МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему Розробка cloud native застосунку для дошкільних навчальних закладів
Cloud native application development for preschool educational institutions

Виконав: студент денної форми навчання
спеціальності 123 – Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)

Дон Станіслав Сергійович
(прізвище, ім'я, по-батькові)

Керівник Розновець О.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Максимов О.С.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№ 11 від «10» червня 2022 р.

Завідувач кафедри

_____ _Євгеній МАЛАХОВ_
(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «___» ____ 2022 р.

Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Надія КАЗАКОВА_
(підпис) (прізвище, ініціали)

Одеса – 2022

АНОТАЦІЯ

Мета роботи – проектування та розробка хмаро-орієнтованого застосунку, призначеного для вдосконалення освітнього процесу та догляду за дітьми у закладах дошкільної освіти, а також хмарної інфраструктури для його розгортання.

Особливістю застосунка є надання вихователям функції ведення електронної звітності кожної дитини, до якої щодня заносяться позначки про самопочуття, поведінку, участь дитини у різних заходах тощо. За допомогою перегляду такої звітності батьки матимуть змогу відслідковувати інформацію про стан та успіхи своїх дітей під час перебування у дошкільному закладі освіти.

Хмарна інфраструктура базується на ресурсах постачальника хмарних послуг DigitalOcean.

Застосунок побудований за принципами мікросервісної архітектури. API розроблено з застосуванням підходу REST та мови програмування Golang в зв’язці з фреймворком Fiber. Для розробки користувацьких інтерфейсів використовується JavaScript-фреймворк Vue.js, бібліотека Vuetify та HTTP клієнт Axios. У якості СКБД використовується PostgreSQL з розширеннями pgcrypto та uuid-ossp. Для зберігання даних про сесії користувачів використовується розподілене сховище Redis.

Застосунок запускається у контейнерах та розгортується за допомогою серверу TeamCity та інструментарію Podman. У якості зворотного проксі для контейнерів використовується веб сервер Caddy.

У застосунку реалізовано розмежування повноважень користувачів на рівні СКБД та реалізований захист від несанкціонованого доступу за допомогою токенів JWT.

ABSTRACT

The purpose of this graduate work is to design and develop a cloud-native application for improvement of educational process and care of children in preschools institutions, as well as cloud infrastructure for its deployment.

The peculiarity of the application is the provision of the electronic reporting of each child function, to which daily marks about health, behavior, participation of the child in various events are entered. By reviewing such reporting, parents will be able to track the status and success of their children while in preschool education.

Cloud infrastructure is based on DigitalOcean provider.

The application is built on the principles of microservice architecture. API is developed using the REST approach and the Golang programming language in conjunction with the Fiber framework. JavaScript-framework Vue.js, Vuetify library and Axios HTTP client are used for development of user interfaces. PostgreSQL with extensions pgcrypto and uuid-ossp is used as a RDBMS. Distributed storage Redis is used for storing user session data.

The application runs in containers and is deployed using TeamCity server and Podman tools. The Caddy web server is used as a reverse proxy for containers.

The application implements the delimitation of user authority at the DBMS level and protection against unauthorized access using JWT tokens.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

API (Application Programming Interface) – прикладний інтерфейс програмування застосунків.

AMPQ (Advanced Message Queuing Protocol) – відкритий протокол прикладного рівня для проміжного програмного забезпечення, орієнтованого на взаємодію окремих компонентів через обробку між ними повідомлень.

AWS (Amazon Web Services) – комерційна публічна хмара від компанії Amazon.

CDN (Content delivery network) – регіонально розподілена мережева інфраструктура для оптимізованого доправлення контенту.

CI/CD (Continuous Integration/Continuous delivery) – низка практик, що поєднує ідеї безперервної інтеграції, доставки для забезпечення стабільності та якості розроблюваного програмного забезпечення через короткі автоматизовані ітерації.

Cloud-Native, або хмаро-орієнтований підхід – низка практик, або методологія розробки при створенні застосунків орієнтованих на виконання та розгортання у хмарному середовищу та використанню хмарних послуг.

CSRF (Cross Site Request Forgery) – міжсайтова підробка запиту. Тип веб атаки, що призводить до виконання певних зловмисних дій від імені аутентифікованого користувача на веб сторінці.

CRUD (Create, Read, Update, Delete) – чотири основні функції управління даними «створення, читання, оновлення і видалення».

GCP (Google Cloud Platform) – набір хмарних служб, що надається компанією Google.

IaaS (Infrastructure as a Service) – модель надання хмарних послуг «Інфраструктура як послуга».

JSON (JavaScript Object Notation) – текстовий формат зберігання та обміну даних.

JWT (JSON Web Token) – стандарт для створення токена доступу на основі формату JSON, що використовується для передачі аутентифікаційних даних в клієнт-серверних застосунках.

PaaS (Platform as a Service) – модель надання хмарних послуг «Платформа як послуга».

SaaS (Software as a Service) – модель надання хмарних послуг «Програмне забезпечення як послуга».

SOA (Service-Oriented Architecture) – архітектурний шаблон модульного, децентралізованого підходу розробки програмного забезпечення.

SOAP (Simple Object Access Protocol) – протокол обміну структурованими повідомленнями, що базується на форматі XML.

SSH (Secure Shell) – мережевий протокол віддаленого управління, що шифрує увесь трафік передачі даних у тунелі з’єднання із машиною.

SSL (Transport Layer Security) – криптографічний протокол, що надає можливості безпечної передачі даних.

UUID (Universally Unique Identifier) – стандарт ідентифікації, який використовується при створенні унікального ідентифікатора.

VPC (Virtual Private Cloud) – логічно ізольована віртуальна мережа (середовище) для спільно виділених хмаро-обчислювальних ресурсів, таких як VPS, у певному регіоні розташування.

VPS (Virtual Private Server), або VDS (Virtual Dedicated Server) – послуга постачальників хмарних послуг, що надають виділену віртуальну машину з певними апаратними характеристиками для користування у якості персонального або комерційного сервера.

ВСТУП

Тенденція зростання впливу освіти на сфери життя, що спостерігається в багатьох країнах світу, змінює підхід до організації освітнього процесу та ставлення до багатьох освітніх інституцій, що, в свою чергу, стосується і закладів дошкільної освіти, які стали обов'язковою ланкою загальної системи освіти. В дошкільних закладах на перше місце виходять освітні та педагогічні функції: як виховують та навчають дітей.

Діти є активними учасниками власного процесу розвитку, а не лише пасивними здобувачами результатів. Кожна дитина є унікальною особистістю зі своїми емоційними, фізичними, соціальними й когнітивними потребами, які треба враховувати. Якість процесу навчання розглядається як цілісність розвитку дитини та співпраця педагогів та сімей, що враховують погляди одне одного. Якість результатів навчання повинна враховувати переваги для дітей (добробут, залучення, соціальні навички) та їхніх батьків і суспільства [1].

Результати навчання включають моніторинг та оцінювання діяльності дітей з метою вдосконалення окремих навичок чи виявлення потенціалу у певній діяльності та для надання педагогам та батькам актуальної інформації про розвиток дитини. Моніторинг та оцінювання можна здійснювати за допомогою спостережень за поведінкою дітей, станом здоров'я, участю в різноманітних заходах, засвоєнням нового матеріалу тощо.

На сьогодні великий відсоток сучасних закладів дошкільної освіти не мають власних інформаційних систем, що в повній мірі забезпечують засоби збору, зберігання, передачі, обробки даних для оцінювання результатів навчання та досягнення мети у поліпшені освітнього процесу, або ж у догляді за дітьми, дотримуючись цих рекомендацій. Отже, задача проектування, створення та впровадження подібної інформаційної системи є актуальною. При цьому, орієнтуючись на такий універсальний фундамент, як хмарні обчислення та веб-технології, може бути створена інформаційно-освітня платформа, яка, у тому числі у рамках дистанційної дошкільної освіти,

повинна забезпечувати взаємодію всіх учасників освітнього процесу: дітей, їхніх батьків та педагогічних працівників. При цьому вона може одночасно використовуватися численною кількістю дошкільних навчальних закладів. Застосування хмарних технологій дозволить забезпечити доступність такої системи, її захищеність, гнучкість до масштабування та розподілення даних.

Мета роботи – проектування та розробка хмаро-орієнтованої інформаційної системи, призначеної для вдосконалення освітнього процесу та догляду за дітьми у закладах дошкільної освіти, а також хмарної інфраструктури для його розгортання.

Задачі, які необхідно вирішити для досягнення мети:

1) здійснити аналіз предметної області «Дошкільна освіта», виявити коло користувачів проектованої системи та визначити їх задачі;

2) проаналізувати існуючі аналогічні інформаційні системи;

3) скласти вимоги до проектованої хмаро-орієнтованого застосунку;

4) обґрунтувати вибір архітектури проектованого застосунку;

5) спроектувати модель хмарної інфраструктури для розгортання застосунку;

6) обґрунтувати вибір моделі надання хмарних послуг та постачальника хмарних послуг;

7) спроектувати інформаційну модель предметної області;

8) обрати необхідний інструментарій для розробки застосунку;

9) виконати проектування та розробку інтерфейсу презентаційного та прикладного рівня;

10) створити хмарну інфраструктуру для розгортання застосунку та розгорнути застосунок у хмарі;

11) забезпечити захищеність застосунку через розмежування повноважень з боку різних категорій користувачів у хмарній інфраструктурі, СКБД та прикладному інтерфейсі системи, а також захист від несанкціонованого доступу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД АНАЛОГІЧНИХ СИСТЕМ

1.1 Визначення користувачів проектованого застосунку та їх задач

Ідентифікація кола користувачів предметної області є суттєвою складовою проектування інформаційної системи, для того, щоб визначати функціональні задачі, які повинні виконуватися за допомогою такої системи.

Розглядаючи предметну область «Заклади дошкільної освіти» слід виділити дві основні групи осіб, що взаємодіють між собою: батьки та співробітники закладів. Вони представляють потенційну групу користувачів, які мають певні потреби для вирішення своїх функціональних задач.

З боку батьків є необхідність у відстеженні стану своєї дитини, її успіхів у різних сферах діяльності та підготовки до закладів середньої освіти. Разом з цим, батьки зацікавлені у розгляді актуальної інформації пов’язаного закладу дошкільної освіти, наприклад, запланованих подій та підтримці зворотного зв’язку із співробітниками закладу. З боку співробітника закладу є першочергова необхідність у підтримці зворотного зв’язку із батьками та інших процесів пов’язаних з роботою у закладі дошкільної освіти.

Слід також зазначити, що основними учасниками власного розвитку є діти, котрі можуть відслідковувати власні досягнення, які є мотиваційною складовою їх діяльності. Але, через специфіку користування інформаційними системами, дітей та їх батьків слід об’єднати у одну категорію користувачів.

Отже, беручи до уваги вищезазначене, у предметній області, що розглядається, можна виділити наступні основні категорії користувачів: Користувач (батьки, їх діти), Співробітник закладу, Адміністратор. Адміністратор системи є допоміжною категорією користувачів для додавання загальної інформації, а також для керування та адміністрування системою в цілому.

Задачі різних категорій користувачів з вхідними та вихідними даними описані у додатку А.

1.2 Огляд аналогічних існуючих систем

Перед початком проектування інформаційної системи також виникає задача у дослідженні ринку, порівнянні вже існуючих аналогічних систем, на основі чого можна зробити висновок щодо необхідності такої розробки та її актуальність. Нижче приведені декілька прикладів існуючих систем.

ELIIS [2] – це онлайн-рішення інформаційної системи для дошкільних навчальних закладів від естонського розробника Eliis Tarkvara OÜ. Система захищена від неавторизованих користувачів включає в себе численні функції та послуги для вихователів, батьків, директорів, державних службовців та доступна для всіх пристроїв за допомогою веб браузера або мобільного додатка, що полегшує батькам надсилання та отримання сповіщень.

ELLIS позиціонує себе, як «Все на одній платформі» та має ряд функцій, серед яких планування навчальної програми, аналіз розвитку дитини, цифровий щоденник, комунікаційні засоби, управління документами, харчуванням, робочий розклад, галерея та календар подій.

Вартість залежить від призначення системи, всього є два варіанта: для особистого використання у конкретному дошкільному навчальному закладі, або у муніципалітеті для групи таких закладів з можливістю інтеграції з іншими сервісами. Перший варіант пропонує вартість, що залежить від кількості дітей дошкільного віку у закладі, близько 0.49 євро за дитину на місяць. Другий варіант не встановлює конкретну вартість системи, тобто вона визначається після зв’язку та консультації з представником компанії. Також є варіант тестового використання системи протягом двох місяців на безоплатній основі.

Розробником системи Brightwheel [3] є однойменна американська компанія. Інформаційна система являє собою веб застосунок, який націлений на управління доглядом за дітьми у закладах дошкільної освіти або у центрах розвитку та догляду за дітьми. Компанія пропонує переносити групи закладу на свою систему для подальшої автоматизації деяких процесів цього закладу та взаємодії між ним та батьками.

Brightwheel надає ряд послуг та функцій, такі як автоматизація та цифровізація щоденних звітів про догляд за дітьми, облік звітів про успіхи дитини, а також спрощує системи виставлення рахунків центрів розвитку та автоматизування оплати.

Вартість системи не зазначена, ціна встановлюється безпосередньо після зв'язку та консультацій з представництвом компанії. Наявний варіант запланованого демонстративного тестування системи через запит з наданням контактних даних.

Загалом, можна підкреслити те, що систем для дошкільних навчальних закладів, які дозволяють проводити педагогічний моніторинг і забезпечувати взаємодію дітей, батьків та педагогічних працівників, існує доволі мало. По більшій мірі, вони націлені на певні регіони, інституції та місцеві законодавства. З цього випливає, що в деяких місцевостях немає змоги користуватися цими системами, або ж це унеможливлює певні функції та послуги, що не дає змогу повноцінно користуватись ними. Слід також зазначити, що багато з розглянутих систем є платними. Вартість визначається у зворотному зв'язку з представниками компаній, і остаточно встановлюється після консультацій з ними.

