде база даних MySQL, яка містить повну інформацію про гуртки, секції, клуби, батьків, їх дітей, надавачів послуг тощо.

Основними користувачами вебдодатку батьки, які шукають гуртки, клуби, секції для своїх дітей та надавачі послуг, які розміщують свою гуртки на порталі.

Основною функціональною частиною інформаційної системи буде можливість пошуку, фільтрування, додавання, видалення, зміна гуртків, секцій, клубів, а також можливість реєстрації батьків, надавачів послуг, організацій. Для цього потрібно створити таблиці, які будуть відображати актуальну інформацію, беручі всю необхідну інформацію із бази даних.

2 ПРОЕКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Моделі процесів інформаційної системи

Модель – об’єкт-замінник вихідного об’єкта, що забезпечує дослідження деяких властивостей; це спрощене представлення системи для аналізу і прогнозування, та отримання кількісних і якісних результатів, які можуть бути необхідними, щоб прийняти відповідне управлінське рішення.

Під час розв’язання певної задачі, коли є потреба виявлення певної властивості об’єкта дослідження , модель є не тільки корисною, а іноді тільки одним інструментом дослідження. Один і той же об’єкт може мати багато моделей, і одна модель може описувати різні об’єкти.

Єдиної класифікації типів моделей немає із-за багатозначності поняття «модель» у науці. Це можна зробити з кількох причин: за характером об'єктів і моделей; за сферами застосування тощо [16].

Для проектування ІП використовуються інформаційні моделі, що представляють об’єкти та процеси у вигляді креслень, схем, таблиць, формул, текстів тощо.

Інформаційна модель – модель об’єкта, явища або процесу, яка представляє інформаційні аспекти змодельованого об’єкта, процесу чи явища. Вона є основою для розробки моделей інтелектуальної власності.

Природні мови зазвичай використовуються для створення описових моделей текстової інформації. Поряд з природними формальними мовами розробляються і використовуються: системи числення, алгебра відносин, мови програмування і т. д. Основна відмінність формальних і природних мов полягає в тому, що формальні мови мають не тільки жорстко закріплений алфавіт, а й також суворі правила граматики та синтаксису.

Під час вивчення нового об’єкта, будують спочатку описову модель цього об’єкту, потім формалізують, тобто документують використовуючи математичні

формули та об’єкти геометрії.

Модель має враховувати якомога більше факторів. Однак цю ситуацію важко реалізувати, особливо в погано структурованих системах. Тому вони часто намагаються створювати досить прості моделі елементів, враховуючи їх мікро- та макрозв’язки. Це дозволяє отримати адекватні результати [17].

Моделювання процесів – це техніка, яка передбачає створення візуального опису процесу. Зазвичай це досягається за допомогою використання інструментів моделювання процесів, таких як блок-схеми та універсальні позначення процесу бізнес-моделювання (також відомі як BPMN) [18].

Існує багато різних інструментів моделювання процесів, які можна використовувати для покращення робочих процесів, роблячи їх більш ефективними та прибутковими:

– діаграма SIPOC – це інструмент, який використовується в методології Six Sigma. SIPOC – допомагає зацікавленим сторонам визначити ключові елементи проекту вдосконалення процесу. Елементами є постачальники, вхідні ресурси, процес, який вдосконалюється, продукти та клієнти, які отримують продукцію;

– діаграми BPMN – це інструменти моделювання бізнес-процесів, розроблені Business Process Management Initiative (BPMI). Тому при створенні моделі процесу ви використовуєте елементи, зазначені в методології BPMN;

– діаграми уніфікованої мови моделювання (UML). UML – це мова моделювання розробки, яка використовується для забезпечення стандартизованого способу візуального представлення системи. Діаграми включають акторів, дії, ролі та класи системи та допомагають краще зрозуміти або задокументувати систему. UML було створено в 1994 році, і його швидке зростання популярності призвело до того, що в 2005 році він був опублікований як затверджений стандарт ISO [19].

Для чіткого розуміння процесів та для уникнення непорозумінь при розробці веборієнтованої інформаційної системи для гуртків, секцій, клубів, було прийнято рішення побудувати BPMN діаграму процесів (рис 2.1).

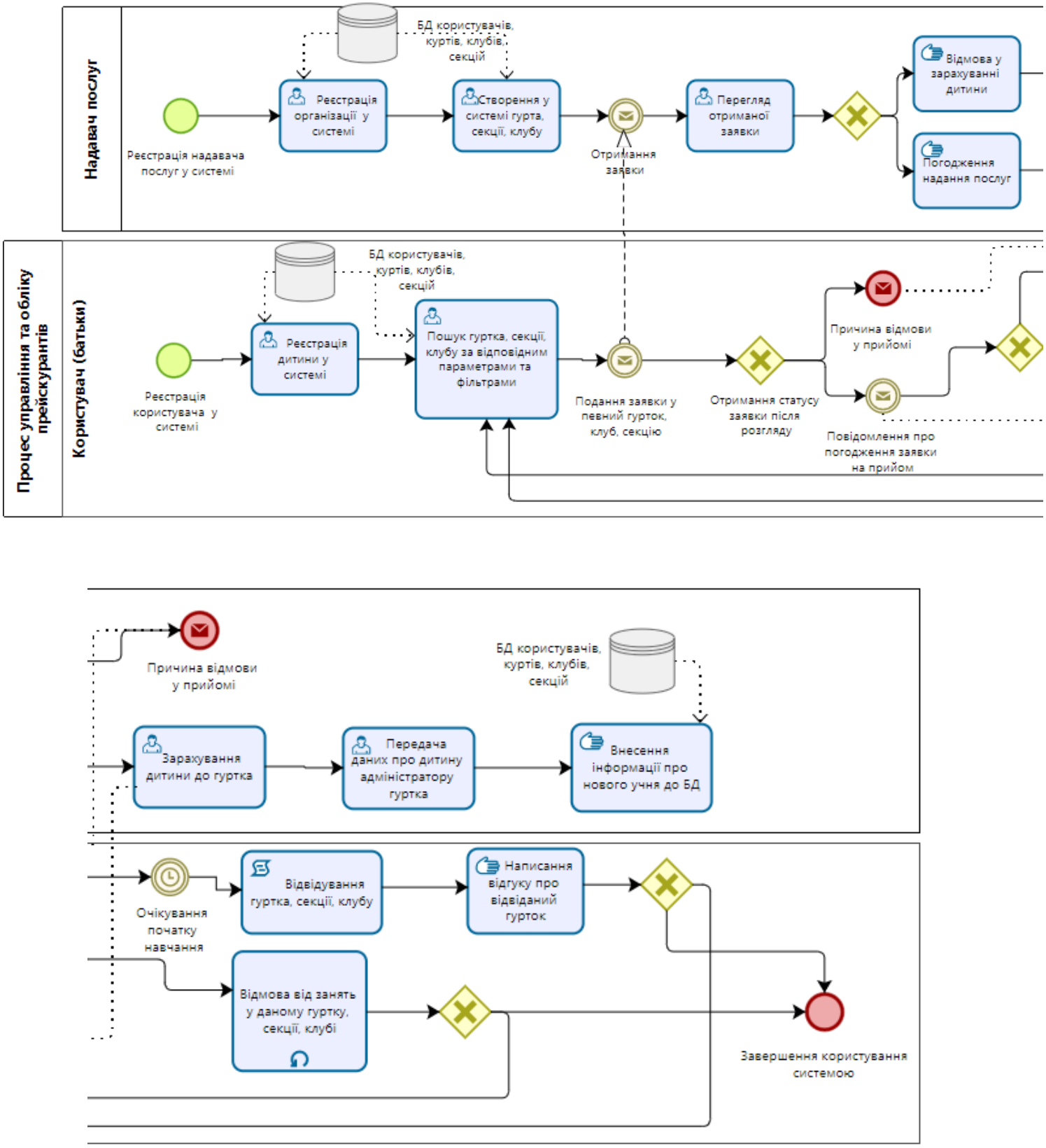

Рисунок 2.1 – BPMN діаграма процесів інформаційної системи

2.2. Архітектура інформаційної системи та технології вирішення поставлених завдань

Архітектура містить детальний опис поточного дизайну, змісту, переліку апаратних засобів, програмного забезпечення та мережевих можливостей комп’ютеризованої системи. Ця архітектура включає як апаратне, так і програмне забезпечення, які використовуються для надання рішення клієнту. Він також містить детальну інформацію про довгострокові плани, такі як оновлення та/або заміна старого обладнання та програмного забезпечення [20, 21].

Архітектура – це опис дизайну та змісту комп'ютеризованої системи. Якщо задокументовано, архітектура може містити таку інформацію, як детальний опис поточного обладнання, програмного забезпечення та мережевих можливостей; опис довгострокових планів і пріоритетів майбутніх закупівель, а також план оновлення та/або заміни застарілого обладнання та програмного забезпечення [22]. Архітектура повинна задокументувати:

– які дані зберігаються?

– як функціонує система?

– де розташовані компоненти?

– коли в системі відбуваються дії та події?

– і чому система існує?

Для розробки веборієнтованої інформаційної системи було обрану клієнт-серверну архітектуру.

Архітектура клієнт/сервер – це обчислювальна модель, в якій кілька компонентів працюють у строго визначених ролях для спілкування. Клієнт споживає ресурси і послуги, які сервер доставляє та розміщує. Такий тип архітектури спільних ресурсів має де-кілька, або один клієнтський комп’ютер, під’єднаний до центрального сервера через мережу або інтернет-з’єднання (рис. 2.2).

Архітектура клієнт/сервер стала відомою як модель мережевих обчислень,

тому що всі послуги та запити доставляються за допомогою мережі. Це вважається формою розподіленої обчислювальної системи, оскільки компоненти виконують свою роботу незалежно один від одного [23, 24].

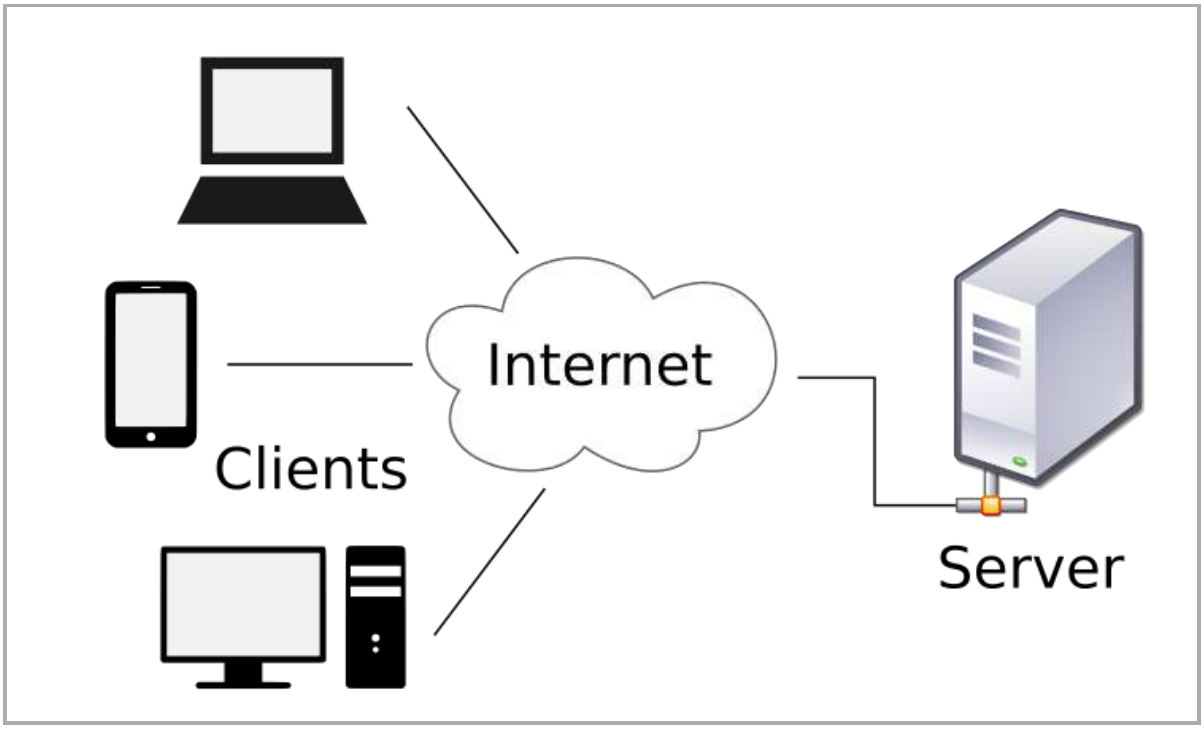

Рисунок 2.2 – Схема клієнт-серверної архітектури

Клієнт – це графічна складова, в якій перший рівень формується, по суті, програмний продукт для користувача. На першому рівні можна просуватися і виконувати рутинно: алгоритми шифрування, перевірка введених значень на відповідність і дійсність формату, примітивні операції (групування, сортування, підрахунок значень) з даними, інтерфейс користувача. На першому рівні не обов'язково мати реальні зв'язки з базою даних [25].

Сервер розташований на другому рівні. На цьому рівні знаходиться більша частина бізнес-логіки. Поза рівнем є фрагменти, які експортуються до терміналів, а також зберігаються в збережених процедурах і тригерах третього рівня.

Сервер баз даних забезпечує зберігання даних і виводиться на третій рівень. Частіше за все може бути стандартна реляційна або об'єктно-орієнтована БД. Якщо третій рівень сприймається, як база даних із збереженими тригерами, процедурами та схемою, яка пояснює програму з точки зору реляційної моделі, то

другий рівень є програмним інтерфейсом, який з’єднує клієнтські компоненти з прикладною логікою бази даних [26].

Найбільш популярними СУБД для веборієнтованих інформаційних систем є Oracle, MySQL, MS SQL, PostgreSQL, MongoDB, Firestore [27].

Розробка структури навігації – одна з найважливіших складових, що впливають на створення сайтів. Цей блок містить все, що пов'язане із наповненням сайту, а також з інформаційною стратегією проекту. Важливо, щоб користувачі могли швидко і легко знаходити потрібну інформацію. Для цього відображається можливий режим користувачів на сторінках сайту та розробляється структурна схема, яка має бути інтуїтивно зрозумілою для кожного відвідувача.

Професійно розроблена структура дозволяє значно знизити витрати на збирання інформаційної складової сайту та уникнути великої кількості помилок на етапі виробництва. Такий сайт працює набагато ефективніше та не потребує постійних коригувань у процесі своєї роботи [28, 29].

Створення оптимальної структури – дуже важлива складова створення вебсайту. Якщо користувач отримує доступ до сторінки, перевантаженої інформацією, швидше за все, він просто закриє її. Сторінка повинна бути створена таким чином, щоб можна було одразу визначити її зміст та знайти необхідну інформацію або одне з посилань [30].

На рисунку 2.3 відображена мапа створюваної веборієнтованої інформаційної системи. Зі схеми можна побачити, що вебсайт складається з багаторівневої ієрархії сторінок, тому можна зазначити, що рівень вкладеності є оптимальним, адже дає змогу представити структуроване інформаційне навантаження, без створення складнощів для користувача в навігації, а це є одною із основних вимог до користувальницьких інтерфейсів інформаційних систем.

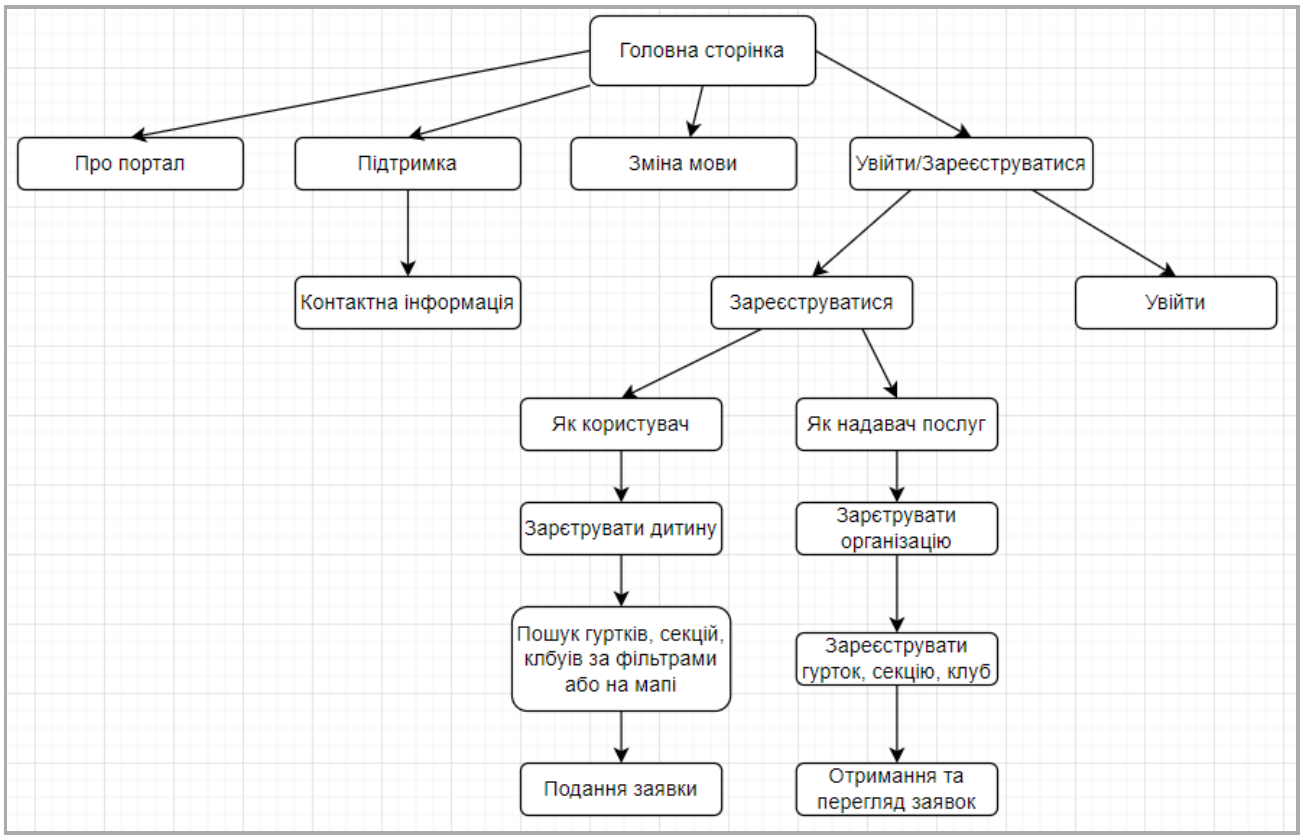

Рисунок 2.3 – Структурна схема веборієнтованої системи гуртків, секцій, клубів

Після вивчення можливостей створення веборієнтованих інформаційних систем та аналізу аналогічних організацій, сформувалося бачення, який саме вигляд повинна мати інформаційна система, яка виконує всі умови створення сучасних систем.

Після відкриття головної сторінки сайту користувач може не тільки застосувати всі пошукові інструменти до бази даних гуртків, секцій, клубів, але й дізнатися більше про вебресурс, звернутися за необхідністю до служби підтримки, змінити мову сайту на англійську або українську, зареєструватися для створення власного кабінету користувача системи.

Наступним кроком, спираючись на структурну схему, було створено прототип головної сторінки за допомогою Figma (рис 2.4).

Figma – це вебдодаток для редагування графіки та дизайну інтерфейсу користувача. Ви можете використовувати його для виконання будь-яких видів

графічного дизайну від вебсайтів, розробки інтерфейсів мобільних додатків, створення прототипів, створення публікацій у соціальних мережах і всього, що між ними.

Figma відрізняється від інших інструментів для редагування графіки. В основному тому, що він працює безпосередньо у вашому браузері. Це означає, що користувач отримує доступ до своїх проектів і починаєте проектування з будь-якого комп'ютера або платформи без необхідності купувати кілька ліцензій або встановлювати програмне забезпечення [31, 32].

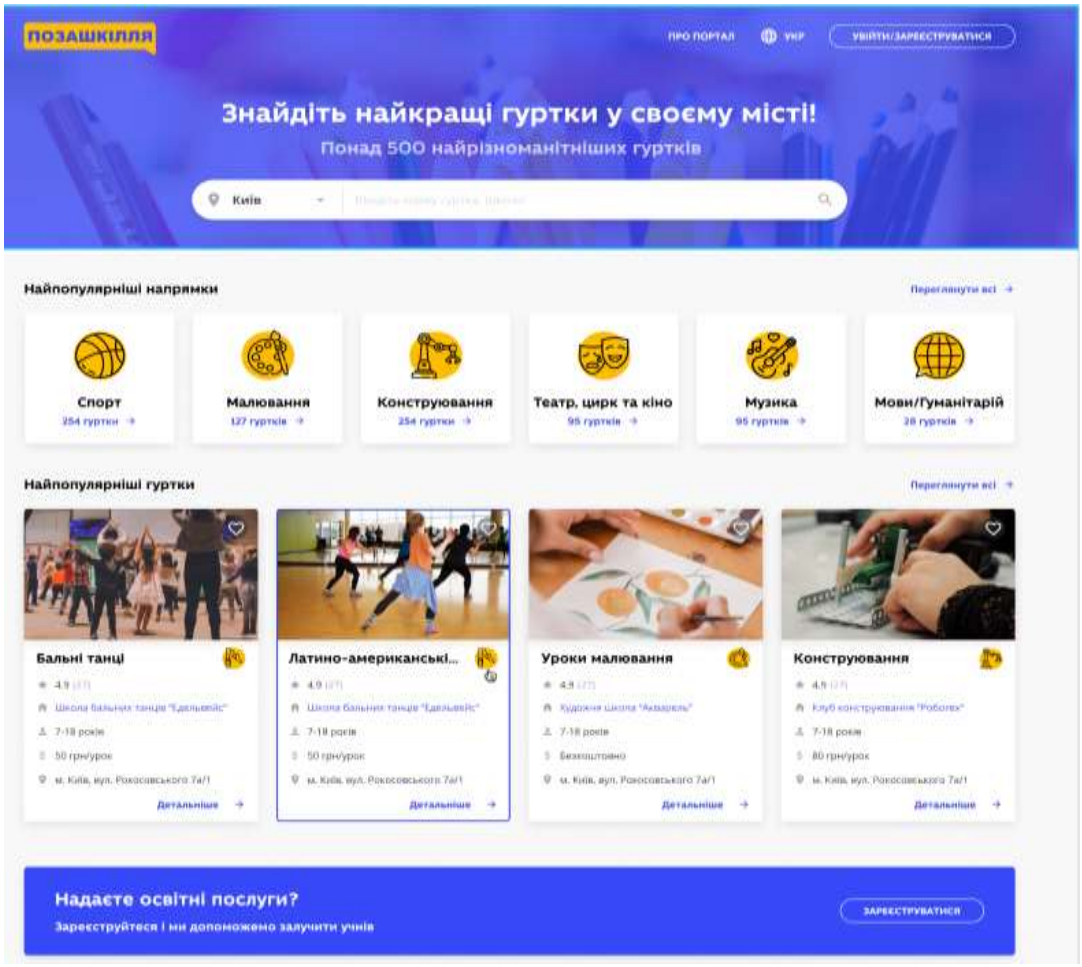

Рисунок 2.4 – Макет головної сторінки у Figma

Аналогічно розроблено макети інших сторінок порталу (див. додаток Б). Перша сторінка, яку бачить користувач, повинна давати чітке уявлення про призначення даного порталу, формувати певні асоціації та створювати гарний імідж, який відрізняє інформаційну систему від інших.

Для зв’язку БД та головною сторінкою у браузері служать HTTP запити. HTTP визначає різноманітні методи запиту, які вказують, якою буде бажана дія для даного ресурсу. Хоча їх імена можуть бути іменниками, ці методи запиту іноді називають дієсловами HTTP.

Методи запиту HTTP:

– GET: запит на представлення ресурсу. Запити, що використовують цей метод, можуть лише отримувати дані. У нього немає тіла.

– POST: використовується для відправки об’єктів на певний ресурс. Це часто викликає зміну стану або якийсь побічний ефект на сервері. Має тіло.

– PUT: замінити всі поточні представлення ресурсів даними запиту.

– DELETE: видаляє вказаний ресурс [33].

На головній сторінці ми можемо побачити найпопулярніші напрямки та найпопулярніші гуртки, які повертаються за допомогою GET запитів.

Далі, для пошуку гуртків, секцій, клубів, користувачу потрібно натиснути на «Переглянути всі» гуртки. З’явиться можливість знайти саме той гурток, який потрібен користувачу за допомогою фільтрації. Фільтрація та пошук реалізований за допомогою функціоналу Elasticsearch.

Elasticsearch (ES) – це масштабована утиліта повнотекстового пошуку та аналізу, яка дозволяє швидко зберігати, шукати та аналізувати великі обсяги даних у режимі реального часу. ES є ядром стеку ELK (Elastic Stack), який, крім Elasticsearch

Elasticsearch – це представник NoSQL, який входить у комплект JSON REST API.

Ми можемо вважати це нереляційним сховищем документів JSON і повнотекстовою пошуковою системою Lucene. Апаратна платформа - віртуальна машина Java.

Офіційні клієнти доступні на Java, NET (C #), Python, Groovy, JavaScript, PHP, Perl, Ruby [34, 35].

У лістингу 2.1 представлений фрагмент фільтрації гуртків, клубів, секцій за

ціною.

```
if (filter.IsFree && (filter.MinPrice == 0 && filter.MaxPrice == int.MaxValue))
        {
           queryContainer &= new TermQuery()
           {
              Field = Infer.Field<WorkshopES>(w => w.Price),
              Value = 0,
           };
        }
        else if (!filter.IsFree && !(filter.MinPrice == 0 && filter.MaxPrice == int.MaxValue))
        {
           queryContainer &= new NumericRangeQuery()
           {
              Field = Infer.Field<WorkshopES>(w => w.Price),
              GreaterThanOrEqualTo = filter.MinPrice,
              LessThanOrEqualTo = filter.MaxPrice,
           };
        }
        else if (filter.IsFree && !(filter.MinPrice == 0 && filter.MaxPrice == int.MaxValue))
        {
           var tempQuery = new QueryContainer();

           tempQuery = new NumericRangeQuery()
           {
              Field = Infer.Field<WorkshopES>(w => w.Price),
              GreaterThanOrEqualTo = filter.MinPrice,
              LessThanOrEqualTo = filter.MaxPrice,
           };
           tempQuery |= new TermQuery()
           {
              Field = Infer.Field<WorkshopES>(w => w.Price),
              Value = 0,
           };
           queryContainer &= tempQuery;
        }

        if (filter.MinAge != 0 || filter.MaxAge != 100)
        {
           var ageQuery = new QueryContainer();

           ageQuery = new NumericRangeQuery()
           {
              Field = Infer.Field<WorkshopES>(w => w.MinAge),
              LessThanOrEqualTo = filter.MaxAge,
           };
           ageQuery &= new NumericRangeQuery()
           {
              Field = Infer.Field<WorkshopES>(w => w.MaxAge),
              GreaterThanOrEqualTo = filter.MinAge,
           };
           queryContainer &= ageQuery;
        }
```

Лістинг 2.1 – Фрагмент фільтрації гуртків, клубів, секцій за ціною

Для подальшого користування порталом, користувачу пропонується увійти до особистого кабінету, або зареєструватися як батьки/надавач послуг. Авторизація та аутентифікація у формі виконується за допомогою функціоналу IdentityServer. Також на порталі існує адміністраторський доступ та певні дозволи

на використання функцій. Технічний адмін відрізнятиметься від звичайного користувача правом видаляти або блокувати надавача послуг та підтверджувати його реєстрацію. Адміністратор гуртка буде слідкувати за учнями певного гуртка та зможе переглядати заяви, приймати та відхиляти їх, позначати завершення навчання учнем, створювати та редагувати гурток та матиме можливість блокувати здобувачів послуг.

IdentityServer – сервер автентифікації, який реалізує стандарти openid connect Підключення (OIDC) і OAuth 2,0 для ASP.NET Core. Він призначений для надання загального способу автентифікації запитів до всіх програм незалежно від того, чи є вони вебдодатками, власними, мобільними або інтерфейсними кінцевими точками API. Ця технологія використовується щоб реалізувати єдиний Sign-On (єдиного входу) для кількох програм та типів програм. Його можна використовувати для перевірки справжності реальних користувачів за допомогою форм входу та аналогічних інтерфейсів користувача, а також для перевірки автентичності на основі служб, яка зазвичай включає видачу, перевірку та оновлення маркерів без будь-якого користувальницького інтерфейсу. IdentityServer розроблена як рішення, що налаштовується. Кожен екземпляр зазвичай налаштовується відповідно до потреб конкретної організації та/або набору додатків [36, 37].

Основними програмними засобами, які використані для розробки веборієнтованої інформаційної системи: Microsoft Visual Studio 2019, MySQL, GitHub, Figma, IdentityServer, ElasticSearch та були використані такі технології та мови програмування як: C#, середовище ASP .Net Core 3, WebApi.

Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft.

За допомогою Visual Studio ви можете розробити:

– класичні програми для комп’ютера під управлінням операційної системи Windows;

– мобільні програми (Windows, iOS, Android);

– вебдодатки;

– хмарні програми;

– ігри;

– бази даних SQL Server і SQL Azure.

У Visual Studio можна використовувати такі технології та мови програмування: .NET, Node.js, C, C#, C++, Python, Visual Basic, F#, JavaScript.

Community – це безкоштовна версія середовища розробки Visual Studio. Щоб використовувати його, необхідно створити обліковий запис Visual Studio; в іншому випадку він буде дійсний протягом 30 днів. Видання має менше функціональних можливостей порівняно з платними версіями, але включає все необхідне для створення повних додатків. Підходить для індивідуальних розробників і навчання;

Professional – видання містить професійні інструменти для розробки додатків. Функціональність у цій версії ще не завершена, наприклад, вона відображається з точки зору інструментів діагностики, налагодження та тестування. Підходить для невеликих команд розробників;

Enterprise – це повнофункціональна версія Visual Studio. Повне рішення для розробки додатків. Підходить для груп будь-якого розміру з високими вимогами до якості та масштабу [38-39].

MySQL — це система управління реляційною базою даних (RDBMS) з відкритим вихідним кодом від Oracle, вбудована на мові структурованих запитів (SQL). MySQL може працювати практично на будь-якій платформі, включаючи Linux, Windows і UNIX. Хоча його можна використовувати в широкому діапазоні, MySQL найчастіше асоціюється з веб-додатками. MySQL є важливим компонентом корпоративного стека з відкритим кодом під назвою LAMP. LAMP — це платформа веб-розробки, яка використовує Linux як операційну систему, Apache як веб-сервер, MySQL як систему керування реляційною базою даних і PHP як об’єктно-орієнтовану мову сценаріїв [40, 41].

Спочатку задуманий шведською компанією MySQL AB, MySQL був придбаний компанією Sun Microsystems у 2008 році, а потім Oracle, коли вона

купила Sun у 2010 році. Розробники можуть використовувати MySQL за Загальною публічною ліцензією GNU (GPL), але компанії повинні отримати комерційну ліцензію від Оракул. Сьогодні MySQL є СУБД, що стоїть за багатьма провідними веб-сайтами у світі та незліченною кількістю споживчих і бізнес-веб-додатків, включаючи Facebook, Twitter та YouTube.

ASP.NET Core — це високопродуктивне кросплатформне середовище з відкритим кодом для створення сучасних хмарних додатків, підключених до Інтернету. ASP.NET Core дозволяє виконувати такі завдання:

– створювати вебпрограми та служби, програми Інтернету речей (IoT) та серверні частини для мобільних додатків;

– використовувати вибрані засоби розробки у Windows, macOS та Linux;

– розгортання в хмарі або локальному середовищі;

– запускати у .NET Core [43].

ASP.NET Core має низку архітектурних змін, які призводять до більш компактної та модульної структури. ASP.NET Core більше не ґрунтується на файлі System.Web.dll. Він базується на наборі детальних і добре структурованих пакетів NuGet. Це дозволяє оптимізувати програму за допомогою пакетів NuGet, які вам необхідні. Переваги меншої площі поверхні програми включають: більш суворий захист, знижений рівень обслуговування, покращену продуктивність та зниження витрат у моделі [44, 45].

Web API є способом побудови програми ASP.NET, який спеціально заточений для роботи в стилі REST (Representation State Transfer або "передача стану подання") (рис. 2.5). REST-архітектура передбачає застосування таких методів або типів запитів HTTP для взаємодії із сервером:

– GET;

– POST;

– PUT;

– DELETE.

GET використовується для отримання ресурсів, POST – для додавання

нових ресурсів, PUT – для оновлення ресурсів, а DELETE – для видалення ресурсів [46].