Наразі в Україні власних подібних інформаційних систем взагалі не існує, а ті, що існують поза країною, не підтримуються на регіональному, або місцевому рівні, і актуальність розробки є беззаперечною.

1.3 Вимоги до створюваного застосунку

У даній дипломній роботі необхідно розробити застосунок для закладів дошкільної освіти, який дозволить вести облік дітей, надавати функції педагогічного моніторингу для сприяння загальному розвитку та вдосконаленню окремих навичок дітей дошкільного віку і, як наслідок, підвищення якості освіти у дошкільних навчальних закладах.

Розроблюваний застосунок повинен являти собою єдину платформу для надання онлайн послуг різноманітним дошкільним навчальним закладам.

У створюваному застосунку пропонується надати вихователям функції ведення електронного щоденника кожної дитини, до якого щодня заноситимуться позначки про самопочуття, поведінку, участь дитини у різних заходах, оцінки її діяльності у різноманітних сферах освітнього процесу. За допомогою перегляду такого щоденника батьки матимуть змогу відслідковувати інформацію про стан своїх дітей під час перебування у дошкільному закладі. Позначки повинні бути відображені в такому вигляді, щоб їх могли зрозуміти не тільки дорослі, а й діти, наприклад, у вигляді емотиконів.

Для створення єдиної онлайн-платформи для дошкільних навчальних закладів пропонується застосування хмарних технологій, що має наступні переваги:

– значне зниження витрат дошкільних навчальних закладів на придбання, підтримку і модернізацію програмного забезпечення та комп’ютерної техніки;

– наявність можливості отримання доступу до застосунку в будь-який час, в будь-якому місці за допомогою комп’ютера або мобільного пристрою, підключеного до мережі Internet;

– зберігання інформації на сервері провайдера хмарних послуг, при цьому працівники дошкільних навчальних закладів мають можливість спільної роботи над документами;

– створення майданчика для взаємодії персоналу дошкільних навчальних закладів та батьків дітей, які відвідують ці заклади, з наданням різноманітних засобів підтримки комунікації.

2 ПРОЕКТУВАННЯ CLOUD NATIVE ЗАСТОСУНКУ ДЛЯ ДОШКІЛЬНИХ НАВЧАЛЬНИХ ЗАКЛАДІВ

Матеріали про ідею та проектування застосунка доповідались на дев’ятнадцятій всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології» [4].

2.1 Вибір архітектури застосунку

У дипломній роботі для реалізації поставлених задач, у якості шаблону архітектури обрано SOA – сервісно-орієнтовану архітектуру [5], що обумовлює модульний підхід при розробці застосунків, використовуючи розподілені, слабо пов’язані компоненти, які оснащені стандартизованими інтерфейсами для взаємодії між стандартизованими протоколами.

Прикладом реалізації SOA є мікросервісна архітектура [6,7], яка розподіляє застосунок як сукупність невеликих служб (мікросервісів), що працюють незалежно один від одного та містять в собі певний функціонал. Сервіси працюють, як самодостатня одиниця, але, при необхідності, можуть взаємодіяти один з одним. Взаємодія між ними, зазвичай, відбувається за рахунок використання протоколів передачі даних, таких як HTTP, що повністю сумісний із підходом REST. Також мікросервіси можуть взаємодіяти між собою, як синхронно, так й асинхронно, використовуючи інші існуючі протоколи, такі як gPRC, SOAP або AMQP.

Кожен мікросервіс відповідає за конкретну функцію програми та розгортуються вони незалежно один від одного з використанням певних автоматизованих середовищ у вигляді одного або кількох ізольованих процесів. Згодом мікросервіси об'єднуються, щоб сформувати один застосунок. Сервіси можуть бути написані на різних мовах програмування і використовувати різні технології зберігання даних.

Структура застосунку, створеного за принципами мікросервісної архітектури, наведена на рис. 2.1.

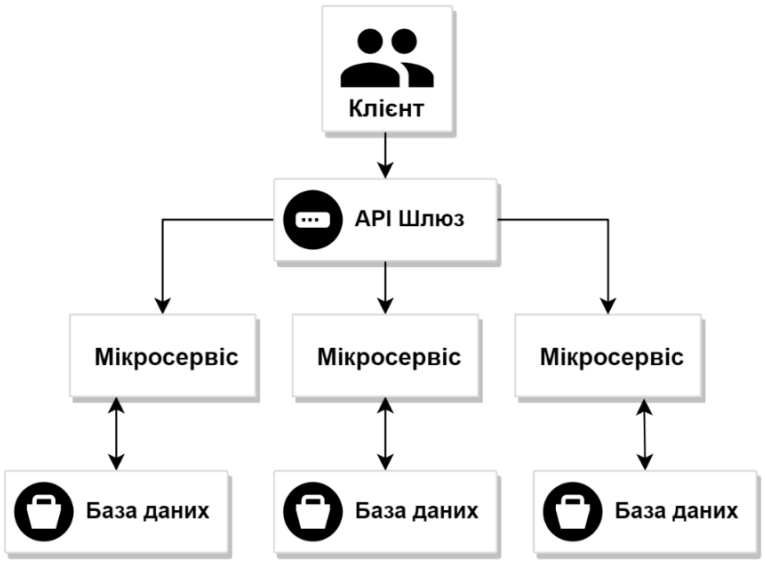

Рисунок 2.1 – Структура застосунку, створеного за принципами мікросервісної архітектури

Зазвичай архітектура мікросервісів структурована рівнями. Однак, не існує низки конкретних правил щодо того, які рівні та яка кількість мають бути у архітектурі, але існують поширені практики щодо її структуризації [8]. На рис. 2.2 зображені найпоширеніші рівні мікросервісної архітектури.

Рисунок 2.2 – Рівні мікросервісної архітектури

Отже, дана структура має схожі риси із класичною трьохрівневою архітектурою та має такі логічні рівні:

– Front end, що поєднує клієнтський застосунок та статичний контент (у випадку веб застосунку це веб браузер);

– Back end, що поєднує шлюз API (посередник між запитами клієнта та усіма прикладними інтерфейсами програмування), мікросервіси досвіду (передають запити Back end сервісам) та доменного рівня (відповідають за бізнес-логіку);

– рівень даних, що поєднує бази даних та інтеграційні інтерфейси.

На противагу мікросервісам можна виділити монолітну архітектуру, у якій навіть невеликі зміни потребують перебудови та розгортання всього застосунку [9].

У порівнянні із монолітною архітектурою можна виділити наступні переваги мікросервісної архітектури:

– незалежна масштабованість, розробка та розгортання;

– висока надійність та стійкість системи, компоненти слабо пов'язані між собою та можуть бути легко заміненими;

– організованість навколо бізнес-можливостей, що структурує певний необхідний функціонал у сервісах;

– ізольованість рівнів один від одного дозволяє швидко і простими засобами переналаштувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів;

– високий рівень безпеки через децентралізацію компонентів;

– ефективне використання апаратної частини та низькі вимоги до технічних характеристик, як наслідок зниження їхньої вартості.

Підходящим прикладом реалізації архітектури мікросервісів є контейнери, оскільки вони дозволяють зосередитися на розробці сервісів, не турбуючись про залежності. Сучасні застосунки зазвичай будуються як мікросервіси з використанням контейнерів, контейнерних оркестраторів та хмарних обчислень.

2.2 Вибір архітектурного стилю створення програмного забезпечення

Для створення програмного забезпечення застосунку для взаємодії із API проектованої системи обраний архітектурний стиль REST [10]. Дані керуються завдяки протоколу передачі даних HTTP за допомогою методів (рис. 2.4):

1) GET – отримати представлення ресурсу;

2) PUT – додати, оновити новим представленням;

3) POST – додати новий ресурс через передане представлення;

4) DELETE – видалити ресурс.

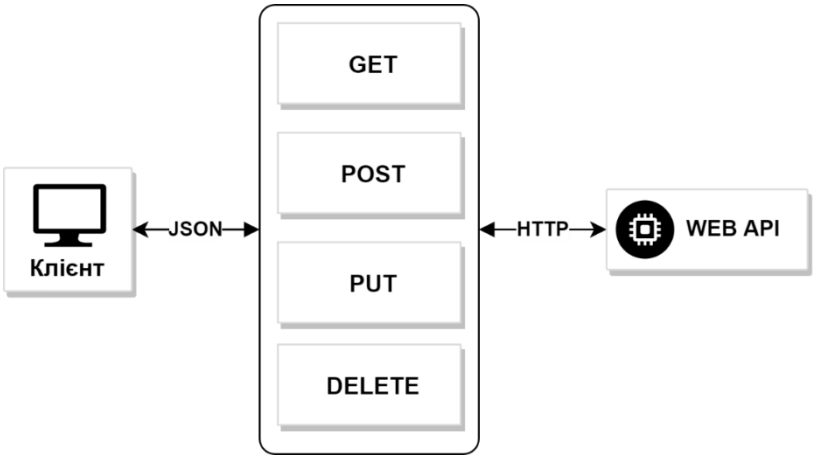

Рисунок 2.4 – Модель REST API

Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, кінцевою точкою, причому абсолютно не має значення, в якому форматі знаходяться дані за вказаною уніфікованою адресою ресурсу.

Переваги використання REST як архітектурного стилю:

– клієнт та сервер незалежні один від одного, при взаємодії між собою контекст запитів не зберігається;

– масштабованість та гнучкість розроблюваних застосунків;

– підтримка відкритих стандартів авторизації та аутентифікації;

– використання легковісного протоколу передачі даних HTTP, або відповідного до нього, що значно спрощує взаємодію клієнта та сервера.

2.4 Інформаційна модель предметної області

У якості моделі представлення даних обрано реляційну модель. Причинами з застосування реляційної моделі являються такі вимоги від проектованої бази даних, як необхідність у масштабуванні, висока структурованість даних, потреба в забезпеченні цілісності даних, забезпечення роботи транзакцій.

Спираючись на задачі за категоріями користувачів, наведені у додатку А, виконано інформаційне моделювання предметної області. В табл. 2.1 наведений опис сутностей предметної області «Дошкільні навчальні заклади».

Таблиця 2.1 – Опис сутностей, їх властивостей та обмежень цілісності

Атрибут	Призначення	Порожні значення	Унікальні значення	Обмеження цілісності
Області (Regions)				
id	Ідентифікатор області	Ні	Так	Первинний ключ
name	Назва області	Ні	Так	До 255 символів
Райони (Districts)				
id	Ідентифікатор району	Ні	Так	Первинний ключ
name	Назва району	Ні	Ні	До 255 символів
Населенні пункті (Locality)				
id	Ідентифікатор населеного пункту	Ні	Так	Первинний ключ
name	Назва населеного пункту	Ні	Ні	До 255 символів
Заклади дошкільної освіти (Kindergartens)				
id	Ідентифікатор ЗДО	Ні	Так	Первинний ключ
name	Назва ЗДО	Ні	Ні	До 255 символів
short_name	Коротка назва ЗДО	Так	Ні	До 50 символів
institution_name	Код закладу в ЄДЕБО	Ні	Так	Регулярний вираз, має дорівнювати 7 символам

Продовження таблиці 2.1

Атрибут	Призначення	Порожні значення	Унікальні значення	Обмеження цілісності
status	Статус роботи	Ні	Ні	Булеве значення
ownership	Форма власності	Ні	Ні	Значення: приватна, комунальна
email	Електронна пошта	Так	Так	Регулярний вираз, до 255 символів
tel_no	Телефон	Так	Так	Регулярний вираз, до 255 символів
website	Веб сайт	Так	Так	Регулярний вираз, до 255 символів
locality_id	Ідентифікатор населеного пункту	Ні	Ні	Зовнішній ключ
address	Фактична адреса	Ні	Так	До 510 символів
Посади (Positions)				
id	Ідентифікатор посади	Ні	Так	Первинний ключ
name	Назва посади	Ні	Так	До 255 символів
kindergarten_id	Ідентифікатор ЗДО	Ні	Ні	Зовнішній ключ
Події (Events)				
id	Ідентифікатор події	Ні	Так	Первинний ключ
title	Заголовок події	Ні	Ні	До 255 символів
description	Опис події	Ні	Ні	До 1020 символів
picture	Унікальний ідентифікатор зображення події	Так	Так	UUID
kindergarten_id	Ідентифікатор ЗДО	Ні	Ні	Зовнішній ключ
date	Дата проведення події	Ні	Ні	Значення має відповідати даті
Оцінки (Grades)				
id	Ідентифікатор оцінки	Ні	Так	Первинний ключ
children_id	Ідентифікатор дитини	Ні	Ні	Зовнішній ключ
staff_id	Ідентифікатор співробітника	Ні	Ні	Зовнішній ключ

Продовження таблиці 2.1

Атрибут	Призначення	Порожні значення	Унікальні значення	Обмеження цілісності
activity_id	Ідентифікатор діяльності	Ні	Ні	Зовнішній ключ
grade	Значення оцінки	Ні	Ні	Значення: 1-5
date	Дата	Ні	Ні	Значення має відповідати даті
Діяльність (Activities)				
id	Ідентифікатор діяльності	Ні	Так	Первинний ключ
name	Назва діяльності	Ні	Ні	До 255 символів
description	Опис	Ні	Ні	До 255 символів
kindergarten_id	Ідентифікатор ЗДО	Ні	Ні	Зовнішній ключ
Співробітники (Staffs)				
id	Ідентифікатор співробітника	Ні	Так	Первинний ключ
surname	Прізвище	Ні	Ні	До 255 символів
name	Ім’я	Ні	Ні	До 255 символів
patronymic	По батькові	Так	Ні	До 255 символів
gender	Стать	Ні	Ні	Значення: Male, Female, Other
dob	Дата народження	Ні	Ні	Значення має відповідати даті
position_id	Ідентифікатор посади	Ні	Ні	Зовнішній ключ
kindergarten_id	Ідентифікатор ЗДО	Ні	Ні	Зовнішній ключ
profile_id	Унікальний ідентифікатор профіля	Так	Так	Зовнішній ключ
Батьки (Parents)				
id	Ідентифікатор батьків	Ні	Так	Первинний ключ
surname	Прізвище	Ні	Ні	До 255 символів
name	Ім’я	Ні	Ні	До 255 символів
patronymic	По батькові	Так	Ні	До 255 символів
gender	Стать	Ні	Ні	Значення: Male, Female, Other
dob	Дата народження	Ні	Ні	Значення має відповідати даті