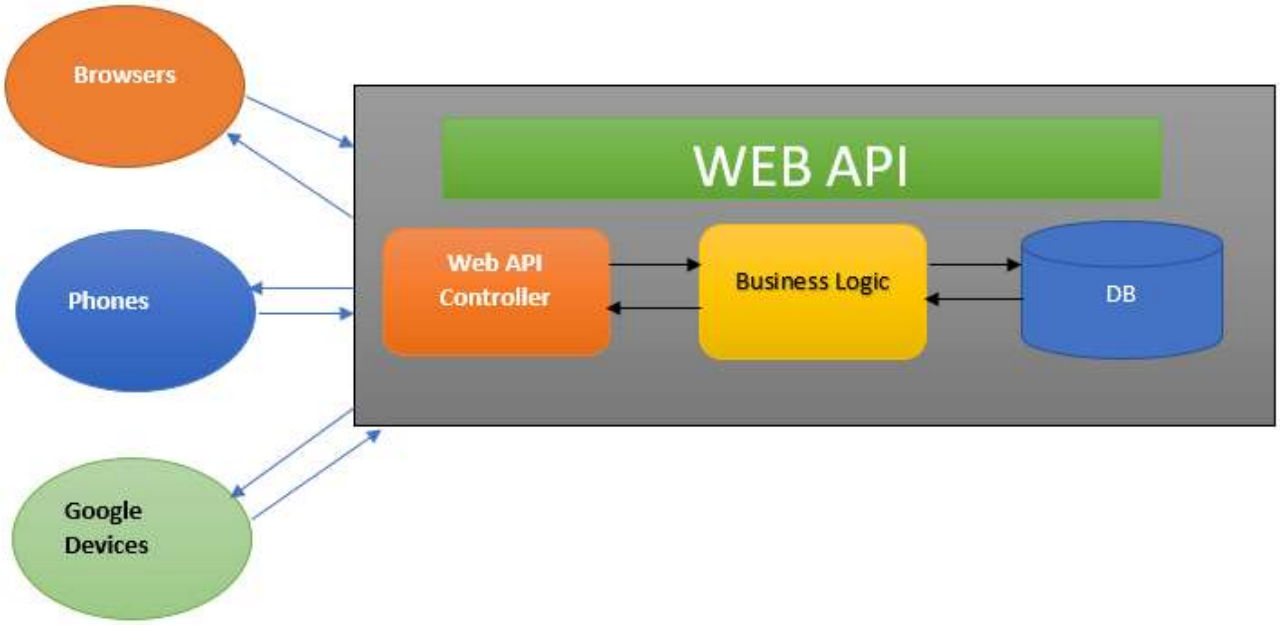

Рисунок 2.5 – Діаграма архітектури Web API

3 РЕАЛІЗАЦІЯ ПРОТОТИПУ ВЕБОРЇЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Структура та особливості реалізації інформаційного забезпечення

Однією із основних задач побудови веборієнтованої інформаційної системи є реалізація програмного забезпечення. Організація інформаційного забезпечення інформаційних систем є надзвичайно важливою, оскільки процес управління неможливий без трансформації інформаційних потоків. Система інформаційного забезпечення інформаційних систем повинна бути спроектованою з урахуванням різних факторів і що відповідає за створення єдиного інформаційного фонду, розробку засобів формалізованого опису даних, систематизацію та уніфікацію показників та документів. Із завданнями, загальною структурою системи, складом функцій управління, формами подання даних та засобами перетворення інформації. Для функціональних підсистем ІС носій інформації - це сукупність інформаційних ресурсів та найважливіший елемент інформаційної системи, оскільки він є допоміжною підсистемою для вирішення завдань та наповнює їх конкретним змістом [47, 48].

Інформаційна підтримка інформаційної системи призначена для підвищення якості управління за допомогою отримання ймовірних, підтверджених і своєчасних даних.

Для ефективної роботи та зберігання даних було прийнято рішення використовувати реляційну БД MySQL. Фрагмент діаграми зв’язків відображений на малюнку 3.1. Реляційна база даних – це тип бази даних, яка зберігає та надає доступ до точок даних, пов’язаних одна з одною. Реляційні бази даних засновані на реляційній моделі, інтуїтивно зрозумілому, простому способі представлення даних у таблицях. У реляційній БД під унікальним ідентифікатором, який називається ключем, мають на увазі кожен рядок таблиці де цей ідентифікатор не повторюється. Стовпці таблиці мають атрибути даних, та кожен запис, внесений до таблиці, зазвичай має значення для кожного атрибута, що полегшує

встановлення зв’язків між точками даних [49].

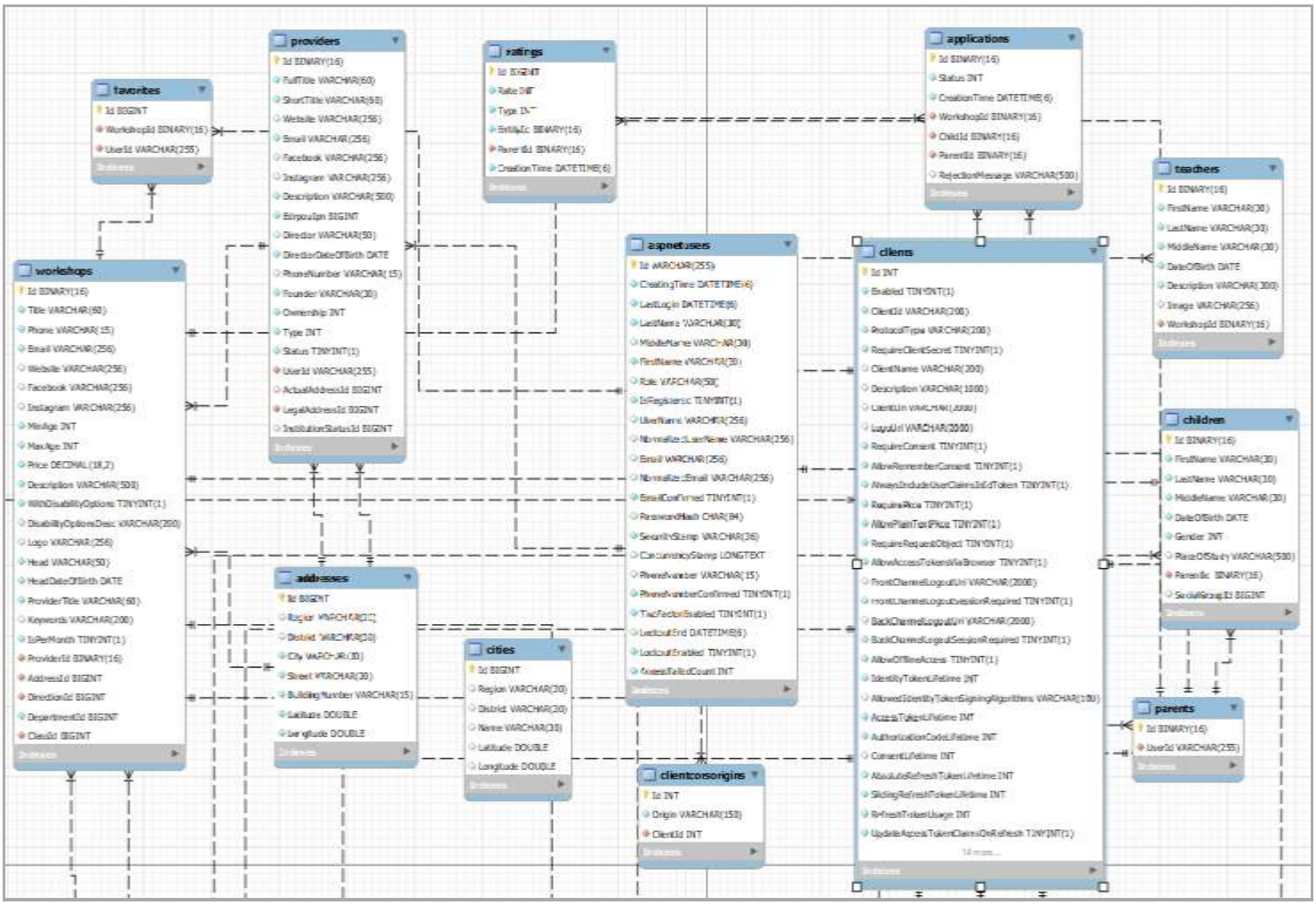

Рисунок 3.1 – Фрагмент діаграми зв’зків таблиць БД

При створенні зв'язків між сутностями необхідно створити первинний і зовнішній ключі, які використовуються для встановлення зв'язків.

Щоб побудувати реляційну базу даних веборієнтованої інформаційної системи гуртків, секцій, клубів, необхідно створити базові таблиці для сутностей, де кожен стовпець відповідає певному атрибуту сутності.

Переглянемо таблиці створені у MySQL. У створеній БД є декілька основотворчих таблиць, зокрема: «workshops», «providers», «applications», «aspnetusers». У більшості таблиць ідентифікатором є поле Id, з типом даних GUID (Binary). GUID означає глобальний унікальний ідентифікатор. GUID – це 128-розрядне ціле число (16 байт), яке можна використовувати на всіх комп’ютерах і в мережах, де потрібен унікальний ідентифікатор [50].

У таблиці «workshops» зберігаються дані, які містять інформацію про гуртки, секції, клуби, а також зв’язок із таблицями «directions», «providers», «departments», «classes» за допомогою зовнішніх ключів (рис 3.2). Дана таблиця має такі поля:

– Id (ідентифікатор);

– Title (назва гуртка, секції, клубу);

– Phone (контакний номер телефону);

– Email (поштова скринька);

– Website (вебсайт);

– Facebook;

– Instagram;

– MinAge (мінімальний допустимий вік дитини);

– MaxAge (максимальний допустимий вік дитини);

– Price (ціна);

– Description (опис);

– WithDisabilityOptions (чи має умови для дітей із інвалідністю);

– DisabilityOptionsDesc (опис умов для дітей із інвалідністю);

– Logo;

– Head (власник гуртка, клубу, секції);

– HeadDateOfBirth (дата народження власника);

– ProviderTitle (надавач послуг);

– Keywords (ключові слова для пошуку);

– IsPerMonth (чи ціна за місяць);

– ProviderId (id надавача послуг);

– DirectionId (id напрямку);

– DepartmentId (id відділу);

– ClassId (id класу).

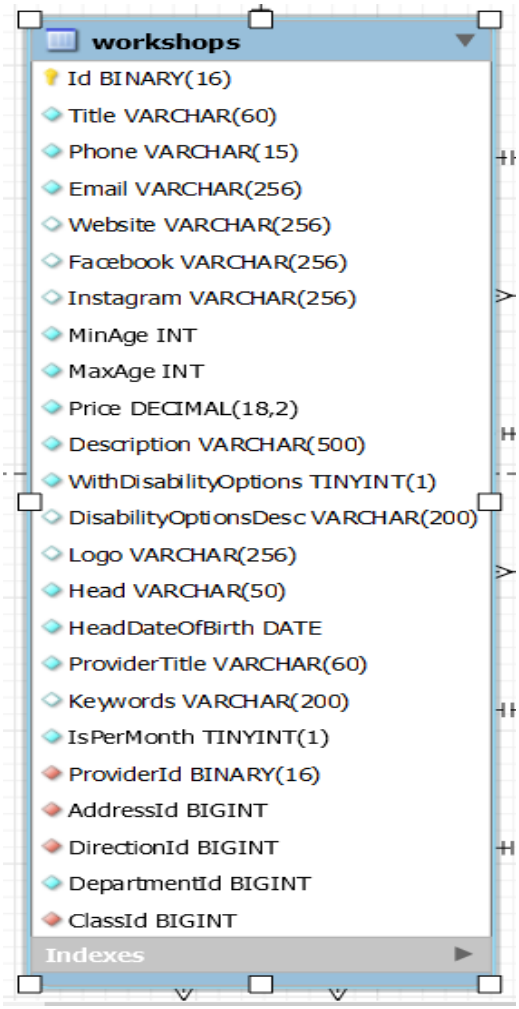

Рисунок 3.2 – Структура таблиці «workshops»

У таблиці «providers» зберігається інформація про надавачів послуг, а саме: повна та коротка назва, вебсайт, email, соціальні мережі, опис, код ЄДРПОУ, інформація про директора, інформація про засновника, тип власності, статус, та зв’язки з іншими таблицями за допомогою зовнішніх ключів (рис 3.3).

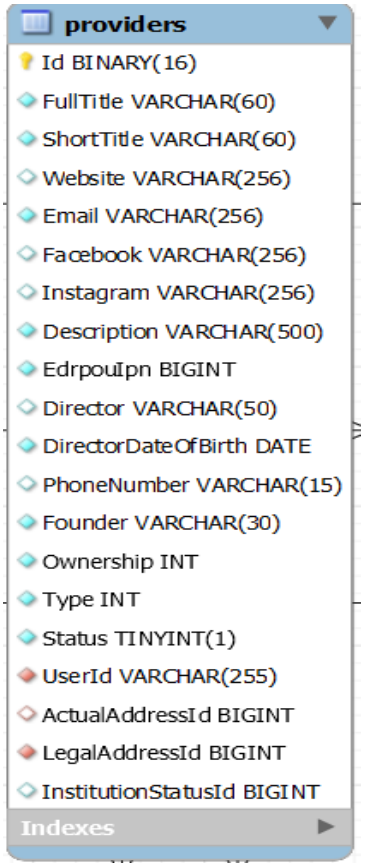

Рисунок 3.3 – Структура таблиці «providers»

У таблиці «applications» зберігаються дані про статус, дату та час створення application, ідентифікатор воркшопу, ідентифікатор дитини та батьків, повідомлення відмови (рис. 3.4).

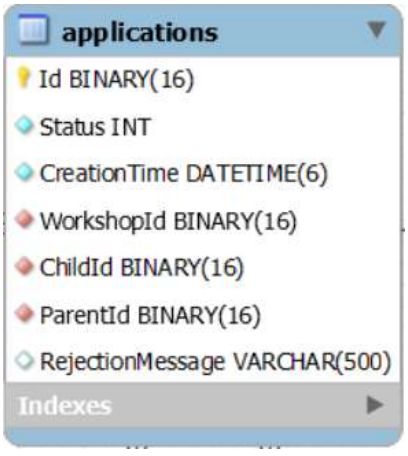

Рисунок 3.4 – Структура таблиці «applications»

Також розглянемо таблицю «aspnetusers», у якій зберігається інформація про: дату та час створення, останній логін, ПІБ, роль, ім’я юзера, нормалізоване

ім’я юзера, email, нормалізований email, хеш паролю, перевірка надійності, номер телефону, двофакторна аутентифікація, перелік невірних спроб входу (рис 3.5).

Рисунок 3.5 – Структура таблиці «aspnetusers»

3.2 Інструкція щодо використання інформаційної системи

Для успішного впровадження інформаційної системи гуртків, секцій, клубів, потрібно підготувати інструкцію з її використання та підтримки.

У горі відображено хедер із логотипом та навігаційні кнопки (рис 3.6). Щоб отримати детальну інформацію про портал треба натиснути кнопку «Про портал». Для звернення до підтримки, при виникненні питань, треба натиснути кнопку «Підтримка». Також можна змінити мову сайту натиснувши на іконку глобусу. Для повноцінного використання функцій порталу здобувачу/надавачу послуг пропонується увійти або зареєструватися у особистому кабінеті. Також за заздалегідь створеними акаунтами увійти можуть технічні адміністратори та адміністратори гуртків.

Нижче від кнопок надається можливість пошуку гуртків за назвою, попередньо обравши місто в Україні.

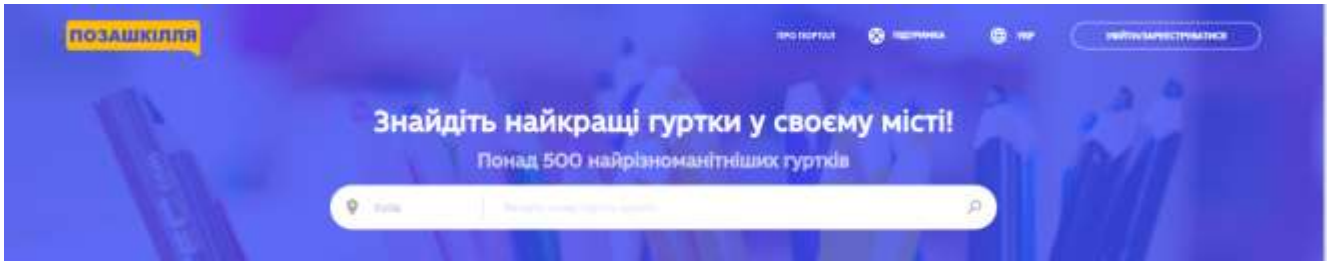

Рисунок 3.6 – Хедер сторінки

Під хедером розташовані найпопулярніші напрямки та найпопулярніші гуртки (рис. 3.7).