Продовження таблиці 2.1

Атрибут	Призначення	Порожні значення	Унікальні значення	Обмеження цілісності
position	Посада на роботі	Ні	Ні	До 255 символів
tel_no	Номер телефону	Ні	Так	Регулярний вираз, до 255 символів
address	Адреса проживання	Так	Так	До 255 символів
parent_type	Тип опікунства	Так	Ні	Значення: батько, мати
profile_id	Унікальний ідентифікатор профілю	Так	Так	Значення має відповідати структурі UUID
additional information	Додаткова інформація	Так	Ні	До 255 символів
Діти (Childrens)				
id	Ідентифікатор дитини	Ні	Так	Первинний ключ
surname	Прізвище	Ні	Ні	До 255 символів
name	Ім’я	Ні	Ні	До 255 символів
patronymic	По батькові	Так	Ні	До 255 символів
gender	Стать	Ні	Ні	Значення: Male, Female, Other
dob	Дата народження	Ні	Ні	Значення має відповідати даті
entry_date	Дата входження	Ні	Ні	Значення має відповідати даті
end_date	Дата закінчення	Ні	Ні	Значення має відповідати даті
picture	Унікальний ідентифікатор зображення дитини	Так	Так	Значення має відповідати структурі UUID
additional information	Додаткова інформація	Так	Ні	До 512 символів
Групи (Groups)				
id	Ідентифікатор групи	Ні	Так	Первинний ключ
name	Назва групи	Ні	Ні	До 255 символів
kindergarten_id	Ідентифікатор ЗДО	Ні	Так	Зовнішній ключ
group_type	Тип групи	Так	Ні	До 255 символів

Продовження таблиці 2.1

Атрибут	Призначення	Порожні значення	Унікальні значення	Обмеження цілісності
Документи (Documents)				
id	Ідентифікатор документу	Ні	Так	Первинний ключ
document_id	Унікальний ідентифікатор документу	Ні	Ні	Значення має відповідати структурі UUID
staff_id	Ідентифікатор співробітника	Ні	Ні	Зовнішній ключ
kindergarten_id	Ідентифікатор ЗДО	Ні	Ні	Зовнішній ключ
description	Опис	Ні	Ні	До 255 символів
Групи співробітників (Staff Groups)				
staff_id	Ідентифікатор співробітника	Ні	Так	Зовнішній ключ
group_id	Ідентифікатор групи	Ні	Так	Зовнішній ключ
Групи дітей (Children Groups)				
children_id	Ідентифікатор дитини	Ні	Так	Зовнішній ключ
group_id	Ідентифікатор групи	Ні	Так	Зовнішній ключ
Сім’я (Family)				
parent_id	Ідентифікатор батьків	Ні	Так	Зовнішній ключ
children_id	Ідентифікатор дитини	Ні	Так	Зовнішній ключ
Користувачі (Users)				
id	Унікальний ідентифікатор користувача	Ні	Так	Первинний ключ, UUID
profile_id	Унікальний ідентифікатор профілю	Ні	Так	Зовнішній ключ, Значення має відповідати структурі UUID
email	Електронна пошта	Ні	Так	Регулярний вираз перевірки електронної пошти, до 127 символів
username	Псевдонім користувача	Ні	Так	До 32 символів

Продовження таблиці 2.1

Атрибут	Призначення	Порожні значення	Унікальні значення	Обмеження цілісності
password	Пароль користувача			До 255 символів, повинно бути більше, або дорівнювати 8 символам
category	Категорія користувача	Ні	Ні	Значення: Користувач, Співробітник, Адміністратор
picture	Ідентифікатор зображення профілю	Так	Так	UUID
created_at	Мітка часу при створенні	Ні	Ні	Значення повинно відповідати мітки часу
updated_at	Мітка часу при оновленні	Ні	Ні	Значення повинно відповідати мітки часу

Між таблицями реалізовано 2 типи зв’язків:

– «один-до-одного»;

– «один-до-багатьох».

Зв'язок між сутностями «Користувачі» та «Співробітники». Один співробітник має лише один обліковий запис, з іншого боку один обліковий запис відповідає одному співробітнику. Таким чином, між цими сутностями існує зв'язок «один-до-одного», який формалізований шляхом додавання ідентифікатора співробітника у сутність «Користувачі» як зовнішнього ключа. Зв’язки «один-до-одного» наявні також між сутностями «Користувачі» та «Батьки».

Зв'язок між сутностями «Заклад дошкільної освіти» та «Групи». У одному закладі може бути одна, або декілька груп; з іншого боку одна група відноситься тільки до одного закладу дошкільної освіти. Отже, між ними існує зв'язок «один-до-багатьох», який формалізований шляхом додавання ідентифікатору закладу (первинного ключа) в сутність «Групи» як зовнішнього ключа.

Зв'язки «один-до-багатьох» наявні також між сутностями:

– «Заклади дошкільної освіти» та «Групи»;

– «Заклади дошкільної освіти» та «Посади»;

– «Заклади дошкільної освіти» та «Співробітники»;

– «Заклади дошкільної освіти» та «Події»;

– «Заклади дошкільної освіти» та «Документи»;

– «Області» та «Райони»;

– «Райони» та «Населенні пункти»;

– «Населенні» та «Заклади дошкільної освіти»;

– «Види діяльності» та «Оцінки»;

– «Співробітники» та «Оцінки»;

– «Співробітники» та «Документи».

Зв’язок «багато-до-багатьох» виникає між сутностями «Батьки» та «Діти»: один з батьків може мати одну, або декількох дітей, з іншого боку одна дитина може мати одного, або декілька батьків. Для формалізації даного зв’язку створюється допоміжна сутність «Сім’я», яка містить первинні ключі двох вище зазначених сутностей у якості зовнішніх. Ця сукупність атрибутів є первинним ключем сутності «Сім’я».

Зв'язки «один-до-багатьох» наявні також між сутностями:

– «Групи» та «Діти»;

– «Співробітники» та «Діти».

ER-діаграма створеної бази даних наведена у додатку Б.

3 ПРОЕКТУВАННЯ ХМАРНОЇ ІНФРАСТРУКТУРИ ДЛЯ CLOUD NATIVE ЗАСТОСУНКУ ДЛЯ ДОШКІЛЬНИХ НАВЧАЛЬНИХ ЗАКЛАДІВ

3.1 Підхід Cloud Native при розробці застосунків

Cloud native [11], або хмаро-орієнтованість – це підхід до створення та виконання застосунків, що використовують переваги хмарних систем. Хмаро-орієнтована розробка включає в себе наступні концепції.

1) DevOps [12] – набір практик, який поєднує розробку програмного забезпечення та IT-операції, що має на меті скоротити життєвий цикл розробки програмного забезпечення і забезпечити безперервну його поставку.

2) CI/CD [13] – процес постійного і безперервного додавання різних змін та оновлень програмне забезпечення, що об’єднує його розробку, тестування та розгортання, в тому числі за допомогою автоматизації.

3) Архітектура мікросервісів.

4) Контейнеризація [14] – програмна віртуалізація на рівні ОС, яка забезпечує запуск застосунку та необхідний йому мінімум ресурсів в деякому стандартизованому просторі (контейнері). Контейнери ізольовані один від одного та кожен має унікальну файлову систему з можливістю запису і квоту ресурсів.

Беручи до уваги порівняння хмаро-орієнтованих та традиційних застосунків, можна зробити висновок, що саме cloud native підхід є найбільш актуальним для розробки сучасних веб-застосунків завдяки тому, що хмаро-орієнтовані застосунки є:

1) незалежними завдяки використанню архітектури мікросервісів;

2) передбачуваними, тобто відповідають «контрактам», розробленим для забезпечення максимальної стійкості до зовнішніх впливів за рахунок передбачуваної поведінки;

3) незалежними від потужностей, тобто хмарна платформа автоматизує ініціалізацію і налаштування інфраструктури, динамічно розподіляючи і перерозподіляючи ресурси під час розгортання і роботи відповідно до потреб програмного забезпечення;

4) не прив'язаними до ОС, тобто замість налаштовування і обслуговування ОС, розробник зосереджується на створенні програмного забезпечення.

3.2 Вибір моделі надання хмарних послуг

Хмарні обчислення – це модель, яка дозволяє розробникам ефективно, швидко та з мінімальними витратами вирішувати ряд задач, пов’язаних з розробкою та розгортанням різноманітних систем. Хмарні послуги – це модель сумісного споживання ІТ ресурсів, заснована на використанні в оренді, а не на володінні. Виділяються декілька моделей надання послуг за допомогою хмари [15], основними серед яких є «Програма як послуга» (SaaS), «Платформа як послуга» (PaaS) та «Інфраструктура як послуга» (IaaS). Порівняння цих моделей у розподілі ресурсів між споживачем та постачальником хмарних послуг наведено у табл. 3.1.

Таблиця 3.1 Порівняння традиційної моделі застосунку та IaaS, PaaS, SaaS

Ресурс, яким самостійно керує споживач	Традиційна модель	IaaS	Paas	SaaS
Прикладний рівень ПЗ	+	+	+	–
Рівень даних	+	+	+	–
Середовище виконання	+	+	–	–
Проміжне ПЗ	+	+	–	–
Операційна система	+	+	–	–
Віртуалізація	+	–	–	–
Сервери	+	–	–	–
Сховище	+	–	–	–
Мережа	+	–	–	–

При проектуванні та створенні сучасних веб-систем часто виникає необхідність у повному контролі над ресурсами розроблюваної системи, їх взаємодії між собою, стабільній роботі та здатністю систем до масштабування. Серед вищезазначених моделей, найпридатнішою моделлю для створення застосунку для дошкільних навчальних закладів, що створюється у рамках даної дипломної роботи, є IaaS. В межах цієї моделі споживачу хмарних послуг

надається прикладний інтерфейс високого рівня для керування засобами обчислювальних ресурсів, на основі яких споживач розроблює та розгортає програмне забезпечення. Також завдяки наданому прикладному інтерфейсу вирішуються задачі низького рівня, такі як регулювання мережевої інфраструктури та керування операційною системою.

3.3 Вибір постачальника хмарних послуг

Використовуючи хмари, замовники інформаційних технологій можуть істотно знизити капітальні витрати – на побудову центрів обробки даних, закупівлю серверного та мережевого обладнання, апаратного забезпечення або програмних рішень щодо забезпечення безперервності та працездатності програмного забезпечення, так як всі ці витрати надаються постачальником хмарних послуг. Найпоширенішими постачальниками хмарних послуг на даний момент є Amazon Web Services, Google Cloud Platform, Microsoft Azure та DigitalOcean. У табл. 3.2 приведені характеристики цих постачальників [16].

Таблиця 3.2 – Характеристики постачальників хмарних послуг

Характеристика	Amazon Web Services	Google Cloud Platform	Microsoft Azure	Digital Ocean
Щогодинна вартість	+	+	+	+
ПЗ для управління	Закрите	Закрите	Закрите	–
Командний термінал	–	+	–	+
Масштабування	+	+	+	+
Наявність API	+	+	+	+
AWS-сумісний API	+	–	–	+
NVME SSD	+	–	–	+
Сторонні зображення	+	+	–	+
Гіпервізор	Xen	KVM	Hyper-V	KVM
Блокове сховище	+	+	+	+
Призначені IP	+	–	+	+
Підтримка SMTP	+	–	+	+
Вартість вимкненої машини	Частково	Тільки за сховище	Повна	Частково

Продовження таблиці 3.2

Характеристика	Amazon Web Services	Google Cloud Platform	Microsoft Azure	Digital Ocean
Вибір вендору процесора	–	–	–	+
Тип балансиру навантаження	Програмний	Програмний	DNS	Програмний

За результатом порівняння характеристик, як постачальника хмарних послуг обрано DigitalOcean через ряд послуг, що ним надаються, зокрема:

1) доступність у використанні та менша вартість наданих послуг до виділених обчислювальних ресурсів;

2) націленість на активне постачання UNIX-подібних операційних систем з мінімальними проблемами у роботі;

3) необхідні послуги та функції без зайвої складності у реалізації;

4) професійна та наглядна документація з великою спільнотою;

5) націленість на активне використання та розвиток проектів з відкритим вихідним кодом;

6) надання зовнішніх сховищ різних типів;

7) надання послуги кластера Kubernetes для контейнерів з автоматичним масштабуванням та безкоштовною панеллю керування;

8) послуга балансиру навантаження до виділених віртуальних машин, або до кластеру Kubernetes з підтримкою HTTP/2 протоколу для зовнішнього доступу та інтегрованого сервісу Lets Encrypt для генерації та надання SSL сертифікатів;

9) послуга моніторингу хмарної інфраструктури з відслідковуванням стану мережі, навантаження віртуальних машин та їх внутрішніх ресурсів.

3.4 Проектування моделі хмарної інфраструктури

Перед створенням хмарної інфраструктури виникає необхідність у попередньому проектуванні моделі хмарної інфраструктури, виділені її складових.

Усі складові хмаро-орієнтованого застосунку мають бути розміщені у приватній мережі задля підвищення рівня безпеки.

На рис. 3.1 наведена модель хмарної інфраструктури, яку пропонується створити для розгортання застосунку для дошкільних навчальних закладів.

Нижче описані умовні позначення компонентів моделі, зображеної на рис. 3.1.

Cloud PostgreSQL – хмарний кластер системи керування базами даних PostgreSQL, що використовується для збереження загальних даних застосунку.

Cloud Redis – хмарний кластер розподіленого сховища Redis, що використовується для зберігання сесій користувачів.

Container Registry – репозиторій контейнерів, де зберігаються образи контейнерів для розгортання окремих компонентів застосунку.

Application – виділений віртуальний сервер для розгортання компонентів застосунку, налаштування та забезпечення їх роботи.

APIs – контейнери, що відповідають за роботу API.

Front End – контейнер, що відповідає за надання користувальницького інтерфейсу.

Storage – зовнішнє хмарне сховище для зберігання різних типів файлів користувачів, або файлів, що пов’язані із роботою застосунку.

CDN – мережа доправлення контенту, такого як статичні файли, або файли конфігурації для відтворення користувальницького інтерфейсу.

Management – виділена віртуальна машина для керування інфраструктурою застосунку, виконання процесу безперервної інтеграції та безперервного розгортання (CI/CD).

Cloud Load Balancing – балансир навантаження, що є публічною точкою доступу користувачів для їх взаємодії із інфраструктурою застосунку.

Monitoring – процес моніторингу інфраструктури за її метриками через внутрішні, або зовнішні послуги.

Users – користувачі, що взаємодіють із хмарним застосунком.

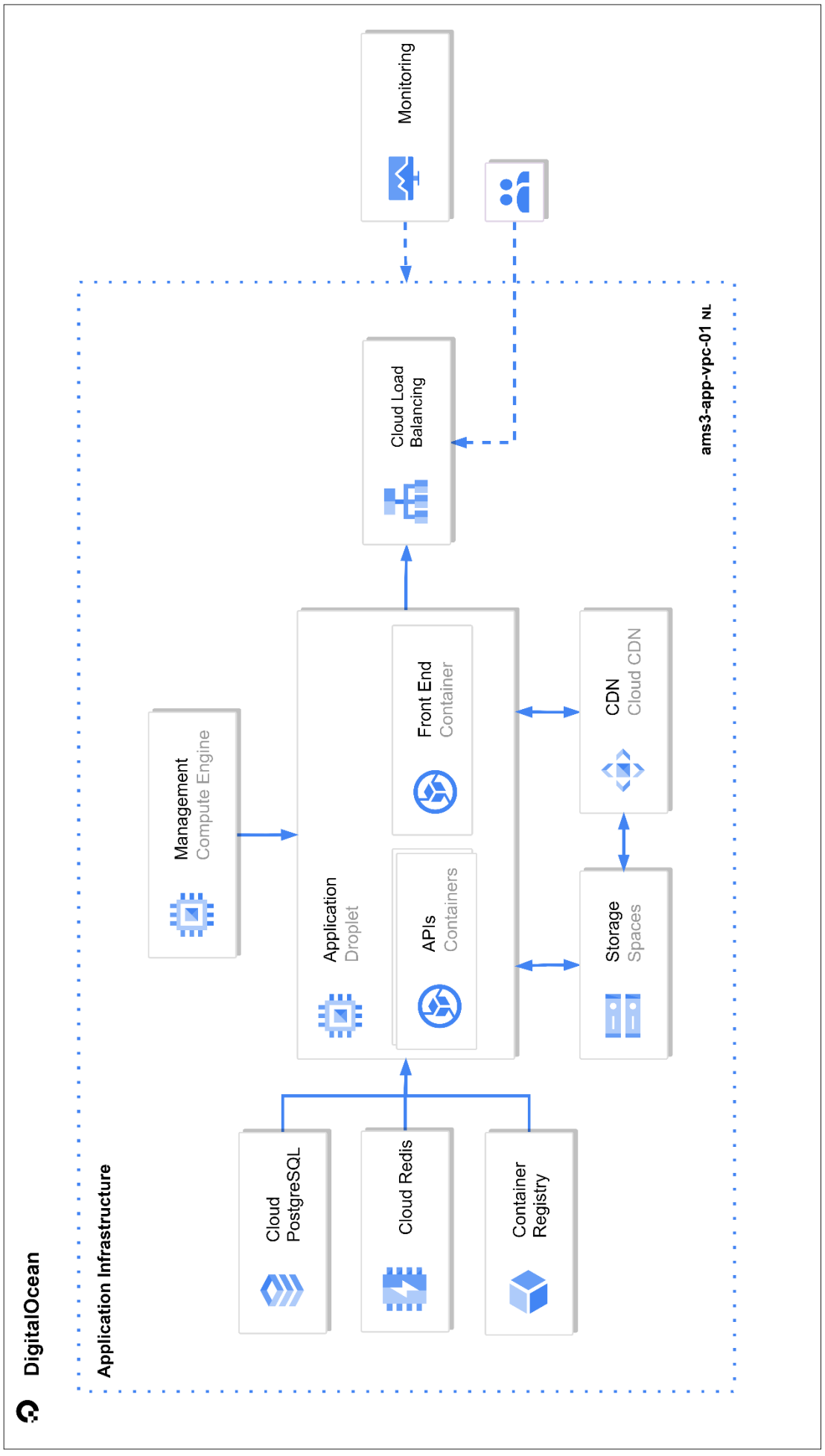

Рисунок 3.1 – Спроектована модель хмарної інфраструктури

4 РОЗРОБКА CLOUD NATIVE ЗАСТОСУНКУ ДЛЯ ДОШКІЛЬНИХ НАВЧАЛЬНИХ ЗАКЛАДІВ

4.1 Стек технологій застосунку

4.1.1 Засоби реалізації інтерфейсу програмування

Для реалізації інтерфейсу програмування застосунку (API) використовується Golang [17] – статично типізована, компільована мова програмування, що має наступні переваги над іншими:

– сильна статична типізація;

– швидкий час компіляції;

– вбудовані рівночасні обчислення;

– підтримка загальних типів;

– вбудований сміттєзбирач у оперативній пам’яті;

– платформна незалежність;

– компіляція програмного забезпечення в єдиний виконавчий файл, що зручно при розгортанні застосунку та при керуванні версіями.

На даний момент, Golang позиціонується як найбільш розвинена мова програмування у розробці хмаро-орієнтованих застосунків через велику кількість програмних пакетів із спільноти та інтеграцій з будь-яким програмним забезпеченням.

Список програмних пакетів, що використовуються при створенні API, наведений у таблиці 4.1.

Таблиця 4.1 – Програмні пакети, що використовуються при створенні API

Назва пакету	Призначення
gofiber/fiber	Веб фреймворк
golang-jwt/jwt	Бібліотека для роботи із JWT
google/uuid	Утиліта для роботи із UUID

Продовження таблиці 4.1

Назва пакету	Призначення
go-gorm/gorm	ORM бібліотека
arsmn/fiber-swagger	Проміжне програмне забезпечення для генерування документації (кінцевих точок, їх тип)
jmoiron/sqlx	Загальний інтерфейс навколо баз даних SQL
joho/godotenv	Утиліта зчитування змінних середовища
gofiber/jwt	Утиліта для роботи із JWT
go-redis/redis	Драйвер для роботи з Redis
stretchr/testify	Тестування програмного коду
cosmtrek /air	Утиліта для перезавантаження серверу у режимі реального часу

4.1.2 Засоби реалізації користувальницького інтерфейсу

Структура користувальницького інтерфейсу базується на моделі MPA (Multi Page Application) [18], що складається із динамічних сторінок, наповнення яких залежить від вхідних даних. На відміну від односторінкових застосунків, динамічні веб сторінки краще підлягають оптимізації та краще масштабуються.

Для розробки користувальницького інтерфейсу обрано Javascript-фреймворк Vue.js [19] та бібліотеку Material-компонентів Vuetify. Vue.js – це прогресивний JavaScript-фреймворк, що має наступні переваги:

– підтримка декларативно зв'язувати відтворення Virtual DOM із основними екземплярами даних в Vue;

– підтримка вбудованих шаблонів;

– ненав'язлива система відгуку компонентів, яка використовує прості об'єкти JavaScript та оптимізований процес відтворення даних;

– екосистема фреймворку Vue.

У якості інструменту HTTP клієнта обрано утиліту Axios, завдяки їй здійснюються HTTP запити до інтерфейсу програмування застосунків. Функції, що підтримуються Axios:

1) перехват запиту та вихідної відповіді;

2) скасування запитів;

3) автоматичне перетворення у JSON формат;

4) підтримка захисту від XSRF зі сторони клієнта.

4.1.3 Система керування базами даних

Серед багатьох реляційних систем керування базами даних, таких як MariaDB, Oracle Database, Microsoft SQL Server та багатьох інших вирішено використовувати PostgreSQL [20]. Серед переваг використання PostgreSQL можна виділити:

– підтримка реляційних (SQL) та нереляційних (JSON) запитів;

– програмне забезпечення з відкритим вихідним кодом;

– вільна ліцензія, яка надає дозвіл використання системи у особистих, комерційних або навчальних рішеннях;

– масштабування баз даних.

Також серед переваг PostgreSQL можна зазначити наявність розширень pgcrypto та uuid-ossp, що підтримують відповідно хеш-функції та унікальні типи, такі як Universally unique identifier (UUID), використовувані у даній роботі.

4.1.4 Стандарт аутентифікації та авторизації

У якості стандарту аутентифікації та авторизації використовується JSON Web Token (JWT), що представляє собою відкритий стандарт, який визначає компактний та самодостатній спосіб безпечного передавання інформації між клієнтом та сервером у об'єкті JSON. Ця інформація може бути перевірена і довірена, оскільки вона має цифровий підпис. Токени можуть бути підписані за допомогою секрету (з алгоритмом HMAC) або пари відкритих/приватних ключів за допомогою RSA або ECDSA [21].

4.1.5 Інструмент контейнеризації

На даний момент, найпоширенішими платформами контейнеризації є Docker, LXC, Podman, CRI-O. У роботі використовується Podman [22], який представляє відкритий інструмент віртуалізації рівня операційної системи на базі ядра Linux. Podman призначений для того, щоб виконувати, будувати, розгортати застосунки, використовуючи Open Containers Initiative (OCI) контейнери. Подібно до інших поширених платформ контейнеризації, Podman покладається на середовище виконання контейнерів, сумісне з OCI, щоб взаємодіяти з операційною системою та створювати запущені контейнери.

Переваги використання Podman на противагу іншим контейнерним платформам:

– менше залежності до іншого програмного забезпечення;

– більш захищене середовище контейнерів;

– не використовує додаткових фонових процесів;

– модульний підхід у виконанні;

– повністю сумісний із командами Docker.

4.1.6 Проксі сервер

У якості проксі серверу використовується Caddy, який є сучасною альтернативною Apache та Nginx. Порівняно з ними, Caddy має багато переваг:

– автоматичне використання HTTPS протоколу при наявності домену;

– адаптери для підтримки файлів конфігурації інших веб серверів;

– підтримка декількох текстових форматів для конфігурації веб серверу;

– власний тип розширення конфігураційного файлу.

Caddy в даній системі виступає у ролі зворотного проксі для працюючих контейнерів, щоб знизити навантаження на сервер через балансування поміж ними. Це значно прискорює роботу та дає сервісам можливість більш ефективно надсилати запити один одному.

4.1.7 Система управління побудовою застосунків та неперервної інтеграції

Серед найпоширеніших інструментів для CI/CD, таких як Jenkins, TeamCity, Gitlab CI, Travis вирішено використовувати TeamCity [23].

TeamCity підтримується багатьма операційними системами та платформами (Linux, MacOS, Windows, Docker), маючи ряд унікальних функцій, таких як деталізовані звіти виконаних задач, миттєвий зворотній зв'язок з помилками тестування та параметри багаторазового використання, що дає змогу не писати надлишковий код. Крім того, TeamCity підтримує багато сценарних мов та утиліт, таких як Bash, PowerShell, Docker, Python тощо. При відсутності підтримки інтеграції з іншими інструментами, або сервісами, можна використовувати зовнішні додатки, шляхом підключення їх через спеціальний майданчик JetBrains Marketplace.

4.2 Створення бази даних

У цьому підрозділі наведені приклади створення об’єктів бази даних, таких як таблиці (на прикладі таблиці kindergartens), представлення, тригери та перелічувальні типи. Повний список запитів для створення бази даних наведений у додатку В.

Для визначення типу даних одного із стовпців таблиці kindergartens необхідно створити тип ownership_type, запит на створення якого наведено у лістингу 4.1. Створення типу відбувається за допомогою запиту CREATE TYPE, далі вказується назва типу та вираз приналежності допустимих значень до перелічувального типу ENUM.

```
CREATE TYPE "OWNERSHIP_TYPE" AS ENUM ('ПРИВАТНА', 'КОМУНАЛЬНА',
'МУНІЦИПАЛЬНА', 'ДЕРЖАВНА', 'КОЛЕКТИВНА');
```

Лістинг 4.1 – Створення перелічувального типу

Для створення таблиці kindergartens використовується запит, наведений у лістингу 4.2.

```
CREATE TABLE "KINDERGARTENS" (ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  SHORT_NAME VARCHAR(50),
  INSTITUTION_CODE INT NOT NULL,
  STATUS BOOL NOT NULL DEFAULT(TRUE),
  OWNERSHIP OWNERSHIP_TYPE,
  EMAIL VARCHAR(255) DEFAULT('') CHECK(EMAIL ~
\'^[A-ZA-Z0-9._%-]+@[A-ZA-Z0-9.-]+[.][A-ZA-Z]+$' OR EMAIL ~ ''),
  TEL_NO VARCHAR(255) DEFAULT(''),
  WEBSITE VARCHAR(255) DEFAULT('') CHECK (WEBSITE ~
'.*?\.[A-Z]+\/' OR WEBSITE ~ ''),
  LOCALITY_ID INT NOT NULL,
  ADDRESS VARCHAR(255) NOT NULL,
  UNIQUE(ADDRESS, EMAIL, WEBSITE, TEL_NO, INSTITUTION_CODE));
```

Лістинг 4.2 – Створення таблиці «kindergartens»

Створення таблиці відбувається за допомогою запиту CREATE TABLE, після чого вказується назва таблиці, а далі кожному полю зіставляються тип даних та обмеження цілісності. Поле ID є первинним ключем та має тип SERIAL, який дозволяє при кожному додаванні даних збільшувати цілочисельні значення цього стовпця на одиницю. Це спрощує процес вставки даних у таблиці, без необхідності визначення значення ідентифікатору. Поле ownership має користувальницький перелічувальний тип ownership_type.