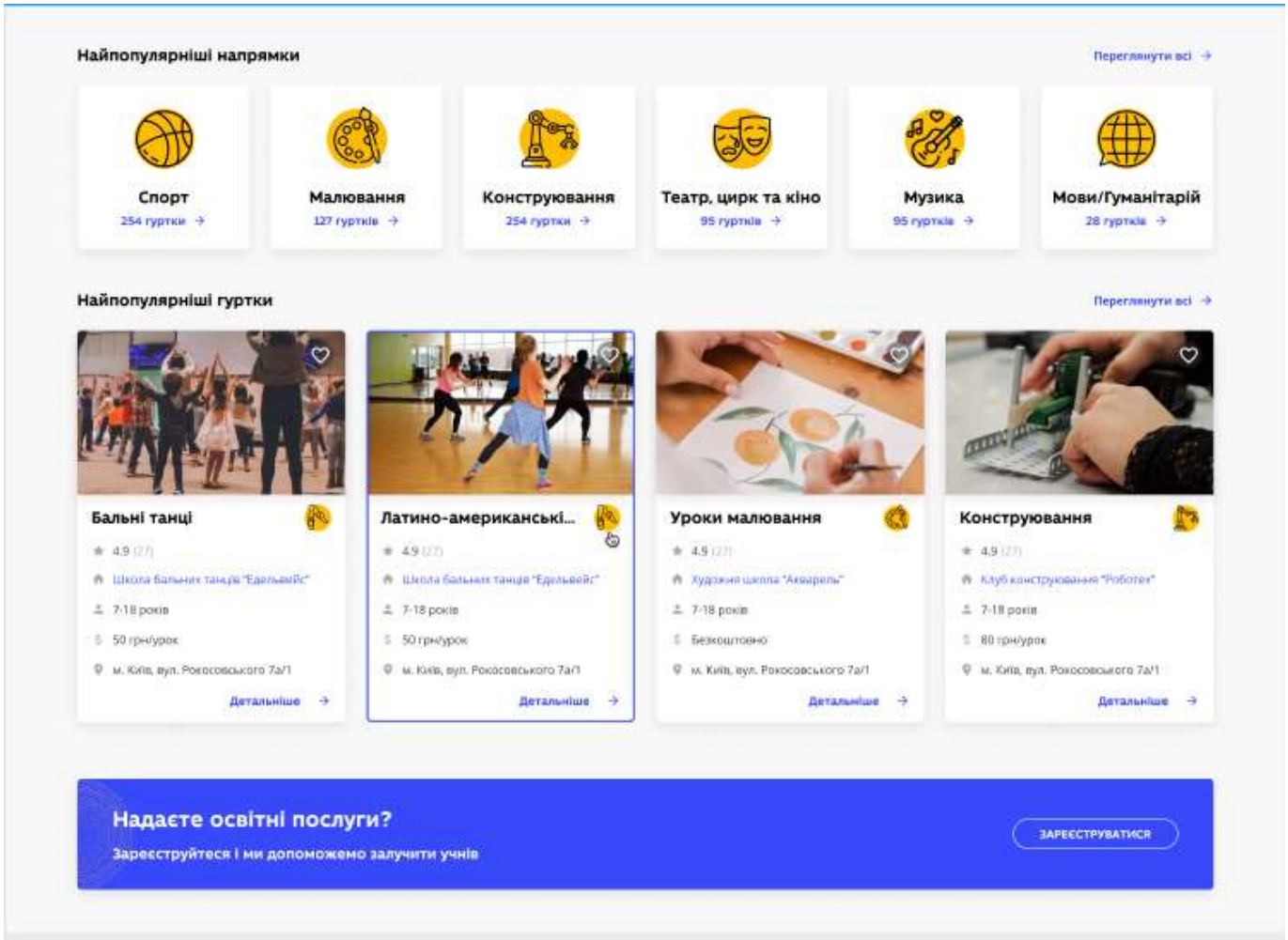

Рисунок 3.7 – Найпопулярніші гуртки та категорії

Далі, для пошуку гуртків, секцій, клубів за фільтрами користувачу потрібно натиснути на «Переглянути всі» гуртки. З’явиться можливість знайти саме той гурток, який потрібен користувачу за допомогою фільтрації у списку чи на

мапі(рис. 3.8, 3.9).

Фільтрувати гуртки можна за:

– напрямками;

– віком дитини;

– днями роботи;

– годинами роботи;

– вартістю;

– наявністю ресурсів для осіб з інвалідністю.

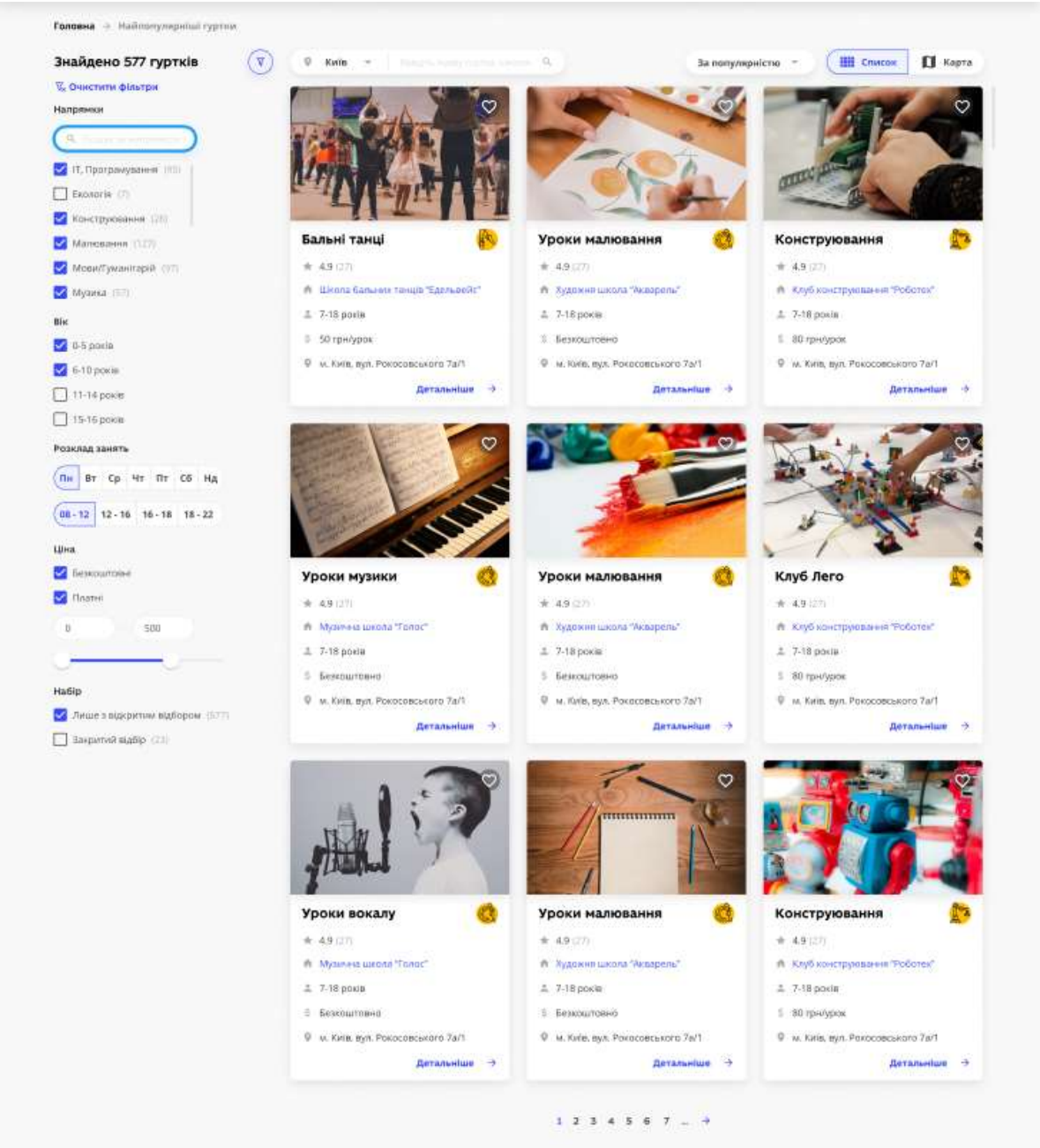

Рисунок 3.8 – Пошук гуртків за фільтрами

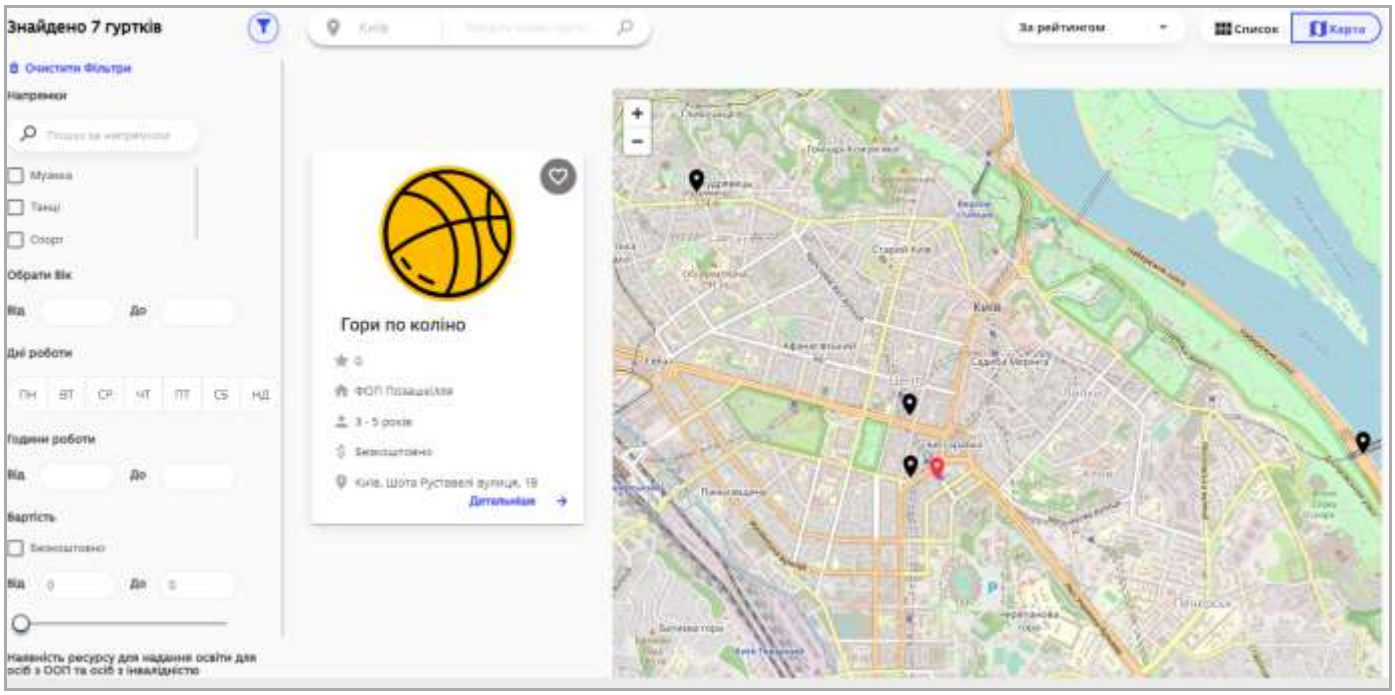

Рисунок 3.9 – Пошук гуртків на мапі

Щоб подати заяву у гурток, клуб, секцію, батькам дитини потрібно виконати де-кілька кроків. Потрібно натиснути кнопку «Увійти/Зареєструватися». Якщо акаунт вже існує, то увійти (рис. 3.10), якщо акаунту не створено, то користувач, або надавач послуг мають зареєструватися у системі (рис 3.11).

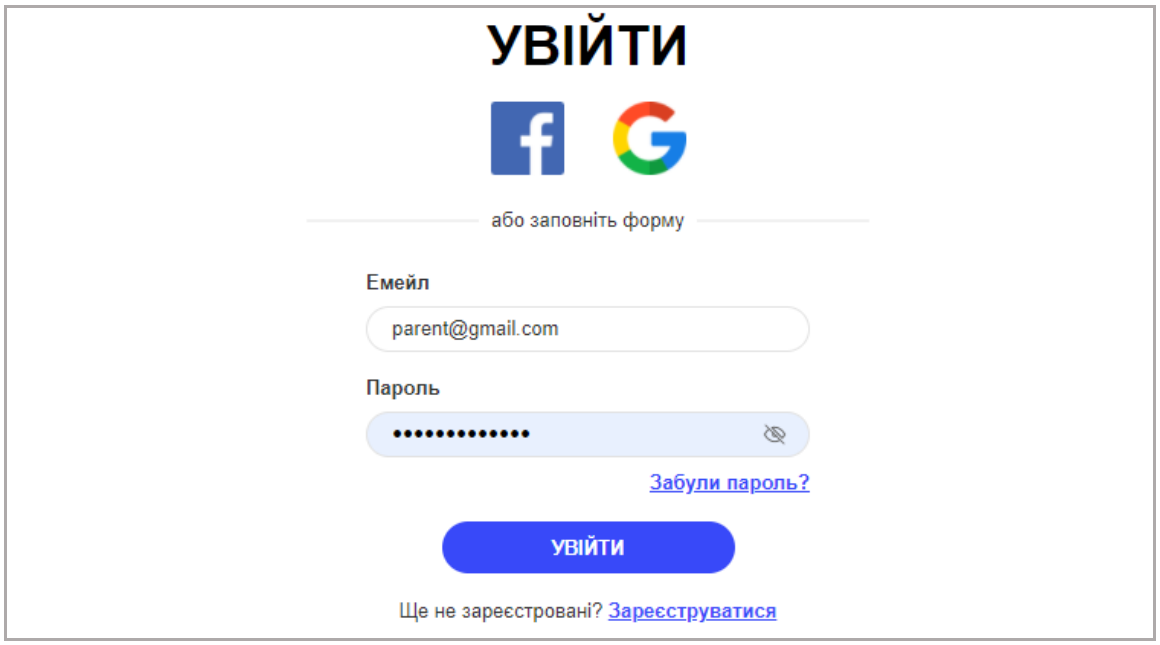

Рисунок 3.10 – Вхід до особистого кабінету

Рисунок 3.11 – Реєстрація особистого кабінету

Після реєстрації користувача, треба натиснути «Особистий кабінет» у відкритому вікні та увійти до свого кабінету. У власно кабінеті за допомогою вкладок можна: переглянути особисту інформацію та відредагувати її (рис. 3.12), переглянути та відредагувати інформацію про дитину, додати дитину (рис. 3.13), переглянути гуртки, заяви та улюблене (див. додаток Б).