Вираз NOT NULL при визначенні полів означає, що вони не можуть містити порожні значення. Вираз DEFAULT (TRUE) при визначенні поля status вказує, що дане поле, за замовчуванням, має булеве значення логічного «так», відносно позначаючи, що заклад на даний момент працюючий. Вираз CHECK при оголошені поля email вказує на регулярний вираз для коректного визначення електронної пошти, що має зіставлятись з новими введеними значеннями у цьому полі. Вираз UNIQUE вказує на те, що значення комбінації зазначених у дужках стовпців полів мають бути унікальними.

При додаванні, або оновленні облікових даних користувача у таблицю USERS викликається тригер ENCRYPT_USER_DATA, що генерує хеш значення у поле PASSWORD. Створення цього тригера та тригерної функції наведено у лістингу 4.3.

```
CREATE OR REPLACE FUNCTION ENCRYPT_PASSWORD() RETURNS TRIGGER AS
$FUNC$
BEGIN
  NEW.PASSWORD := CRYPT(NEW.PASSWORD, GEN_SALT('BF'));
  RETURN NEW;
END
$FUNC$ LANGUAGE PLPGSQL;

CREATE TRIGGER ENCRYPT_USER_DATA
  BEFORE INSERT OR UPDATE ON USERS
  FOR EACH ROW EXECUTE PROCEDURE ENCRYPT_PASSWORD();
```

Лістинг 4.3 – Створення тригера та тригерної функції

Створення тригерної функції відбувається за допомогою команди CREATE FUNCTION, далі вказується назва тригерної функції, а вже після тип, котрий вона повертає. Всередині тригерної функції описується інструкції для маніпулювання даними таблиці, котрі мають змінитися після події, що сприятиме виклику цієї функції. У цьому випадку, тригерна функція шифрує дані поля PASSWORD, після того, як нові дані з цим полем надійдуть у таблицю.

Останнім об’єктом бази даних для розгляду є представлення. Приклад створення представлення повної інформації родини наведено у лістингу 4.4.

Створення представлення відбувається за допомогою команди CREATE VIEW, далі вказується назва представлення та назва полів таблиць, котрі будуть міститись у даному представленні. Для того, щоб використовувати поля з інших таблиць використовується команда JOIN, щоб об’єднати таблиці одна до одної завдяки зв’язкам, котрі вони мають через первинні та зовнішні ключі. У кінці представлення, поля таблиць вказуються у агрегатних функціях GROUP BY та ORDER BY, де остання є не обов’язковою.

```
CREATE VIEW FAMILY_VIEW AS
SELECT P.PROFILE_ID, P.ID AS "PARENT - ID", P.SURNAME AS "PARENT
- SURNAME", P.NAME AS "PARENT - NAME", P.PATRONYMIC AS "PARENT -
PATRONYMIC", P.PARENT_TYPE AS "PARENT - TYPE", CH.ID AS "CHILDREN
- ID", CH.SURNAME AS "CHILDREN - SURNAME", CH.NAME AS "CHILDREN -
NAME", CH.PATRONYMIC AS "CHILDREN - PATRONYMIC",  CH.ENTRY_DATE
AS "CHILDREN - ENTRY DATE",  CH.END_DATE AS "CHILDREN - END DATE"
FROM FAMILY F
  INNER JOIN PARENTS P ON P.ID = F.PARENT_ID
  INNER JOIN CHILDRENS CH ON CH.ID = F.CHILDREN_ID
GROUP BY P.ID, P.SURNAME, P.NAME, P.PATRONYMIC, P.PARENT_TYPE,
  P.PROFILE_ID, CH.ID, CH.SURNAME, CH.NAME, CH.PATRONYMIC
ORDER BY P.PROFILE_ID;
```

Лістинг 4.4 – Створення представлення

4.3 Реалізація програмного інтерфейсу застосунку

Інтерфейс програмування (API) застосунку розміщено у локальному репозиторії за допомогою системи управління версіями Git. Репозиторій складається з ієрархії директорій, що наведена на рис. 4.5.

```
+---app
|   +---controllers
|   +---models
|   \---queries
+---docs
+---pkg
|   +---configs
|   +---middleware
|   +---repository
|   +---routes
|   \---utils
+---platform
|   +---cache
|   +---database
|   \---migrations
\---build
```

Рисунок 4.1 – Ієрархія директорій у репозиторії прикладного інтерфейсу застосунку

Директорія app містить лише бізнес-логіку застосунку, відокремлюючи себе від рівня платформи, та складається із трьох піддиректорій:

– controllers містить функціональні контролери, що використовуються у описі методів при маршрутизації;

– models містить опис бізнес-моделей та методів;

– queries містить опис запитів до бази даних.

Директорія docs містить документацію API, яка автоматично створюється за рахунок файлів налаштувань, що містяться у ній.

Директорія pkg містить певну функціональність, специфічної до проекту. У цій директорії міститься весь код для підтримки роботи застосунку. Директорія складається із п’яти піддиректорій:

– configs містить конфігураційні файли;

– middleware містить проміжне програмне забезпечення;

– repository містить постійні значення, такі як права доступу для певних категорій авторизованих користувачів;

– routes містить маршрутизацію прикладного інтерфейсу застосунку, маршрути якої поділяються на захищені та не захищені;

– utils містить утиліти, або функції для запуску сервера, перевірки помилок тощо.

Директорія platform містить усю логіку рівня платформи, наприклад, налаштування бази даних або кешу сервера і зберігання міграцій. Директорія містить три піддиректорії:

– cache містить функції налаштування для підключення Redis до проекту;

– database містить функції налаштування для підключення бази даних до проекту;

– migrations містить та зберігає міграційні файли.

Взаємодія клієнта та API застосунка представлена на рис. 4.2. Під кожним із складових рівнів проілюстровано його прив'язку до списку створених програмних класів, що відтворюють їх. Із складових рівнів виділяються:

1) клієнт – веб браузер, що надсилає запит до інтерфейсу;

2) кінцеві точки – глобальна одиниця маршрутизації;

3) контролери – класи, що виконують основну бізнес логіку;

4) моделі – об’єктно-реляційне відображення до таблиць у базі даних;

5) бази даних – сукупність даних інформаційної моделі.

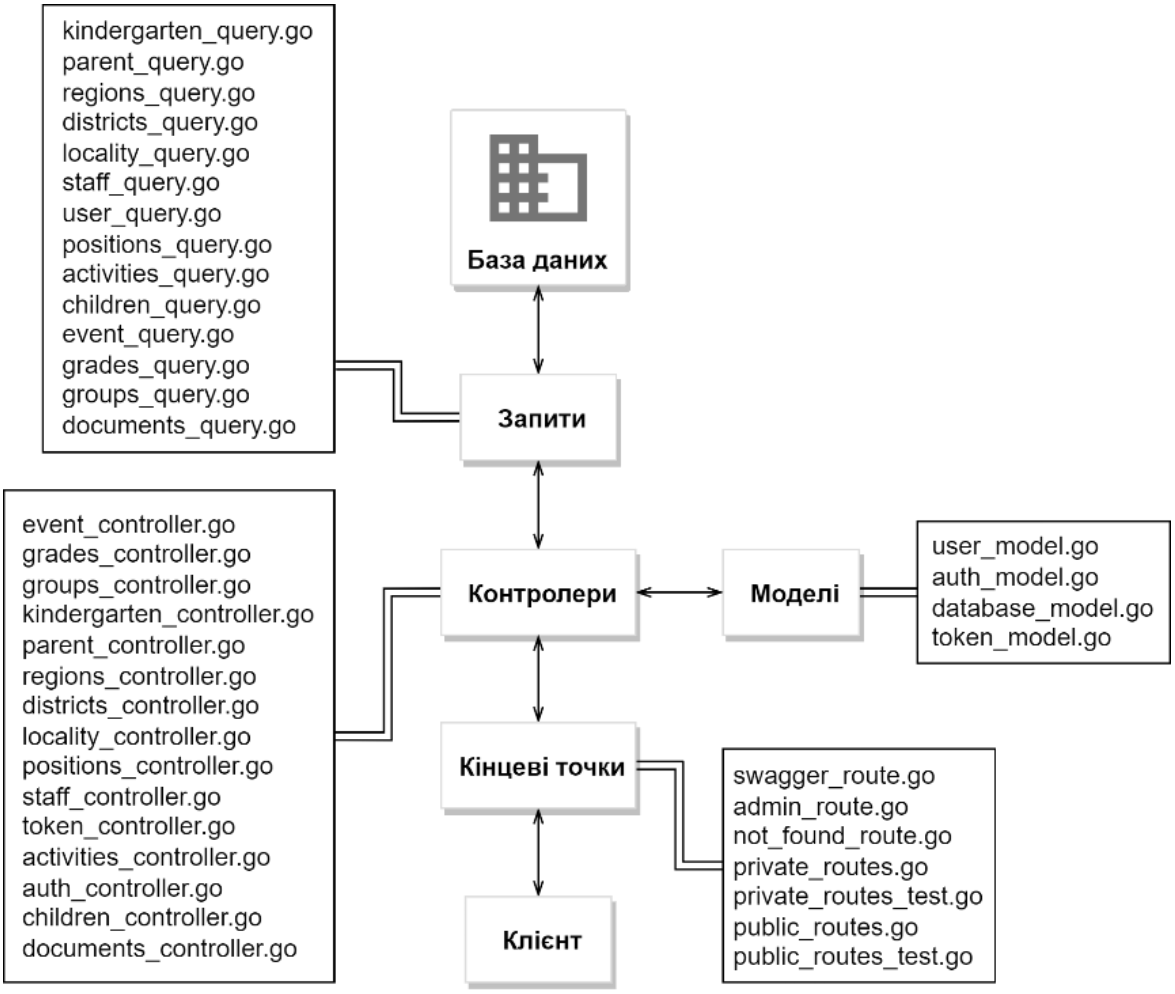

Рисунок 4.2 – Взаємодія клієнта між рівнями API та базою даних, класи рівнів, що їх реалізують

У лістингу 4.5 наведені приклади взаємодії між клієнтом та бази даних на рівні програмної реалізації до схеми, зображеної на рис. 4.2. Так як реалізація інших класів, відповідна до реалізації інших, в тій, чи іншій мірі, то розглядається лише одна сутність «Заклади дошкільної освіти».

```
api.Get("/kindergarten", controllers.GetKindergartens)
api.Get("/kindergarten/:id", controllers.GetKindergarten)
api.Put("/kindergarten", middleware.JWTProtected(),
controllers.UpdateKindergarten)
api.Delete("/kindergarten", middleware.JWTProtected(),
controllers.DeleteKindergarten)
```

Лістинг 4.5 – Маршрути до моделі kindergarten

При надходженні запиту від клієнта до інтерфейсу програмування застосунку, він визначає його структуру та вхідні параметри. За вхідні параметри вважаються дані до бази даних, токени тощо. Через структуру запиту, інтерфейс відокремлює з нього кінцеву точку та звіряє її зі списком усіх маршрутів (захищених, або не захищених). Після того, як кінцева точка збігається із певним маршрутом, інтерфейс додатково зчитує, при наявності, автентичність прикріпленого токену та його права доступу до цього маршруту (лістинг 4.6). Якщо запит успішно проходить авторизацію, інтерфейс перенаправляє його до відповідного контролеру (лістинг 4.7).

```
now := time.Now().Unix()
  claims, err := utils.ExtractTokenMetadata(c)
  if err != nil {
    return
c.Status(fiber.StatusInternalServerError).JSON(fiber.Map{
      "error": true, "msg": err.Error(), })  }
  expires := claims.Expires
  if now > expires {
    return c.Status(fiber.StatusUnauthorized).JSON(fiber.Map{
      "error": true, "msg": "unauthorized, check expiration time
of your token",  })  }
  credential :=
claims.Credentials[repository.KindergartenUpdateCredential]
  if !credential {
    return c.Status(fiber.StatusForbidden).JSON(fiber.Map{
      "error": true, "msg": "permission denied, check credentials
of your token", })  }
```

Лістинг 4.6 – Зчитування JWT токену

```
func GetKindergartens(c *fiber.Ctx) error {
  db, err := database.OpenDBConnection()
  if err != nil {
    return
c.Status(fiber.StatusInternalServerError).JSON(fiber.Map{
      "error": true,
      "msg":   err.Error(),  })
  }
  result, err := db.GetKindergartens()
  if err != nil {
    return c.Status(fiber.StatusNotFound).JSON(fiber.Map{
      "error":  true,
      "msg":    err.Error(),
      "count":  0,
      "result": nil,  })
  }
  return c.JSON(fiber.Map{
    "error":  false,
    "msg":    nil,
    "count":  len(result),
    "result": result,  })
}
```

Лістинг 4.7 – Контролер отримання всіх об’єктів закладів дошкільної освіти

У контролері виконується процес відкриття підключення до бази даних, після чого контролер формує модель об’єктно-реляційного відображення до таблиці та посилає запит до бази даних через відповідний зовнішній клас. Функцію для отримання даних із сутності «Заклади дошкільної освіти» наведено у лістингу 4.8.

```
func (q *KindergartenQueries) GetKindergartens()
([]models.Kindergartens, error) {
  model := []models.Kindergartens{}
  result := q.Find(&model)
  if result.Error != nil {
    return model, result.Error
    }
  return model, nil
}
```

Лістинг 4.8 – Функція для отримання даних із усіх закладів дошкільної освіти

Як і решта функцій, вона використовує бібліотеку GORM при надсиланні запитів, що робить її універсальною для інших моделей. У лістингах 4.9 – 4.11 вказані функції для створення, редагування та видалення закладу дошкільної освіти.

```
func (q *KindergartenQueries) CreateKindergarten(m
*models.Kindergartens) error {
  result := q.Create(m)
  if result.Error != nil {
    return result.Error
  }
  return nil
}
```

Лістинг 4.9 – Функція для створення закладу дошкільної освіти

```
func (q *KindergartenQueries) UpdateKindergarten(model
models.Kindergartens, m *models.Kindergartens) error {
  result := q.Model(&model).Updates(m)
  if result.Error != nil {
    return result.Error
    }
  return nil
}
```

Лістинг 4.10 – Функція для оновлення інформації про заклад дошкільної освіти

```
func (q *KindergartenQueries) DeleteKindergarten(id int) error {
  model := models.Kindergartens{}
  result := q.Delete(&model, id)
  if result.Error != nil {
    return result.Error
  }
  return nil
}}
```

Лістинг 4.11 – Функція видалення закладу дошкільної освіти

4.3 Опис інтерфейсу застосунку

Перед початком користування застосунком необхідно, щоб адміністратор системи надав потенційному користувачу обліковий запис із правами відповідної категорії. Після цього користувач отримує можливість зайти у систему під своїми обліковими даними: логіном (електронна пошта, або псевдонім) та паролем. Для авторизації необхідно увійти у однойменну сторінку та ввести у форму свої облікові дані (рис. 4.3).