Рисунок 3.12 – Особиста інформація користувача

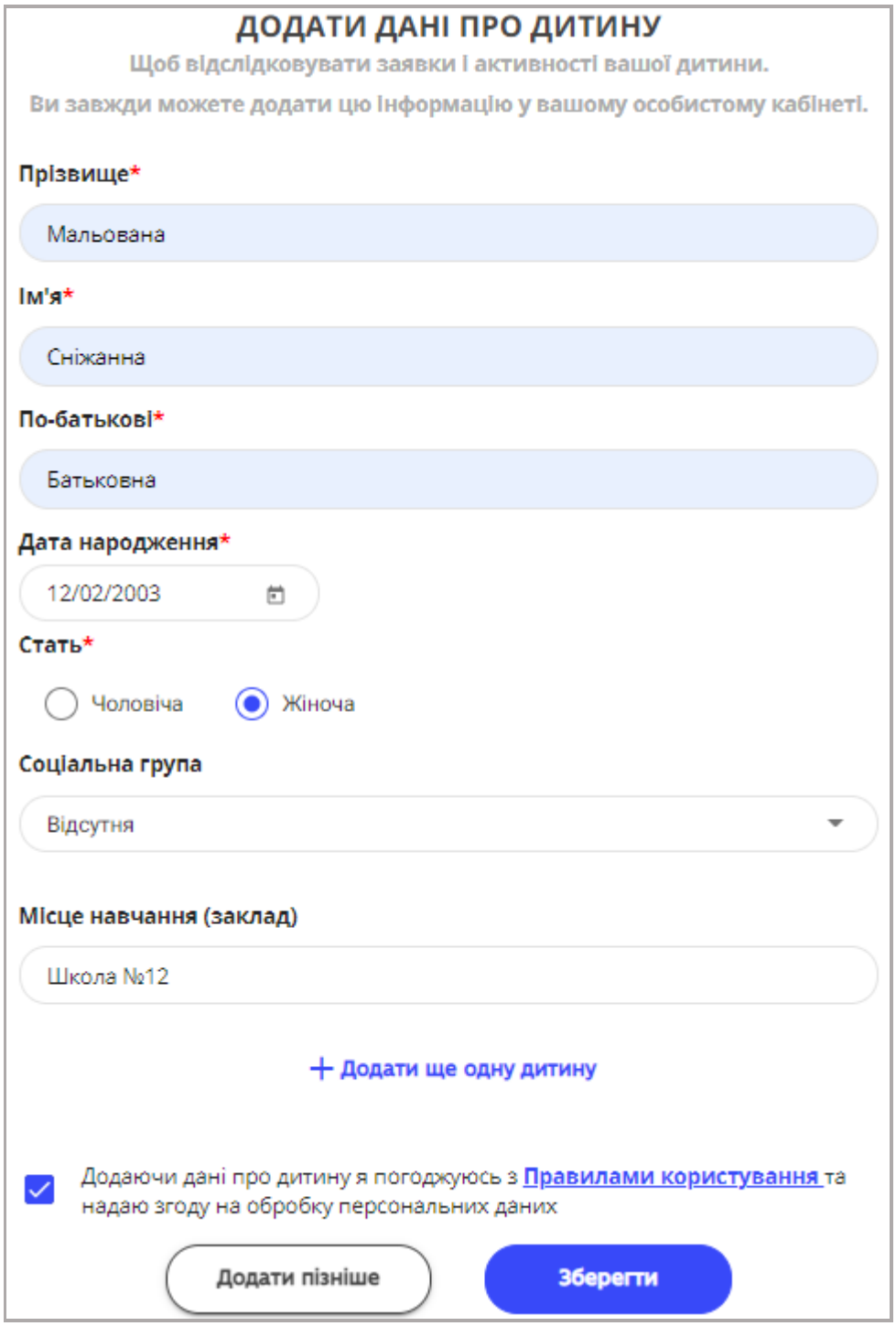

Рисунок 3.13 – Додання даних про дитину

Після додавання даних про дитину батьки можуть подати заяви у гуртки, секції, клуби, які вони можуть знайти за потрібними фільтрами у списку чи на мапі (рис. 3.14).

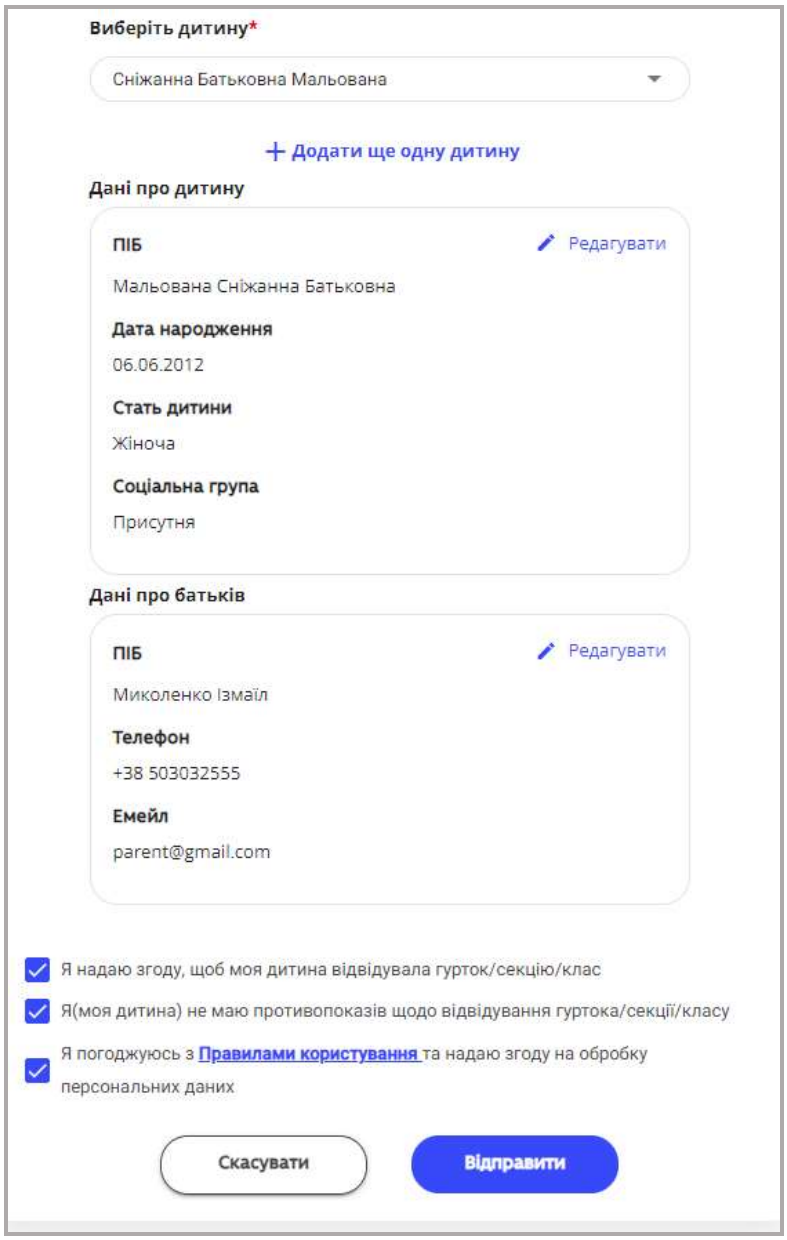

Рисунок 3.14 – Подання заявки у гурток, секцію, клуб

Для подання заявки потрібно обрати дитину, перевірити правильність даних про дитину та власних даних. У разі помилок, відредагувати дані перед створенням заявки натиснувши кнопку «Редагувати». Після встановлення прапорців про надання згоди і погодження з правилами користування, батьки можуть відправити заявку, натиснувши «Відправити», або «Скасувати» для скасування.

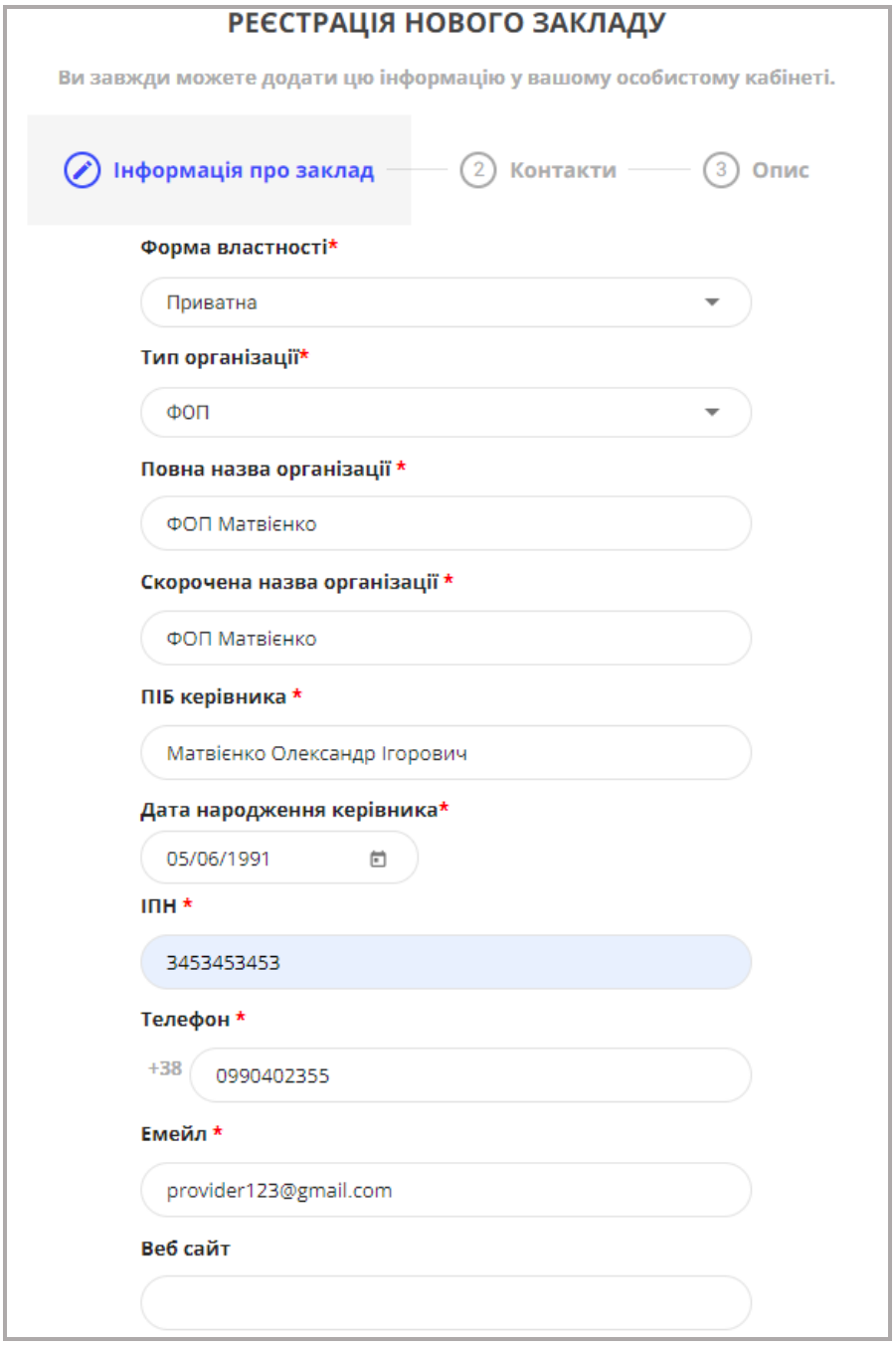

Рисунок 3.15 – Продовження реєстрації надавача послуг

Пройшовши усі кроки реєстрації надавач послуг може переглянути та відредагувати особисту інформацію, переглянути та відредагувати інформацію про організацію, переглянути раніше створені гуртки або створити новий гурток у системі та переглянути отримані заяви на навчання.

Для додавання гуртка створена окрема форма з 4 рівнями реєстрації (рис. 3.16). Потрібно вказати назву гуртка, телефон, email, соціальні мережі,

допустимий вік дітей, дні та години роботи, вартість. У вкладці «Опис» додати фотографії, опис, назву розділу, класи, та ключові слова для пошуку гуртка користувачами. У вкладці «Контакти» вказати місто, вулицю, будинок, або позначити на мапі (рис 3.17). На вкладці «Викладачі» надавач послуг може додати викладачів до гуртка, секції, клубу.

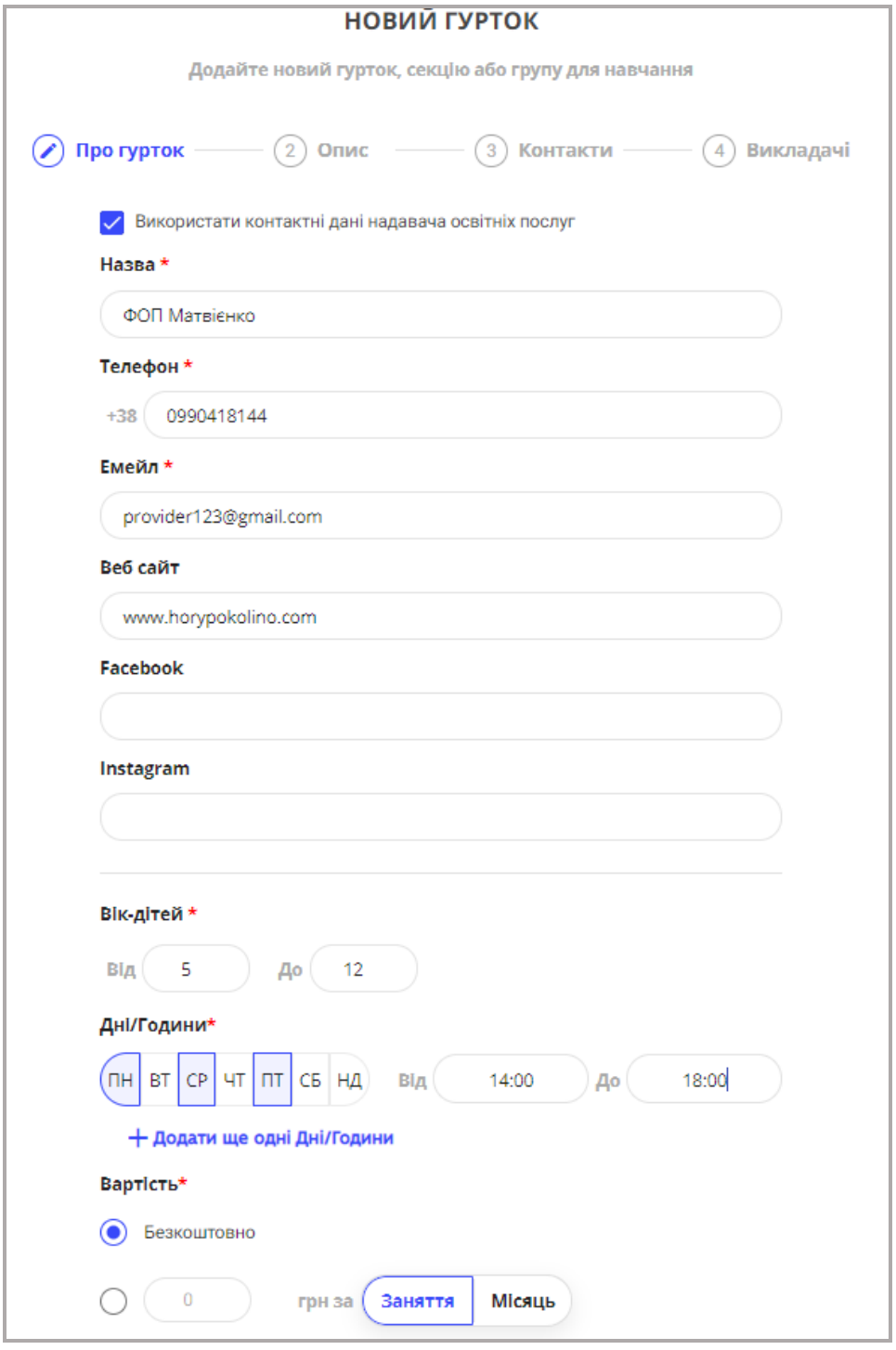

Рисунок 3.16 – Реєстрація гуртка

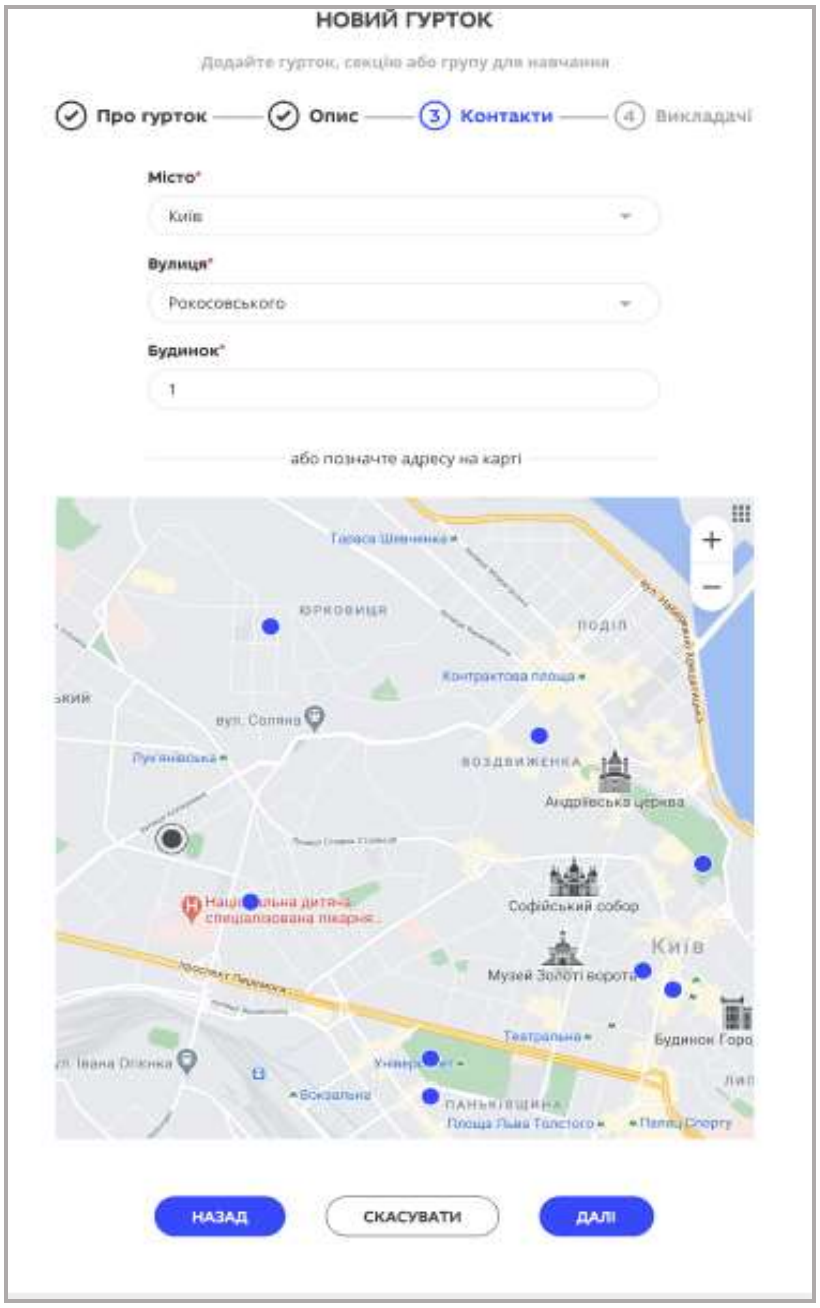

Рисунок 3.17 – Контакти гуртка

Після завершення реєстрації гуртка, надавач послуг та адміністратор гуртка можуть обробляти заявки, які надходять від батьків дітей (рис. 3.18).