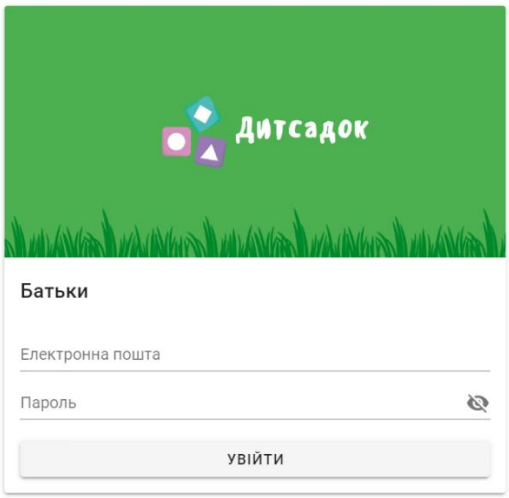

Рисунок 4.3 – Форма авторизації користувача

Після ведення користувачем своїх облікових даних, вони надсилаються запитом на сторону API, де за ними проводиться пошук у таблиці користувачів. Якщо користувача з відповідним обліковим записом знайдено, тоді для нього генерується JWT токен та повертається у відповідь клієнту.

Метод авторизації користувача наведений у лістингу 4.12.

```
signIn() { setTimeout(() => (this.loader = false), 3000);
    if (this.$refs.form.validate()) {
      axios .post("http://127.0.0.1:5000/api/user/sign/in",
this.credentials)
```

Лістинг 4.12 – Метод авторизації користувача

```
.then((result) => {
  localStorage.isLogin = true;
  this.loader = false;
  if (result.data.category == "User") {
    this.error_in_request = false;
    localStorage.access =
    result.data.tokens.access;
    localStorage.refresh =
    result.data.tokens.refresh;
    localStorage.uuid = result.data.uuid;
    localStorage.profile_uuid =
    result.data.profile_uuid;
    window.location.href = "main";
  } else {
    this.error_in_request = true;
    this.error_msg = "Invalid credentials";
    this.$refs.form.reset();
  }    })
.catch((error) => {
  this.error = error.response.data.error;
  this.errorMsg = error.response.data.msg;
  this.$refs.form.reset()
  this.loader = false;
});    }}
```

Лістинг 4.12, лист 2

Після того, як запит пройшов успішно, користувач переходить на головну сторінку системи (рис. 4.4). На цій сторінці знаходиться довідкова інформація про заплановані події у пов’язаному закладі. Усі дії користувача на даній сторінці, що потребують запитів до API, надсилаються із закріпленим токеном у заголовку HTTP, який надає права доступу до використання тієї чи іншої кінцевої точки.

Список усіх створених кінцевих точок наведений у додатку Г.

Довідкова інформація про події на сторінці розміщуються та відтворюються динамічно, в залежності від вмісту отриманого у відповідь JSON об’єкту. Перед створенням сторінки, утиліта Axios посилає декілька подібних запитів та зберігає відповідь у внутрішньому сховищі, як змінну.

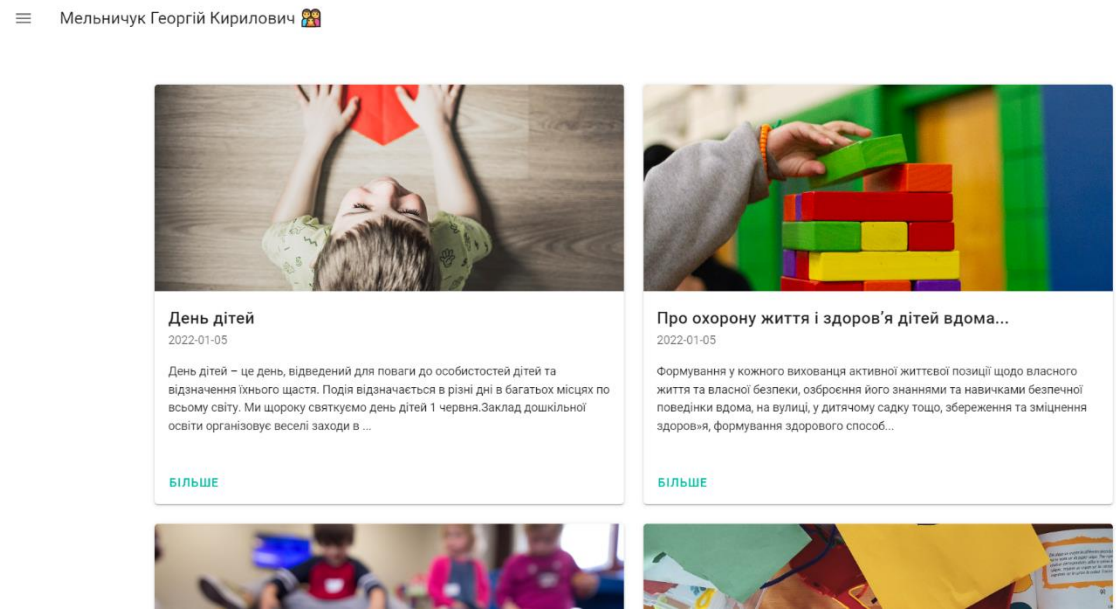

Рисунок 4.4 – Головна сторінка для батьків

У лістингу 4.14 наведений приклад кодової реалізації надсилання HTTP запиту:

```
axios
  .get(`http://127.0.0.1:5000/api/event/${kindergarten_id}`,
  { headers: { Authorization: `Bearer ${cookie.access}` } })
  .then((response) => {
    this.events = response.data.result
  }
})
```

Лістинг 4.14 – Запит до API через утиліту Axios

Аналогічні запити через протокол HTTP надходять для отримання об'єктів інших сутностей в решті сторінок застосунку.

Після того, як змінній events привласнюється значення отриманої відповіді у вигляді JSON об'єкту, вона підставляється у Vue-компонент, котрий через цикл відтворює відповідні картки з довідковою інформацією:

```
<v-col v-for="(item, i) in events" :key="i" cols="6">
```

Користувач при натисканні на позначку «список» (зліва у верхньому куті сторінки), може відкрити висувне меню користувача (рис. 4.5).

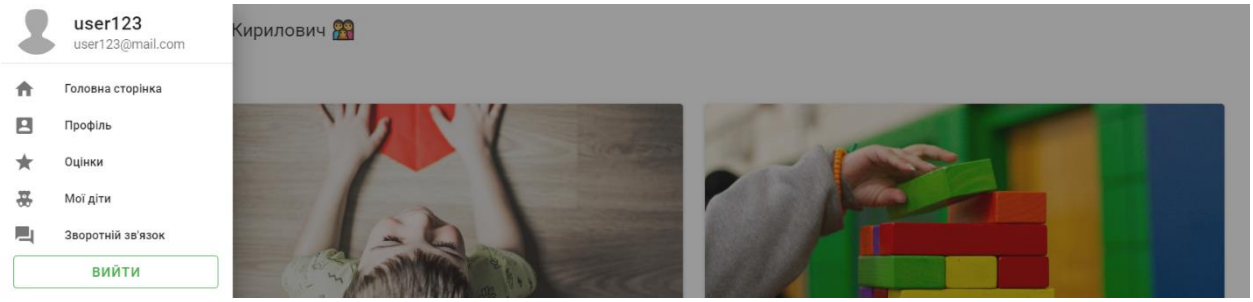

Рисунок 4.5 – Висувне меню користувача

У даному висувному меню відбувається основна навігація по застосунку, де користувач має змогу обрати необхідну сторінку для переходу, вийти із системи або переглянути, через який обліковий запис він авторизований у системі.

Список пунктів навігаційного меню відтворюється також за рахунок динамічного компоненту Vue, що наведено у лістингу 4.15.

```
<v-list-item>
  <v-for="(item, index) in menu_items"
     :key="index"
     @click="menuDialog(item.value)">
     <v-list-item-title>
     {
     { item.title }
     }
     </v-list-item-title>
</v-list-item>
```

Лістинг 4.15 – Vue-компонент для відтворення списку пунктів навігаційного меню

Сторінка «Профіль» містить інформацію про користувача системи: облікові дані, контактні дані тощо. Разом з цим користувач має змогу переглянути інформацію про заклад, який відвідують його діти (рис. 4.6).

Рисунок 4.6 – Профіль користувача

У картці з інформацією користувача знизу знаходяться кнопки, через які користувач має змогу змінити свої облікові дані або пароль (рис. 4.7 та 4.8).

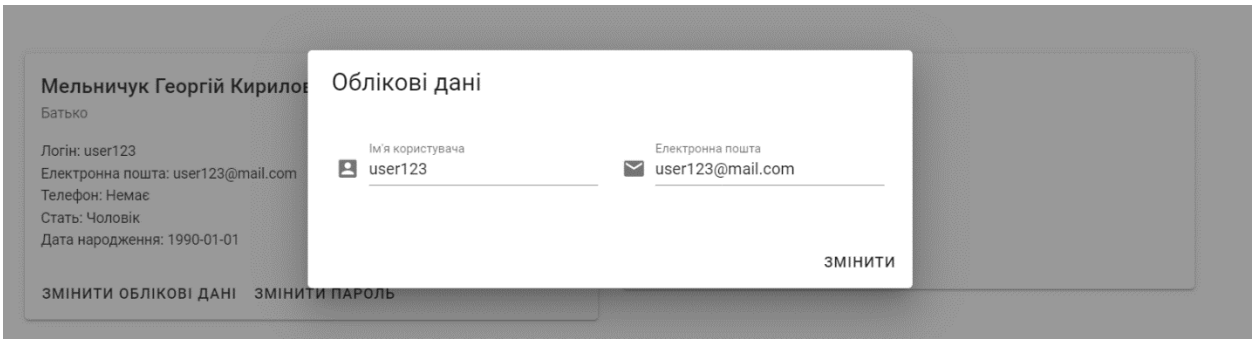

Рисунок 4.7 – Зміна облікових даних

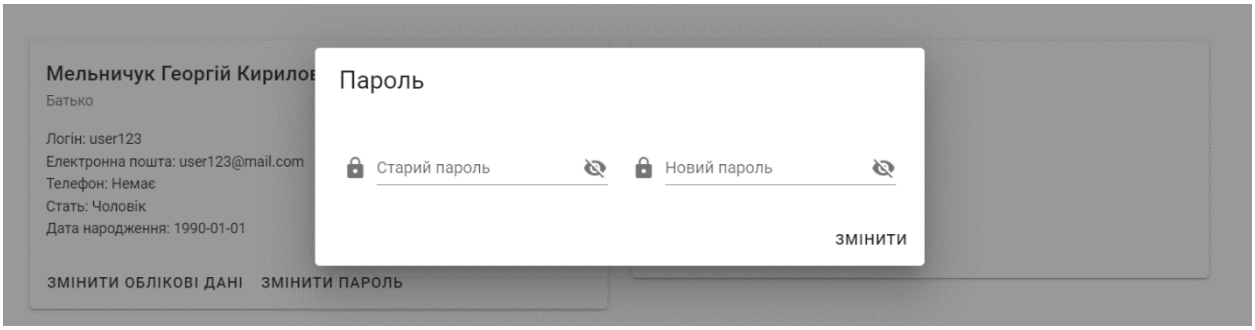

Рисунок 4.8 – Зміна паролю

Сторінка «Оцінки» містить електронний звіт про діяльність дітей у закладі, в ньому користувач має змогу переглянути ім’я та прізвище дитини, отриману ним оцінку, дату отримання оцінки та вид діяльності, за який вона виставлена (рис. 4.9).

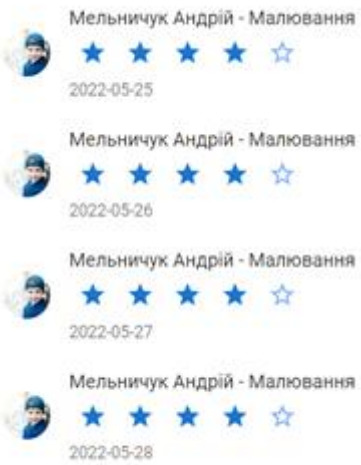

Рисунок 4.9 – Електронний звіт про діяльність дітей

Сторінка «Мої діти» містить інформацію про дітей, також через картки можна перейти на електронний звіт про діяльність дитини (рис. 4.10).

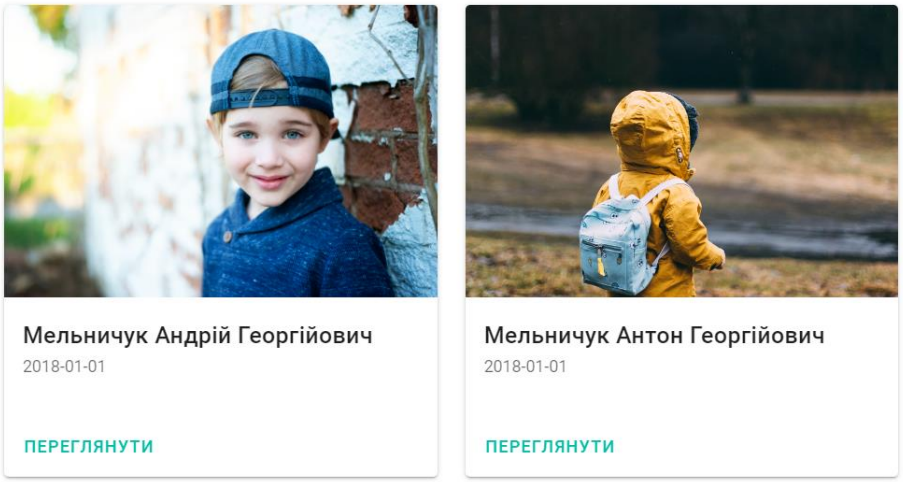

Рисунок 4.10 – Перегляд дітей

На головній сторінці користувач має змогу перемкнути режим системи для переходу на більш сприйнятливий для дитини користувальницький інтерфейс. При натисканні користувачем на позначку «Дитяча іграшка» у правому верхньому куті сторінки (рис. 4.11), для користувача висовується меню із списком його дітей, для котрих необхідно змінити режим перегляду інформації.