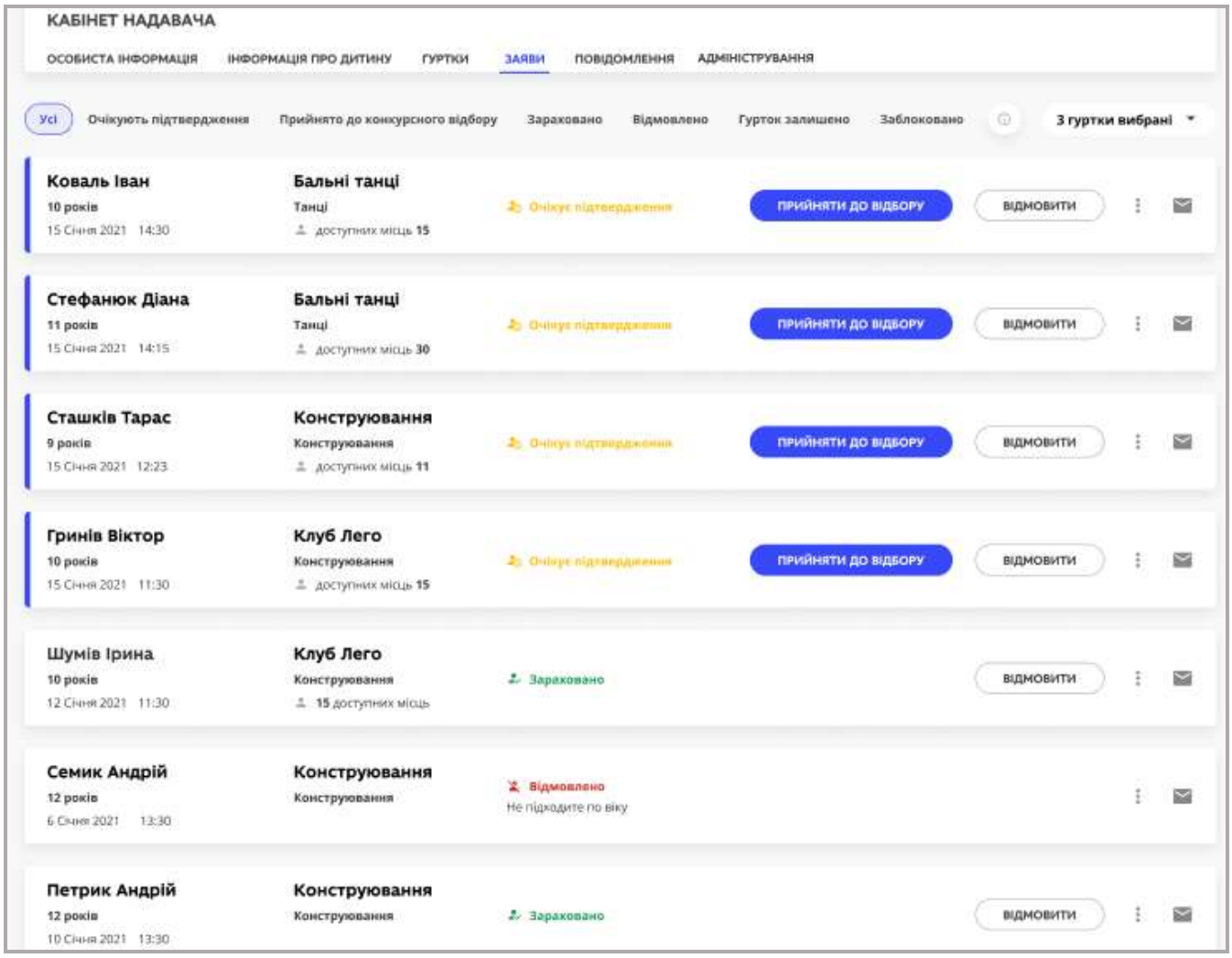

Рисунок 3.18 – Обробка заявок

3.3 Оцінювання очікуваного ефекту від впровадження інформаційної системи

Сучасний розвиток технологій і популяризація вебзастосунків для різних сфер життєдіяльності диктує необхідність постійного підвищення ефективності цих вебзастосунків для забезпечення конкурентоспроможності [51, 52].

Інформація, особливо її автоматизована обробка, залишається важливим фактором підвищення ефективності. Важливу роль у використанні інформації відіграють методи запису, обробки, накопичення та передачі інформації; систематичне зберігання інформації та її видача в необхідну форму; виготовлення нової числової, графічної та іншої інформації [53, 54].

Економічний ефект від впровадження може бути лише опосередкованим, оскільки впроваджені засоби не є прямим джерелом отримання доходу, але мають

велику кількість переваг.

Непрямий економічний ефект важко визначити кількісно. Він проявляється у позитивному впливі впровадження інформаційних систем чи технологій у розвиток позашкільної освіту у всій країні, активізації прогресивних змін, удосконаленні технічного забезпечення та розвитку потенціалу [55].

Оцінка економічного ефекту дає змогу зрозуміти, чи виконано поставлені завдання, для вирішення яких здійснено розробку та впровадження веборієнтованої інформаційної системи гуртків, секцій, клубів.

У нашому випадку непрямий економічний ефект від впровадження веборієнтованої інформаційної системи гуртків, секцій, клубів виявиться у створенні можливості:

– скорочення часу на пошук інформації про гуртки та пошук найближчих за місцем проживання та напрямом діяльності гуртків, клубів, секцій;

– доступ до роботи гуртків (популярність, відвідуваність, наповненість груп);

– отримання швидкого та зручного доступу до унікальної системи в Україні для пошуку закладів позашкільної освіти;

– безкоштовне користування вебресурсом для всіх його користувачів;

– прості алгоритми використання системи для користувачів і надавачів послуг;

– швидка консультація від центру підтримки;

– контроль адміністраторами гуртка, технічного адміна та адміністратора міста за порушеннями та технічними несправностями роботи порталу;

– створення конкурентного середовища між надавачами послуг;

– візуальний аналіз рівню матеріально-технічного забезпечення закладів позашкільної освіти та бачити кадрову наповнюваність (профілі педагогів та відгуки на кожен заклад) [56].

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи магістра досліджено діяльність підприємства «SoftServe». Наведена загальна характеристика діяльності компанії, описані її досягнення. Після дослідження стало зрозуміло, що компанія є однією з найбільших та розвиненіших у нашій країни, та продовжує стрімко прогресувати.

Наступним кроком проаналізовано проблему соціального становлення дітей та молоді, бо у сучасних умовах розвитку України проблема соціального становлення дітей та молоді є дуже актуальною. Деякі соціально-економічні та соціокультурні процеси зумовлюють загострення таких соціальних проблем, як погіршення здоров'я дітей та молоді, різні девіації у молодіжному середовищі, а також впливають на соціально-ціннісну невизначеність частини молодих людей.

Під час виконання кваліфікаційної роботи проведено аналіз процесів інформаційної системи гуртків, секцій, клубів. Сформовано вимоги до створюваної веборієнтованої інформаційної системи.

Структурно-функціональна схема порталу спроектована відповідно до основних задач системи. За допомогою аналізу інформаційних потоків, сформоване розуміння про те, що саме згідно цілей і завдань має бути розміщено на сторінках веборієнтованої системи. Для всіх сторінок додатку розроблені макети та вказані основні елементи інтерфейсу сторінок і форм.

Враховуючи можливості і переваги окремих технологій створення вебдодатків, підібрано набір актуальних та сучасних технологій розробки. Використані такі програмні засоби, технології, та мови програмування:Asp .NET Core 3, C#, MySQL, Figma, Microsoft Visual Studio 2019, GitHub.

За допомогою сучасних технологій розроблено бекенд частину веборієнтованої інформаційної системи, що має такі функції, як: дізнатися інформацію про портал, звернутися до підтримки, змінити мову, увійти або зареєструватися на порталі, переглянути найпопулярніші гуртки та напрямки,

зробити пошук за фільтрами і відображати результат у списку чи на мапі, створити дитину та подати заявку у гурток, переглядати статуси заявок, створити організацію та власний гурток, приймати чи відхиляти заявки від батьків дітей.

Розроблено інструкцію використання веборієнтованої інформаційної системи гуртків, секцій, клубів для успішного впровадження і полегшення розуміння можливостей системи та способу її використання.

Результати дослідження пройшли апробацію – опубліковані тези на Міжнародній науковій інтернет-конференції "Topical tendencies of science and practice" (10 грудня 2021 р., Едмонтон, Канада) [70].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка та застосування інформаційних систем в освітній практиці. URL: https://xreferat.com/71/7163-1-osobennosti-sistemy-dopolnitel-nogo-obrazovaniya-deteiy-i-podrostkov.html (дата звернення: 08.11.2021) .

2. Позашкілля є інструментом, який дозволяє розкрити потенціал дитини. URL: https://mon.gov.ua/ua/news/pozashkillya-ye-instrumentom-yakij-dozvolyaye-rozkriti-potencial-ditini-sergij-shkarlet-pid-chas-vseukrayinskogo-forumu-pozashkilna-osvita-bez-baryeriv-model-majbutnogo (дата звернення: 08.11.2021).

3. Про нас. URL: https://www.softserveinc.com/en-us/about-us (дата звернення: 08.11.2021).

4. Твоє місто медіа. URL: https://tvoemisto.tv/exclusive/istoriya_uspihu_softserve_globalna_kompaniya_z_lvivskym_korinnyam_77612.html (дата звернення: 08.11.2021).

5. SoftServe History. URL: https://uk.wikipedia.org/wiki/SoftServe (дата звернення: 08.11.2021).

6. SoftServe став офіційним торговим представником Google Cloud. URL:https://eba.com.ua/softserve-stav-ofitsijnym-torgovym-predstavnykom-google-cloud-u-velykobrytaniyi-ta-irlandiyi/. (дата звернення: 08.11.2021).

7. Corporate social responsibility. URL: https://www.softserveinc.com/en-us/corporate-social-responsibility. (дата звернення: 08.11.2021).

8. Види й типи організаційних структур та умови їх ефективного застосування. URL: https://pidru4niki.com/10880405/menedzhment/vidi_tipi_organizatsiynih_struktur_umovi_efektivnogo_zastosuvannya (дата звернення: 08.11.2021).

9. Енциклопедія гібридних методів управління проектами. URL: https://kachestvo.pro/kachestvo-upravleniya/proektnoe-upravlenie/entsiklopediya-gibridnykh-metodov-upravleniya (дата звернення: 09.11.2021).

10. What Are Information Systems: Definition & Types. URL:

https://study.com/academy/lesson/what-are-information-systems-definition-types-quiz.html (дата звернення: 09.11.2021).

11. What is FURPS+. URL: https://businessanalysttraininghyderabad.wordpress.com/2014/08/05/what-is-furps/ (дата звернення: 09.11.2021).

12. Complete Beginner's Guide to Information Architecture UX Booth. URL: http://www.uxbooth.com/articles/complete-beginners-guide-toinformation-architecture/.

13. What Are Information Systems: Definition & Types. URL: https://study.com/academy/lesson/what-are-information-systems-definition-types-quiz.html (дата звернення: 09.11.2021).

14. Commentary: Supplementary Specification. URL: https://sites.cs.ucsb.edu/~mikec/cs48/project/RequirementsLarman.pdf (дата звернення: 10.11.2021).

15. The SunUML Analysis Model. URL: http://www.cs.sjsu.edu/~pearce/modules/projects/ooa/sunUML/index.htm (дата звернення: 10.11.2021).

16. Моделі і методи проектування інформаційних систем. URL: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20151118130027/183252/index.html (дата звернення: 10.11.2021).

17. The Extensive Guide to Business Processes. URL: https://kissflow.com/workflow/bpm/business-process/ (дата звернення: 14.11.2021).

18. Using Business Process Modeling and Tools. URL: https://www.processmaker.com/blog/business-process-modeling/ (дата: 14.11.2021).

19. Business Process Diagram maker with professional tools and templates. URL: https://online.visual-paradigm.com/diagrams/features/bpmn-tool/ (дата звернення: 14.11.2021).

20. Information System Architecture (ISA) - Classification, Services, Model and Components of Information System. URL: https://www.toppers4u.com/2020/11/information-system-architecture-isa.html (дата звернення: 16.11.2021).

21. Information system infrastructure and architecture. URL: https://www.britannica.com/topic/information-system/Information-system-infrastructure-and-architecture (дата звернення: 16.11.2021).

22. Розробка структури сайту. URL: https://www.can-all.com/6-steps-to-

conquer-the-internet/development-site-structure.html (дата звернення: 16.11.2021).

23. Client-Server Architecture. URL: https://www.techopedia.com /definition/438/clientserver-architecture (дата звернення: 18.11.2021).

24. Client-Server Architecture. URL: https://www.britannica.com /technology/client-server-architecture (дата звернення: 18.11.2021).

25. Client-Server Architecture. URL: https://www.sciencedirect.com/topics /computer-science/client-server-architecture (дата звернення: 18.11.2021).

26. Взаємодія клієнт-сервер. URL: https://vseosvita.ua/library/urok-z-temi-vzaemodia-klient-server-425357.html (дата звернення: 18.11.2021).

27. Ming Chieh Huang, Fu Yin. Lee John (2015). Comparison Between MongoDB and MS-SQL Databases on the TWC Website. American Journal of Software Engineering and Applications. DOI: 10.11648/j.ajsea.20150402.12.

28. Навігація на сайті. URL: https://www.taina.com.ua/navihacija-na-sajti/ (дата звернення: 18.11.2021).

29. Основні етапи створення сайту. URL: https://repair.lviv.ua/stvorennya-sajtiv/osnovni-etapi-stvorennya-sajtu/ (дата звернення: 18.11.2021).

30. Структура сайту: чому вона важлива і як її змінювати? URL: https://host4.biz/uk/blog/struktura-sajtu-chomu-vona-vazhliva-i-yak-yiyi-zminyuvati.

31. Figma for beginners. URL: https://www.theme-junkie.com/what-is-figma/ (дата звернення: 18.11.2021).

32. Additional tips on component architecture in Figma URL: https://www.figma.com/best-practices/component-architecture/additional-tips/. (дата звернення: 19.11.2021).

33. Методы HTTP запиту. URL: https://qastart.by/class-2/21-metody-http-zaprosa (дата звернення: 19.11.2021).

34. Призначення і основні функціональні можливості. URL: https://www.bigdataschool.ru/wiki/elasticsearch (дата звернення: 19.11.2021).

35. Чому Elasticsearch – хороший вибір для збору та аналізу даних середнього обсягу. URL: https://tproger.ru/blogs/why-elasticsearch-is-a-good-choice/ (дата звернення: 20.11.2021).

36. IdentityServer для власних хмарних додатків. URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/cloud-native/identity-server (дата звернення: 20.11.2021).

37. .NET Core Identity Server 4 VS автентифікація. URL: https://qastack.ru/programming/42121854/net-core-identity-server-4-authentication-vs-identity-authentication (дата звернення: 20.11.2021).

38. Microsoft Visual Studio. URL: https://visualstudio.microsoft.com/ (дата звернення: 20.11.2021).

39. Visual Studio 2019 Community. URL: https://info-comp.ru/programmirovanie/739-install-visual-studio-2019-community.html (дата звернення: 20.11.2021).

40. MySQL. URL: https://searchoracle.techtarget.com/definition/MySQL (дата звернення: 20.11.2021).

41. What is MySQL? URL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. (дата звернення: 21.11.2021).

42. Введення в MySQL. URL: https://metanit.com/sql/mysql/1.1.php (дата звернення: 20.11.2021).

43. Вступ до ASP.NET Core. URL: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0 (дата звернення: 21.11.2021).

44. Що таке ASP.NET Core? URL: https://itvdn.com/ru/blog/article/aspnet_core (дата звернення: 21.11.2021).

45. What is ASP.NET Core? URL: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core. (дата звернення: 21.11.2021).

46. Введення в Web API. URL: https://metanit.com/sharp/aspnet5/23.1.php (дата звернення: 21.11.2021).

47. Особливості та структура інформаційного забезпечення автоматизованої інформаційної системи. URL: http://studcon.org/osoblyvosti-ta-struktura-informaciynogo-zabezpechennya-avtomatyzovanoyi-informaciynoyi-systemy (дата звернення: 21.11.2021).

48. Організація інформаційних систем. URL: https://pidru4niki.com/81325 /tehnika/organizatsiya_informatsiynogo_zabezpechennya (дата звернення: 21.11.2021).

49. What is a Relational Database. URL: https://www.oracle.com/database/what-is-a-relational-database/ (дата звернення: 22.11.2021).

50. GUIDs In C# And .NET. URL: https://www.c-sharpcorner.com/UploadFile /prasoonk/guids-in-C-Sharp-and-net/ (дата звернення: 22.11.2021).

51. Оцінка ефективності застосування інформаційних технологій. URL: https://tourlib.net/statti_ukr/melnychenko15.htm (дата звернення: 23.11.2021).

52. Особливості оцінювання економічної ефективності. URL: https://mmi.fem.sumdu.edu.ua/sites/default/files/mmi2012_4_143_153.pdf (дата звернення: 23.11.2021).

53. Ефективність інформаційних систем. URL: https://financial.lnu.edu.ua/wp-content/uploads/2017/09/L_EIS.pdf (дата звернення: 23.11.2021).

54. Критеріальне оцінювання ефективності інформаційних пристроїв та систем. URL: http://voycehovska.vk.vntu.edu.ua/file/b1aa16068384b41 cba942b1cf70a578b.pdf (дата звернення: 23.11.2021).

55. Асеев Г. Г. Співвідношення різних метричних досліджень у наукознавстві. Системи обробки інформації, вип. No1 (147). 2017. URL: http://www.hups.mil.gov.ua/periodic-app/article/17299 (дата звернення: 23.11.2021).

56. Проектування додатків. URL: https://stud.com.ua/97678/informatika /proektuvannya_dodatkiv (дата звернення: 24.11.2021).

57. Розробка вебдодатків. URL: https://armedsoft.com/ua/services/rozrobka-veb-dodatkiv (дата звернення: 24.11.2021).

58. Вимоги до системи: класифікація FURPS+. URL: https://sysana.wordpress.com/2010/09/16/furps/ (дата звернення: 24.11.2021).

59. The .NET Platform. URL: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet (дата звернення: 24.11.2021).

60. System Testing. URL: https://www.guru99.com/system-testing.html (дата звернення: 25.11.2021).

61. Adam Freeman, 2020, Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC 3, ISBN 978-1484203989.

62. Пестрецова О. О. Розвиток інформаційних технологій як ключовий фактор впливу на процес соціокультурних трансформацій в сучасному суспільстві. Science and Education a New Dimension. Humanities and Social Sciences, V(25), I.: 147. 2017. URL: http://seanewdim.com/uploads/3/4/5/1/34511564/ httpsdoi.org10.31174 sendhs2017-147v25-15.pdf (дата звернення: 25.11.2021).

63. What is a web developer. URL: https://www.bitdegree.org/tutorials/what-is-a-web-developer/ (дата звернення: 26.11.2021).

64. Five Problems Business Process Automation Can Solve. HelpSystems Blog. 2016. URL: https://www.helpsystems.com/blog/fiveproblems-business-process-automation-can-solve (дата звернення: 26.11.2021).

65. McNally M. Enterprise content management systemsand the application of Taylorism and Fordism to intellectual labour. 2010. URL: http://www.ephemerajournal.org/sites/default/files/10-3mcnally.pdf (дата звернення: 26.11.2021).

66. Silberschatz A. B., Korth H. F., and S., 2013. Sudarshan. Database System Concepts. McGraw-Hill, sixth edition.

67. Beynon-Davies, Paul , 2014. Database Systems . Basingstoke, UK: Palgrave: Houndmills. ISBN 1403916012.

68. Питер Лабберс, Брайан Олберс, Фрэнк Салим. HTML5 для профессионалов: мощные инструменты для разработки современных вебприложений. – М.: «Вильямс», 2011. – 272 с.

69. Флэнаган Д. 13.8.1. Чего не может JavaScript. Подробное руководство = JavaScript. The Definite Guide. – 5-е изд., 2012. – С. 280, 281.

70. Necheporenko I., Yatsenko V. Development of a web-based information system of workshops and clubs. XII International Science Conference «Topical tendencies of science and practice» (December 07 – 10, 2021). Edmonton. Canada. P. 506-507. https://isg-konf.com/ru/topical-tendencies-of-science-and-practice-ru/

ДОДАТКИ

Додаток А
(обов’язковий)

SUMMARY

Necheporenko I.D. Development of a web-based information system of workshops and clubs – Master-level Qualification Thesis. Sumy State University, Sumy, 2021.

As a result of the research, the activity of SoftServe enterprise is analyzed, the requirements to the web-oriented information system of workshops and clubs are formed, the architecture and technologies of the system are described, the database with necessary data tables is created, the model of information system is constructed, developed a prototype of a web-oriented information system.

Keywords: web-based information system, workshop, club, database, portal, ASP.Net Core 3, C #, MySQL.

АНОТАЦІЯ

Нечепоренко І.Д. Розробка веборієнтованої інформаційної системи гуртків, секцій, клубів. – Кваліфікаційна робота магістра. Сумський державний університет, Суми, 2021 р.

У результаті дослідження проаналізовано діяльність підприємства «SoftServe», сформовано вимоги до веборієнтованої інформаційної системи гуртків, секцій, клубів, описано архітектуру та технології вирішення задачі розробки системи, створено базу даних з необхідними таблицями даних, побудовано макет інформаційної системи, розроблено прототип веборієнтованої інформаційної системи.

Ключові слова: веборієнтована інформаційна система, гурток, секція, клуб, база даних, портал, ASP.Net Core 3, C#, MySQL.

Додаток Б

Рисунок Б.1 – Кабінет користувача «Заявки»

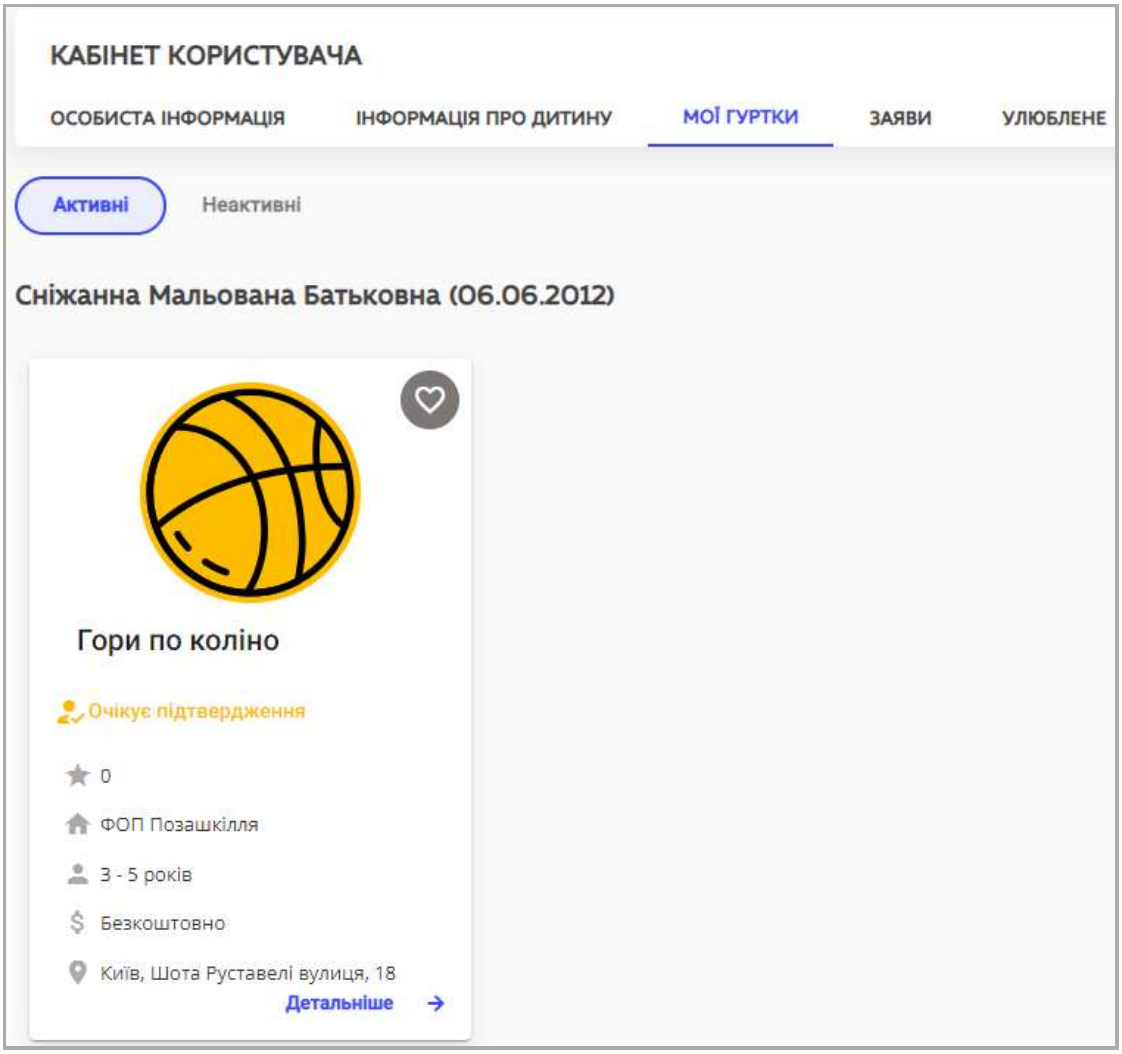

Рисунок Б.2 – Кабінет користувача «Мої гуртки»

Рисунок Б.3 – Кабінет користувача «Інформація про дитину»

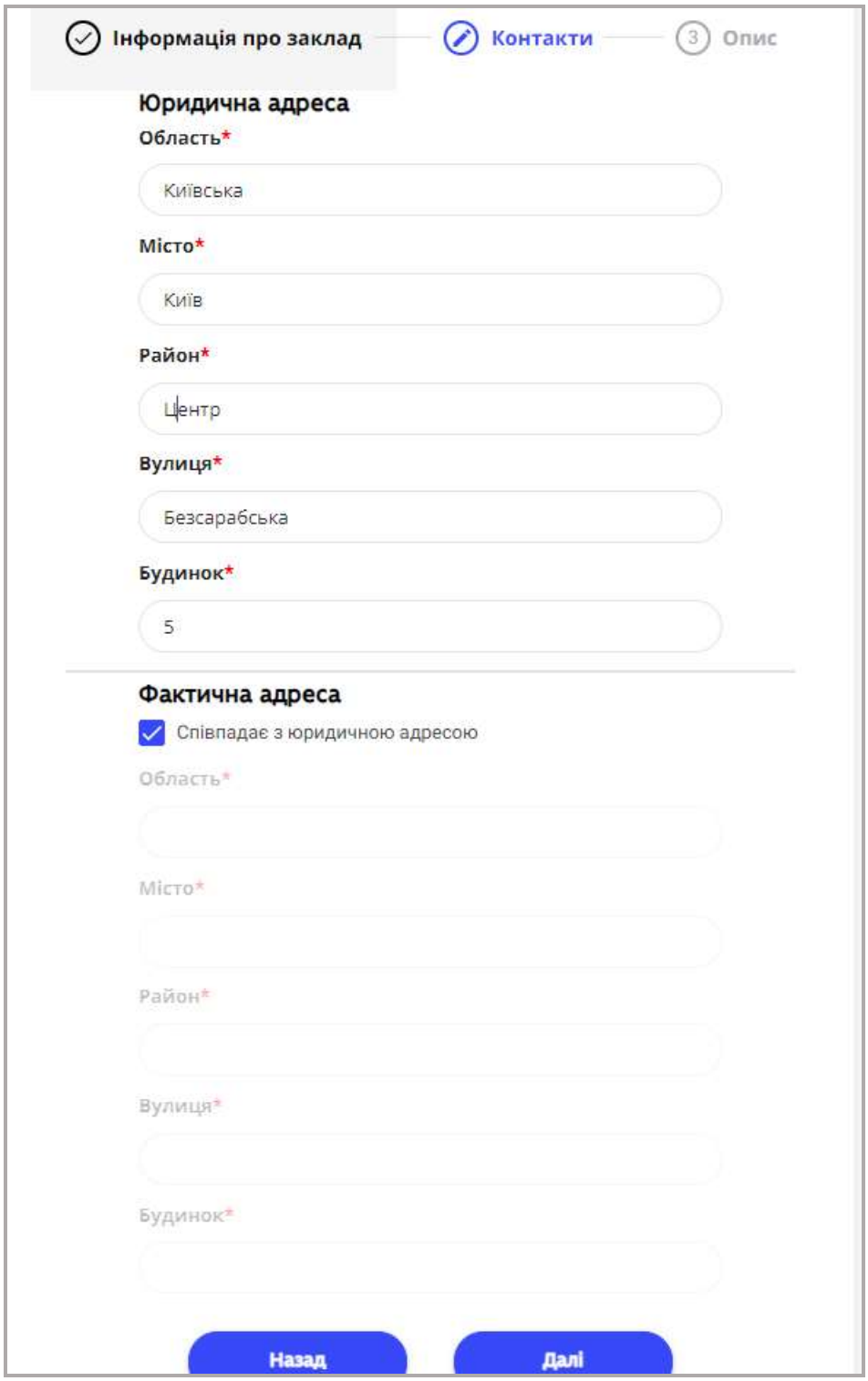

Рисунок Б.4 – Додавання закладу «Інформація про заклад»