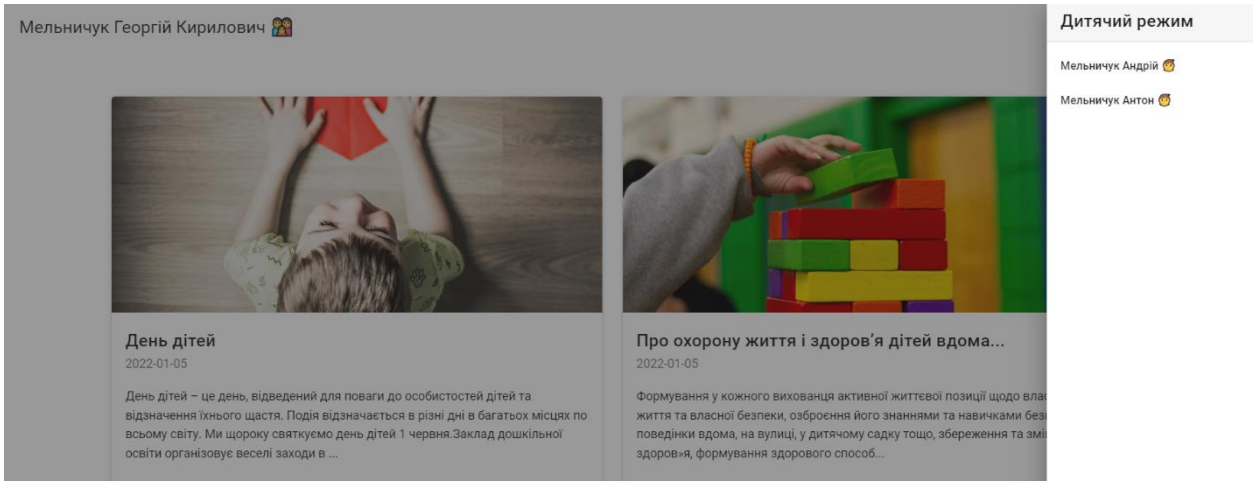

Рисунок 4.11 – Перемикання на дитячий режим

Порівняно зі звичайним режимом, дитячий режим не має багато функціональних змін, окрім оновленого списку опцій у висувному меню, що зображено на рис. 4.12.

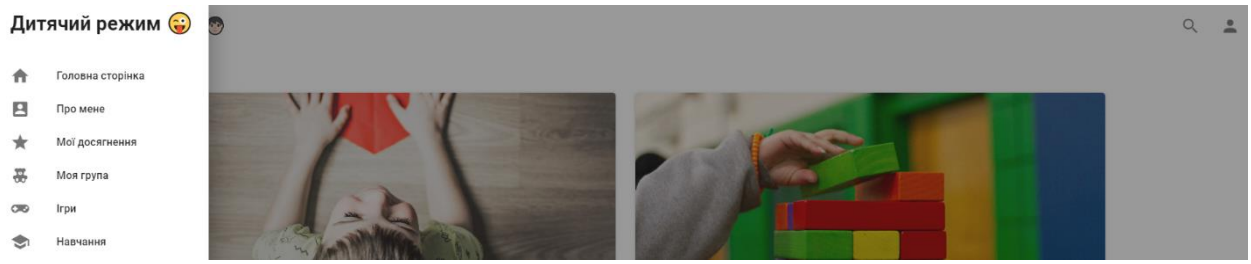

Рисунок 4.12 – Висувне меню у режимі для дітей

Для користування інформаційною системою, співробітнику, так само, як і користувачу, необхідно отримати власний обліковий запис. Після цього співробітник закладу має можливість увійти у систему під своїми обліковими даними через форму авторизації для співробітників (рис. 4.13).

Після успішної авторизації та отримання пари токенів, співробітник переходить на головну сторінку, де відображені його ПІБ (рис. 4.14). Порівняно зі звичайними користувачами системами, співробітник має більший список опцій у висувному меню та функціонал (рис. 4.15).

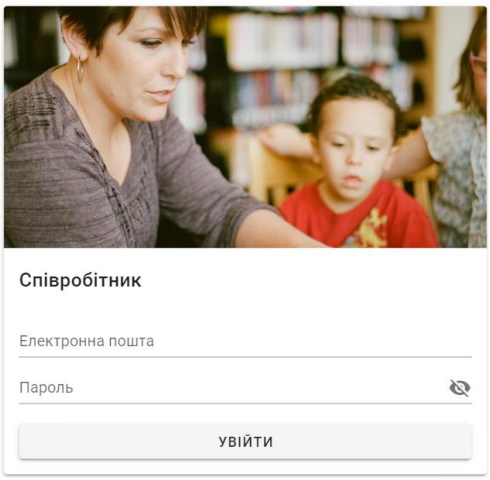

Рисунок 4.13 – Форма авторизації співробітника

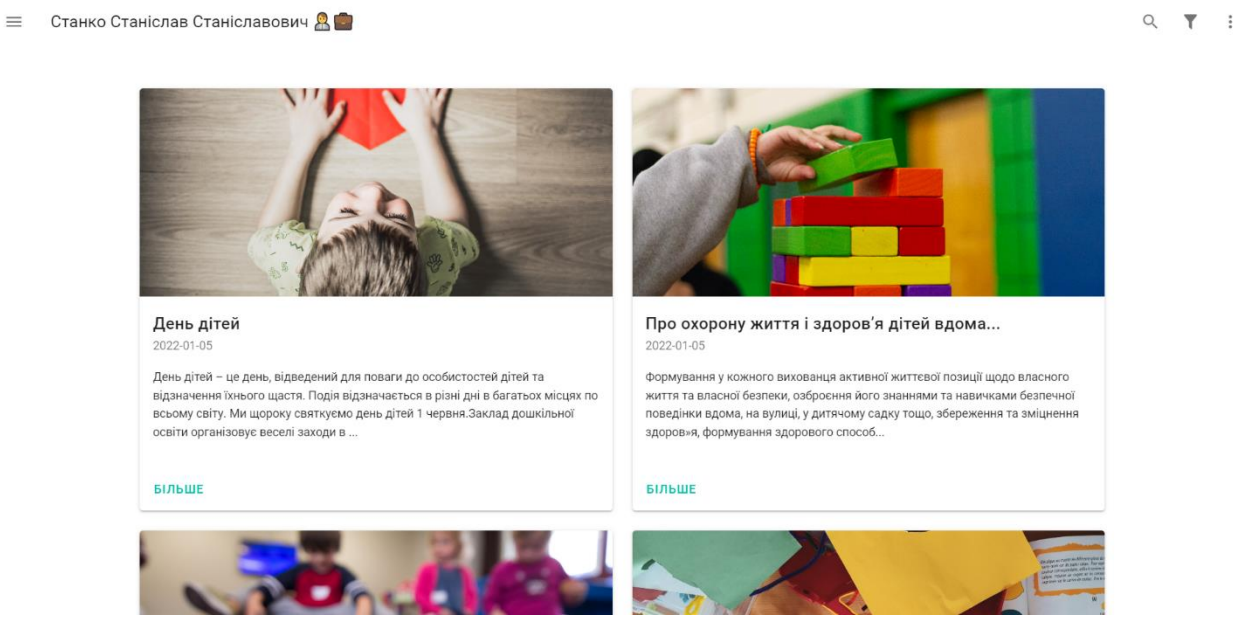

Рисунок 4.14 – Головна сторінка співробітника

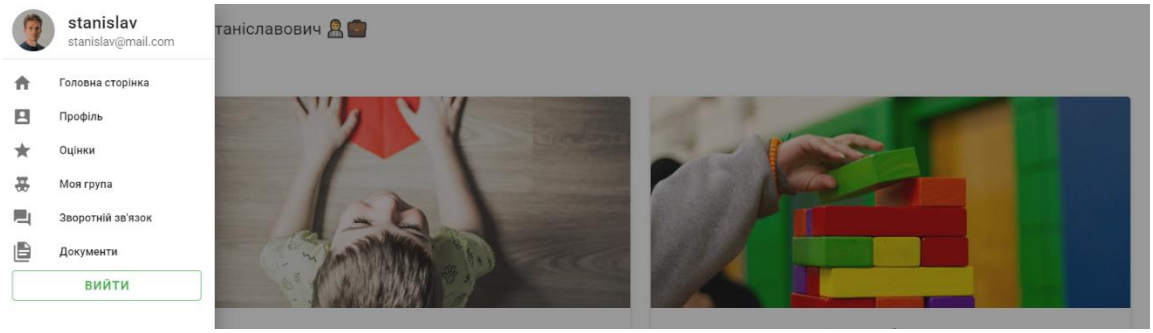

Рисунок 4.15 – Висувне меню співробітника

На головній сторінці системи співробітник, як і користувач, має можливість робити пошук по довідковій інформації. Для цього співробітнику необхідно натиснути на позначку «Збільшувальне скло», праворуч у верхньому кутку, щоб відкрити контекстне меню пошуку та ввести свій запит. Приклад пошуку зображений на рис. 4.16.

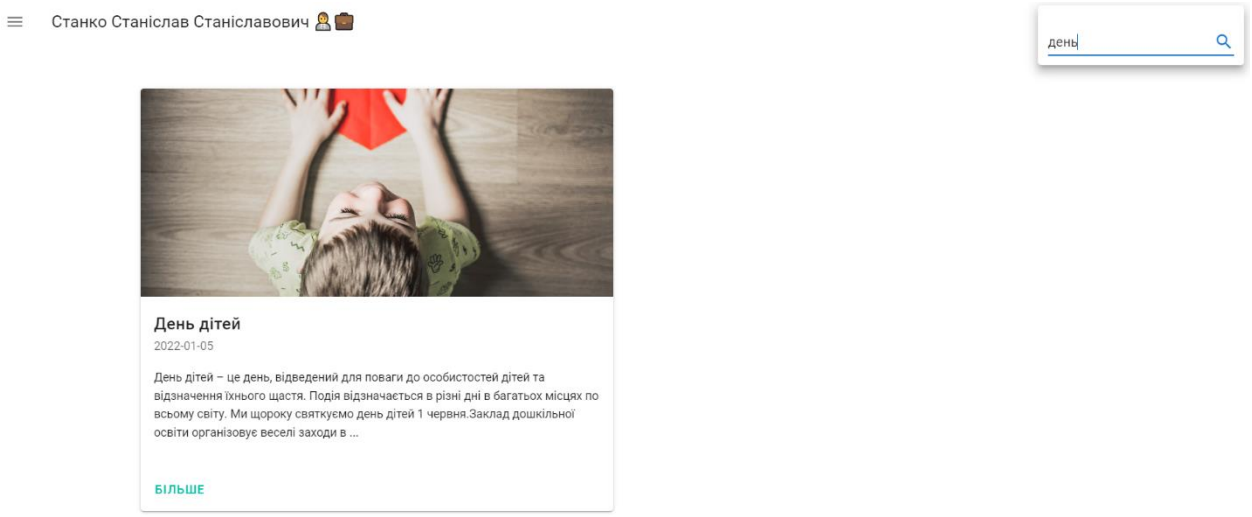

Рисунок 4.16 – Пошук подій

Результат пошуку інформації про події реалізується наступним чином (лістинг 4.16):

```
searched_items: function () {
      if (this.search) {
        return this.items.filter(item => {
          return item.name.toLowerCase().includes
               (this.search.toLowerCase())
               }
            );
          }
     else {
        return this.items;
      }
    },
```

Лістинг 4.16 – Метод пошуку події

На сторінці, зображеній на рис. 4.17, співробітник має можливість керувати інформацією про події. Для цього необхідно натиснути на позначку «три крапки» для відкриття контекстного меню керування подіями та обрати одну із опцій: створення, редагування та видалення події. Для створення події співробітнику необхідно натиснути на відповідну опцію «Додати подію», щоб відкрити форму з полями нової події (рис. 4.18).