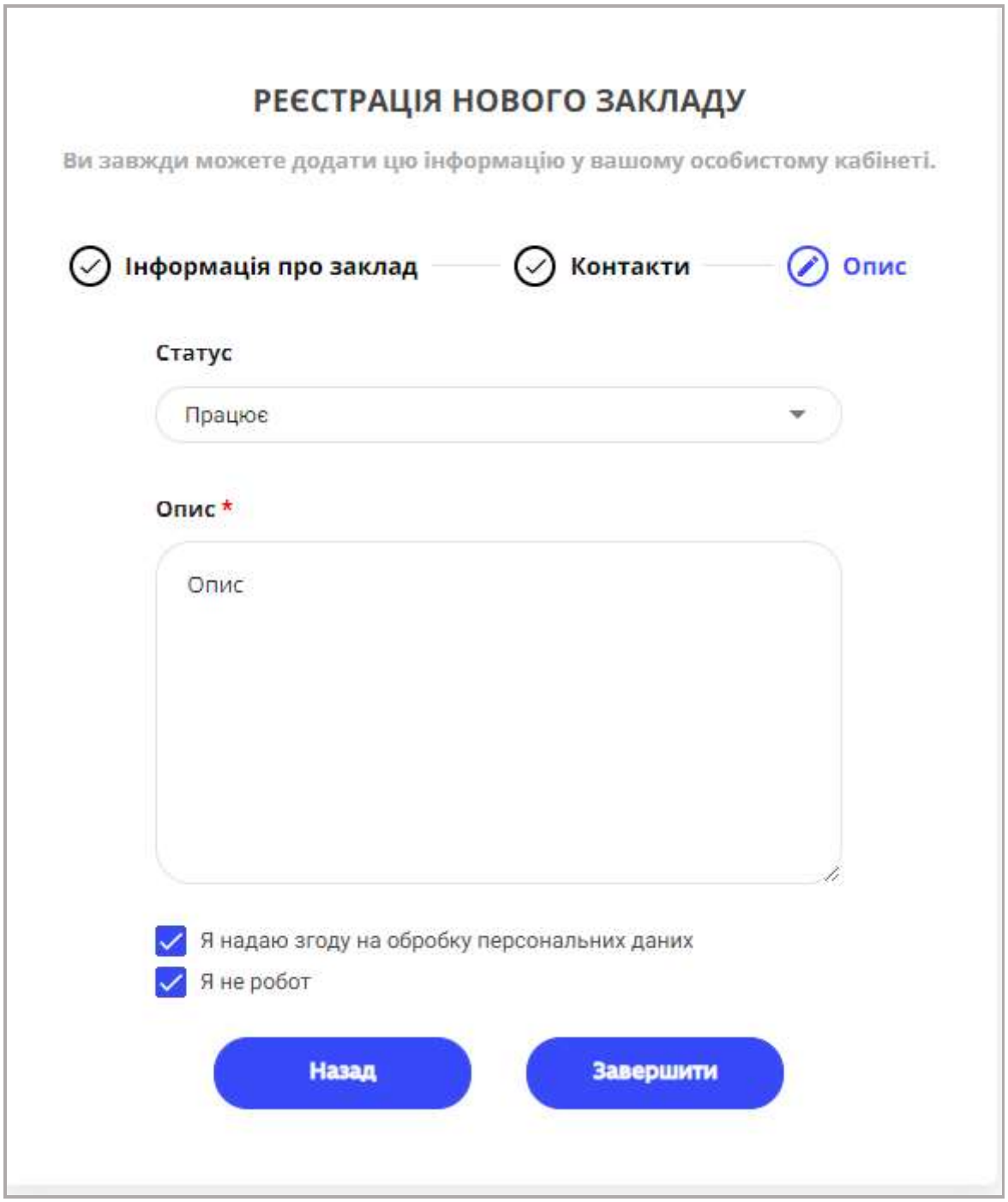

Рисунок Б.5 – Додавання закладу «Опис»

Додаток В

Файл ApplicationController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Localization;
using OutOfSchool.Common.PermissionsModule;
using OutOfSchool.Services.Enums;
using OutOfSchool.WebApi.ApiModels;
using OutOfSchool.WebApi.Extensions;
using OutOfSchool.WebApi.Models;
using OutOfSchool.WebApi.Services;

namespace OutOfSchool.WebApi.Controllers.V1
{
   /// <summary>
   /// Controller with CRUD operations for a Application entity.
   /// </summary>
   [ApiController]
   [ApiVersion("1.0")]

[Route("api/v{version:apiVersion}/[controller]/[action]")]
   public class ApplicationController : ControllerBase
   {
      private readonly IApplicationService applicationService;
      private readonly IParentService parentService;
      private readonly IProviderService providerService;
      private readonly IWorkshopService workshopService;
      private readonly IStringLocalizer<SharedResource> localizer;

      /// <summary>
      /// Initializes a new instance of the <see cref="ApplicationController"/> class.
      /// </summary>
      /// <param name="applicationService">Service for Application model.</param>
      /// <param name="localizer">Localizer.</param>
      /// <param name="providerService">Service for Provider model.</param>
      /// <param name="parentService">Service for


Parent model.</param>
      /// <param name="workshopService">Service for Workshop model.</param>
      public ApplicationController(
         IApplicationService applicationService,
         IStringLocalizer<SharedResource> localizer,
         IProviderService providerService,
         IParentService parentService,
         IWorkshopService workshopService)
      {
         this.applicationService = applicationService;
         this.localizer = localizer;
         this.providerService = providerService;
         this.parentService = parentService;
         this.workshopService = workshopService;
      }

      /// <summary>
      /// Get all applications from the database.
      /// </summary>
      /// <returns>List of all applications.</returns>
      /// <response code="200">All entities were found.</response>
      /// <response code="204">No entity was found.</response>
      /// <response code="500">If any server error occures.</response>
      [HasPermission(Permissions.SystemManagement)]

[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<ApplicationDto>))]

[ProducesResponseType(StatusCodes.Status204NoContent)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]
      [HttpGet]
      public async Task<IActionResult> Get()
      {
         var applications = await applicationService.GetAll().ConfigureAwait(false);

         if (!applications.Any())
         {
            return NoContent();
         }
         return Ok(applications);
      }

      /// <summary>
      /// Get application by it's id.
      /// </summary>
      /// <param name="id">The key in the database.</param>
      /// <returns><see cref="ApplicationDto"/>.</returns>
      /// <response code="200">The entity was found by given Id.</response>
      /// <response code="204">No entity with given Id was found.</response>
      /// <response code="500">If any server error occures.</response>
[HasPermission(Permissions.ApplicationRead)]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ApplicationDto))]
[ProducesResponseType(StatusCodes.Status204NoConte
```

```
nt)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
      [HttpGet("{id}")]
      public async Task<IActionResult> GetById(Guid
id)
      {
         var application = await
applicationService.GetById(id).ConfigureAwait(false);
         if (application is null)
         {
            return NoContent();
         }
         try
         {
            await CheckUserRights(
               parentId: application.ParentId,
               providerId:
application.Workshop.ProviderId)
               .ConfigureAwait(false);

            return Ok(application);
         }
         catch (ArgumentException ex)
         {
            return BadRequest(ex.Message);
         }
      }
      /// <summary>
      /// Get Applications by Parent Id.
      /// </summary>
      /// <param name="id">Parent id.</param>
      /// <returns>List of applications.</returns>
      /// <response code="200">Entities were found by
given Id.</response>
      /// <response code="204">No entity with given Id
was found.</response>
      /// <response code="500">If any server error
occures.</response>
 [HasPermission(Permissions.ApplicationRead)]
[ProducesResponseType(StatusCodes.Status200OK,
Type = typeof(IEnumerable<ApplicationDto>))]
[ProducesResponseType(StatusCodes.Status400BadReq
uest)]
[ProducesResponseType(StatusCodes.Status401Unautho
rized)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
 [HttpGet("{id}")]

      public async Task<IActionResult>
GetByParentId(Guid id)
      {
         try
         {
            await CheckUserRights(parentId:
id).ConfigureAwait(false);
         }
         catch (ArgumentException ex)
         {
            return BadRequest(ex.Message);
         }

         var applications = await
applicationService.GetAllByParent(id).ConfigureAwait(f
alse);

         if (!applications.Any())
         {
            return NoContent();
         }

         return Ok(applications);
      }

      /// <summary>
      /// Get Applications by Provider or Workshop Id.
      /// </summary>
      /// <param name="property">Property to find by
(workshop or provider).</param>
      /// <param name="id">Provider or Workshop
id.</param>
      /// <param name="filter">Application
filter.</param>
      /// <returns>List of applications.</returns>
      /// <response code="200">Entities were found by
given Id.</response>
      /// <response code="204">No entity with given Id
was found.</response>
      /// <response code="500">If any server error
occures.</response>
      [HasPermission(Permissions.ApplicationRead)]

[ProducesResponseType(StatusCodes.Status200OK,
Type = typeof(IEnumerable<ApplicationDto>))]
[ProducesResponseType(StatusCodes.Status400BadReq
uest)]
[ProducesResponseType(StatusCodes.Status401Unautho
rized)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
[HttpGet("{property:regex(^provider$|^workshop$)}/{id
}")]
      public async Task<IActionResult>
GetByPropertyId(string property, Guid id, [FromQuery]
ApplicationFilter filter)
      {
         IEnumerable<ApplicationDto> applications =
default;

         try
         {
            if (property.Equals("provider",
StringComparison.CurrentCultureIgnoreCase))
            {
               applications = await GetByProviderId(id,
filter).ConfigureAwait(false);
            }
            else if (property.Equals("workshop",
StringComparison.CurrentCultureIgnoreCase))
            {
               applications = await GetByWorkshopId(id,
filter).ConfigureAwait(false);
            }
```

```
            }
            catch (ArgumentException ex)
            {
                return BadRequest(ex.Message);
            }

            if (!applications.Any())
            {
                return NoContent();
            }

            return Ok(applications);
        }
        /// <summary>
        /// Get Applications by Status.
        /// </summary>
        /// <param name="status">Application
status.</param>
        /// <returns>List of applications.</returns>
        /// <response code="200">Entities were found by
given status.</response>
        /// <response code="204">No entity with given
status was found.</response>
        /// <response code="500">If any server error
occures.</response>
 [HasPermission(Permissions.ApplicationRead)]
[ProducesResponseType(StatusCodes.Status200OK,
Type = typeof(IEnumerable<ApplicationDto>))]
[ProducesResponseType(StatusCodes.Status400BadReq
uest)]
[ProducesResponseType(StatusCodes.Status401Unautho
rized)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
[HttpGet]
        public async Task<IActionResult> GetByStatus(int
status)
        {
            try
            {
                ValidateStatus(status);
            }
            catch (ArgumentOutOfRangeException ex)
            {
                return BadRequest(ex.Message);
            }

            var applications = await
applicationService.GetAllByStatus(status).ConfigureAw
ait(false);

            if (!applications.Any())
            {
                return NoContent();
            }

            return Ok(applications);
        }

        /// <summary>
        /// Method for creating a new application.
        /// </summary>
        /// <param
name="applicationApiModel">Application api model
with data to add.</param>
        /// <returns>A <see cref="Task{TResult}"/>
representing the result of the asynchronous
operation.</returns>
        [HasPermission(Permissions.ApplicationAddNew)]
[ProducesResponseType(StatusCodes.Status201Created)
]
[ProducesResponseType(StatusCodes.Status400BadReq
uest)]
[ProducesResponseType(StatusCodes.Status401Unautho
rized)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
 [HttpPost("multiple")]
        [Obsolete("This method is obsolete. Call another
Create instead", false)]
        public async Task<IActionResult>
Create([FromBody] ApplicationApiModel
applicationApiModel)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            try
            {
                var applications =
CreateMultiple(applicationApiModel);

                var newApplications = await
applicationService.Create(applications).ConfigureAwait(
false);

                var ids = newApplications.Select(a => a.Id);

                return CreatedAtAction(
                    nameof(GetById),
                    new { id = ids, },
                    newApplications);
            }
            catch (ArgumentException ex)
            {
                return BadRequest(ex.Message);
            }
        }

        /// <summary>
        /// Method for creating a new application.
        /// </summary>
        /// <param name="applicationDto">Application
entity to add.</param>
        /// <returns>A <see cref="Task{TResult}"/>
representing the result of the asynchronous
operation.</returns>
        /// <response code="201">Entity was created and
returned with Id.</response>
        /// <response code="400">If the model is invalid,
some properties are not set etc.</response>
        /// <response code="401">If the user is not
```

```
authorized.</response>
        /// <response code="500">If any server error
occurs.</response>
        [HasPermission(Permissions.ApplicationAddNew)]
[ProducesResponseType(StatusCodes.Status201Created,
Type = typeof(ApplicationDto))]
[ProducesResponseType(StatusCodes.Status400BadReq
uest)]
[ProducesResponseType(StatusCodes.Status401Unautho
rized)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
[HttpPost]
        public async Task<IActionResult>
Create(ApplicationDto applicationDto)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (applicationDto is null)
            {
                return BadRequest(localizer[$"Application dto
should not be null"]);
            }
            try
            {
                await CheckUserRights(parentId:
applicationDto.ParentId).ConfigureAwait(false);

                applicationDto.Id = default;

                applicationDto.CreationTime =
DateTimeOffset.UtcNow;

                applicationDto.Status =
ApplicationStatus.Pending;

                var application = await
applicationService.Create(applicationDto).ConfigureAw
ait(false);
                return CreatedAtAction(
                    nameof(GetById),
                    new { id = application.Id, },
                    application);
            }
            catch (ArgumentException ex)
            {
                return BadRequest(ex.Message);
            }
        }
        /// <summary>
        /// Update info about a specific application in the
database.
        /// </summary>
        /// <param name="applicationDto">Application
entity.</param>
        /// <returns><see
cref="ApplicationDto"/>.</returns>
        /// <response code="200">Entity was updated and
returned.</response>
        /// <response code="400">If the model is invalid,
some properties are not set etc.</response>
        /// <response code="401">If the user is not
authorized.</response>
        /// <response code="500">If any server error
occures.</response>
[HasPermission(Permissions.ApplicationEdit)]
[ProducesResponseType(StatusCodes.Status200OK,
Type = typeof(ApplicationDto))]
[ProducesResponseType(StatusCodes.Status400BadReq
uest)]
[ProducesResponseType(StatusCodes.Status401Unautho
rized)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
[HttpPut]
        public async Task<IActionResult>
Update(ShortApplicationDto applicationDto)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var application = await
applicationService.GetById(applicationDto.Id).Configur
eAwait(false);

            if (application is null)
            {
                return BadRequest(localizer[$"There is no
application with Id = {applicationDto.Id}."]);
            }
            application.Status = applicationDto.Status;

            try
            {
                await CheckUserRights(
                    parentId: application.ParentId,
                    providerId:
application.Workshop.ProviderId).ConfigureAwait(false
);

                var updatedApplication = await
applicationService.Update(application).ConfigureAwait(
false);
                return Ok(updatedApplication);
            }
            catch (ArgumentException ex)
            {
                return BadRequest(ex.Message);
            }
        }
        /// <summary>
        /// Delete a specific Application entity from the
database.
        /// </summary>
        /// <param name="id">Application's key.</param>
        /// <returns>A <see cref="Task{TResult}"/>
representing the result of the asynchronous
operation.</returns>
        /// <response code="204">If the entity was
```

```
successfully deleted.</response>
        /// <responce code="400">If entity with given Id
does not exist.</responce>
        /// <response code="401">If the user is not
authorized.</response>
        /// <response code="500">If any server error
occures.</response>
        [HasPermission(Permissions.SystemManagement)]
[ProducesResponseType(StatusCodes.Status204NoConte
nt)]
[ProducesResponseType(StatusCodes.Status400BadReq
uest)]
[ProducesResponseType(StatusCodes.Status500Internal
ServerError)]
[HttpDelete("{id}")]
        public async Task<IActionResult> Delete(Guid id)
        {
            try
            {
                await
applicationService.Delete(id).ConfigureAwait(false);
                return NoContent();
            }

            // TODO: update exception handling
            catch (ArgumentException ex)
            {
                return BadRequest(ex.Message);
            }
        }
        private static IEnumerable<ApplicationDto>
CreateMultiple(ApplicationApiModel
applicationApiModel)
        {
            var applications =
applicationApiModel.Children.Select(child => new
ApplicationDto
            {
                ChildId = child.Id,
                CreationTime = DateTimeOffset.UtcNow,
                WorkshopId =
applicationApiModel.WorkshopId,
            });

            return applications.ToList();
        }
        {
            if (status < 0 || status > 2)
            {
                throw new
ArgumentOutOfRangeException(nameof(status),
localizer["Status should be from 0 to 2"]);
            }
        }

        private async
Task<IEnumerable<ApplicationDto>>
GetByWorkshopId(Guid id, ApplicationFilter filter)
        {
            var workshop = await
workshopService.GetById(id).ConfigureAwait(false);
            if (workshop is null)
            {
                throw new
ArgumentException(localizer[$"There is no workshop
with Id = {id}"]);
            }
            await CheckUserRights(providerId:
workshop.ProviderId).ConfigureAwait(false);

            var applications = await
applicationService.GetAllByWorkshop(id,
filter).ConfigureAwait(false);

            return applications;
        }
        private async
Task<IEnumerable<ApplicationDto>>
GetByProviderId(Guid id, ApplicationFilter filter)
        {
            await CheckUserRights(providerId:
id).ConfigureAwait(false);

            var applications = await
applicationService.GetAllByProvider(id,
filter).ConfigureAwait(false);

            return applications;
        }
        private async Task CheckUserRights(Guid parentId
= default, Guid providerId = default)
        {
            var userId = User.FindFirst("sub")?.Value;

            bool userHasRights = true;

            if (User.IsInRole("parent"))
            {
                var parent = await
parentService.GetByUserId(userId).ConfigureAwait(fals
e);

                userHasRights = parent.Id == parentId;
            }
            else if (User.IsInRole("provider"))
            {
                var provider = await
providerService.GetByUserId(userId).ConfigureAwait(fa
lse);
                userHasRights = provider.Id == providerId;
            }

            if (!userHasRights)
            {
                throw new
ArgumentException(localizer["User has no rights to
perform operation"]);
            }
        }
    }
}
```