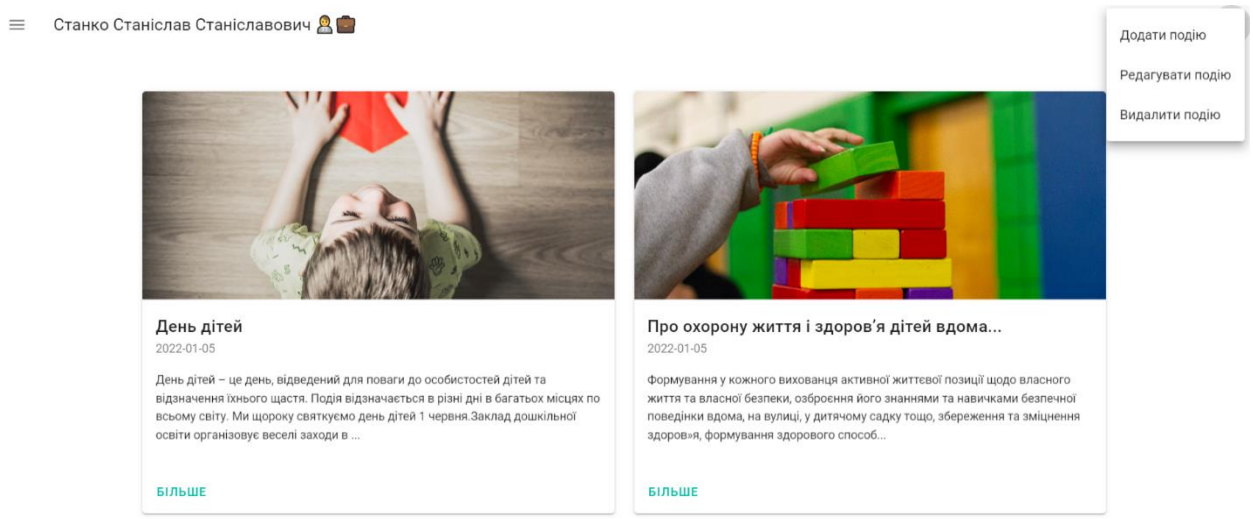

Рисунок 4.17 – Контекстне меню керування подіями

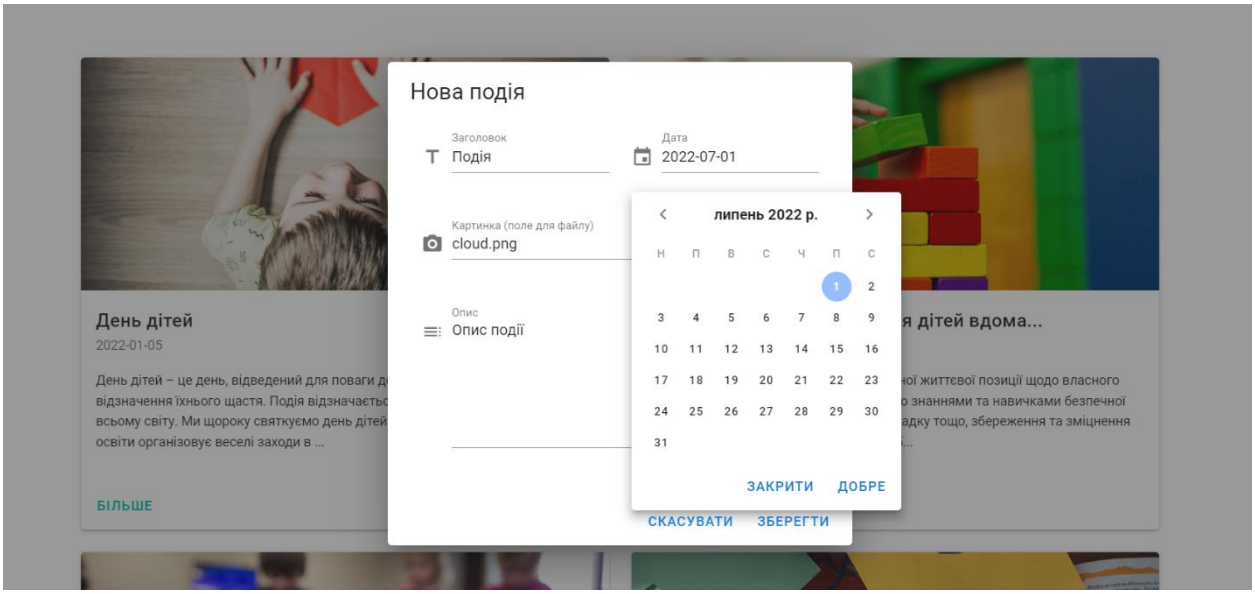

Рисунок 4.18 – Створення нової події

Для використання інших опцій управління інформацією про події, таких як «Редагування події» або «Видалення події», співробітнику необхідно попередньо обрати потрібну подію, натиснувши на відповідну довідкову картку (рис. 4.19).

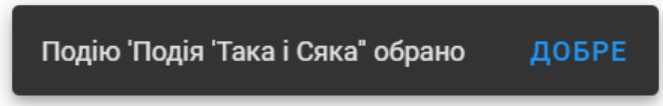

Рисунок 4.19 – Обрання події

Приклади редагування та видалення співробітником інформації про події наведені відповідно на рис. 4.20 та 4.21.

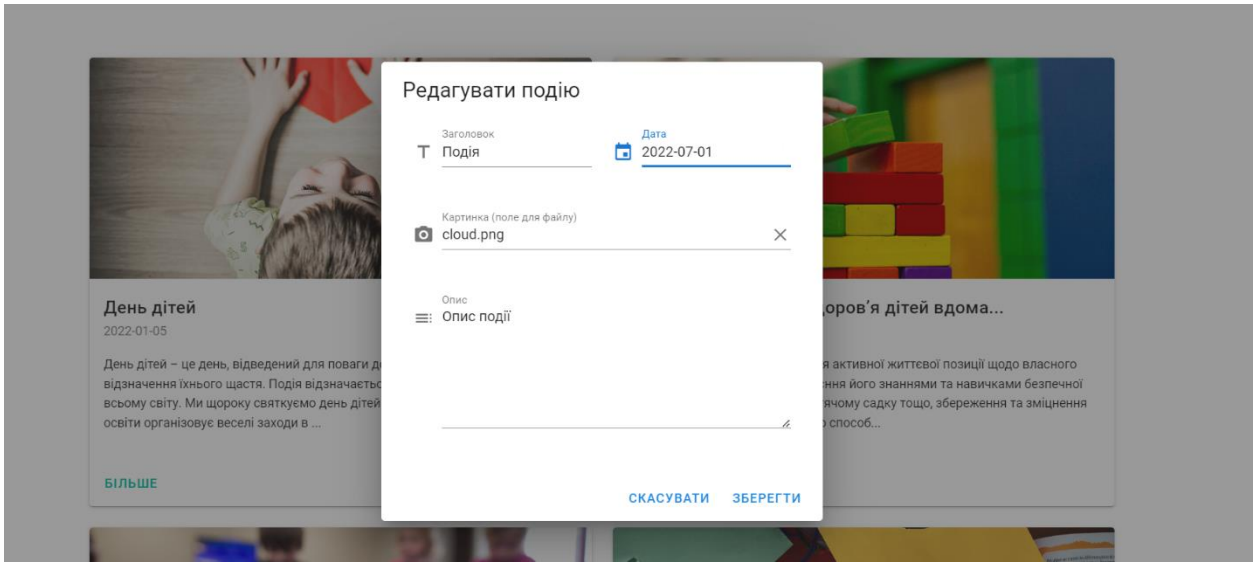

Рисунок 4.20 – Редагування події

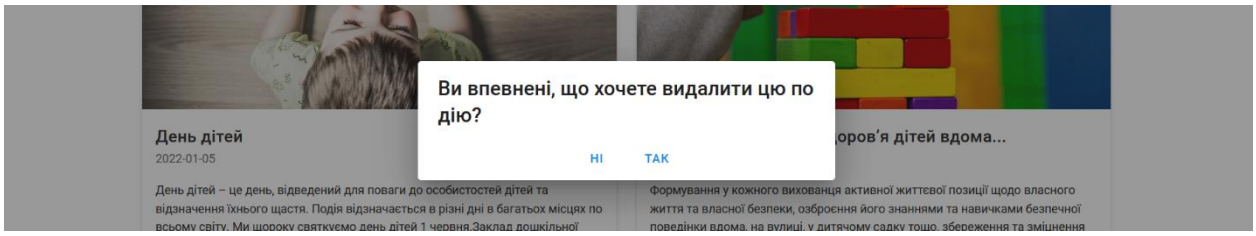

Рисунок 4.21 – Видалення події

На сторінці «Оцінки» співробітник має можливість вносити та редагувати оцінки у електронному звіті дитини через контекстне меню (рис. 4.22).

Рисунок 4.22 – Редагування оцінки співробітником

Адміністратор системи має змогу увійти у систему під своїми обліковими даними через форму авторизації для адміністратора (рис. 4.23).

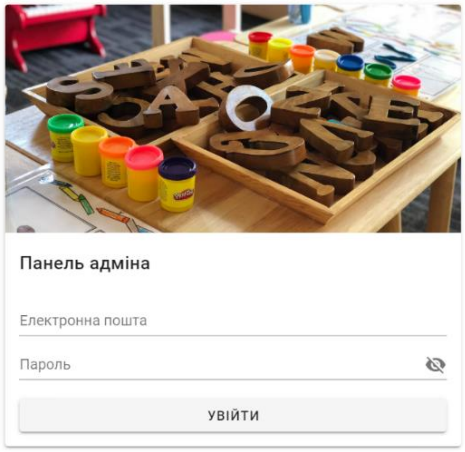

Рисунок 4.23 – Форма авторизації адміністратора

Після того, як адміністратор авторизувався у системі, він переходить на головну сторінку панелі адміністратора, де має можливості перегляду, додавання, редагування та видалення рядків у таблицях, що зберігають дані предметної області. Інтерфейс адміністратора та зображення висувного меню в ньому наведені на рис. 4.24.

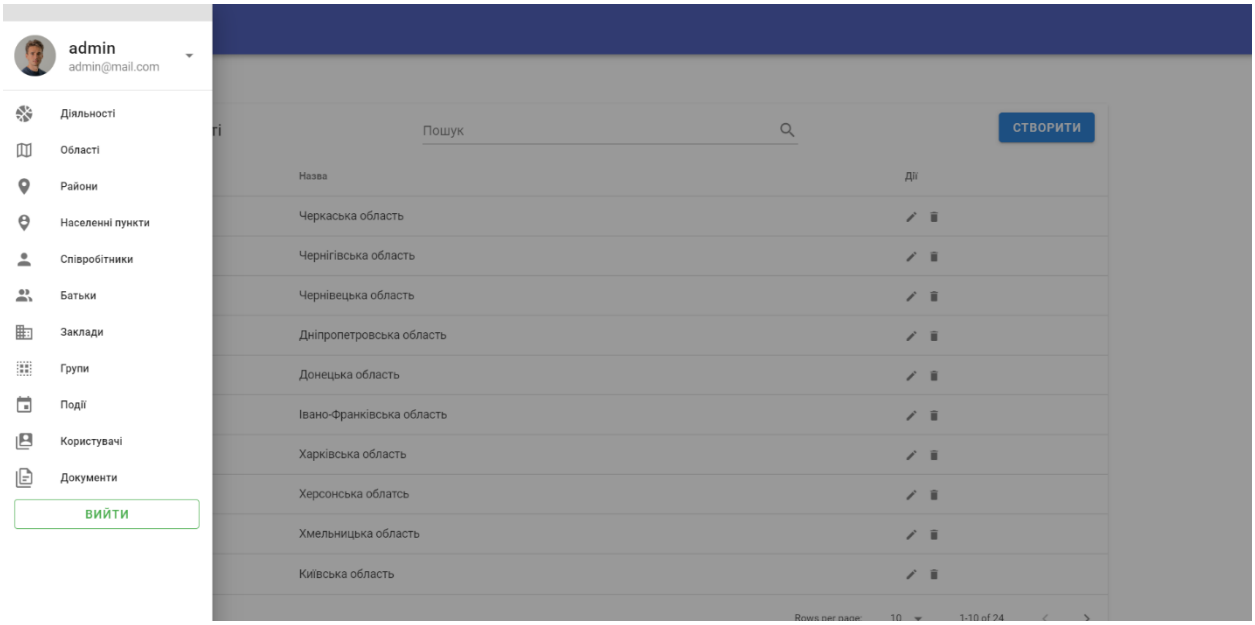

Рисунок 4.24 – Інтерфейс та висувне меню адміністратора

Також адміністратор може додатково взаємодіяти із таблицями, змінюючи кількість рядків на відповідній сторінці, фільтруючи поля таблиці, або роблячи пошук у рядках за допомогою запитів (рис. 4.25).

Рисунок 4.25 – Пошук та фільтрування у таблиці «Області»

4.4 Засоби захисту застосунку

Безпека на рівні бази даних забезпечується шляхом створення категорій користувачів та присвоєння їм привілеїв. З вмістом запитів на створення категорій користувачів та їх привілеїв можна ознайомитись у додатку В.

Присвоєння користувачеві його категорії відбувається наступним чином: після того як користувач ввів свої облікові дані, система надсилає запит до бази даних, де зберігається таблиця з обліковими записами користувачів. Після чого відбувається порівняння існуючих та введених облікових даних та визначається категорія користувача. Якщо існуючі та введені облікові дані співпадають, то створюється нове підключення до бази даних під з привілеями категорії, зазначеної у відповідному полі таблиці.

Крім того, перед надісланням запиту відбувається шифрування введеного користувальницького пароля на рівні сервера. Шифрування пароля відбувається з використання алгоритму хешування bcrypt, тому до бази даних паролі користувачів потрапляють вже у зашифрованому вигляді. Розподіл прав доступу для користувачів наведений в таблиці 4.1.

Таблиця 4.1 – Права доступу користувачів до об’єктів бази даних

Назва об’єкта	Користувачі БД		
	Користувач	Співробітник	Адміністратор
Таблиці			
activities	R	CRU	CRUD
children_groups	R	R	CRUD
childrens	RU	CRU	CRUD
districts	R	R	CRUD
documents	–	CRUD	CRUD
events	R	CRUD	CRUD
family	RU	CRU	CRUD
grades	R	CRUD	CRUD
groups	R	CRU	CRUD
kindergartens	R	RU	CRUD
locality	R	R	CRUD
parents	RU	CRU	CRUD
positions	R	CRU	CRUD
regions	R	R	CRUD

Продовження таблиці 4.1

Назва об’єкта	Користувачі БД		
	Користувач	Співробітник	Адміністратор
staff_groups	R	RU	CRUD
staffs	R	R	CRUD
users	RU	RU	CRUD
Представлення			
childrens_by_parents_profile_view	R	R	R
family_view	R	R	R
staff_user_view	R	R	R

Позначення, що використовуються у таблиці:

– C – додавання нової інформації до таблиці;

– R – перегляд вмісту таблиці;

– U – редагування існуючої інформації;

– D – видалення інформації з таблиці.

Код SQL-запитів для налаштування доступу до системи наведений у додатку В.

При авторизації користувача на сервері застосунку генерується пара JWT токенів, щоб ідентифікувати його для подальшого обміну інформацією із API. Пара токенів складається із токену для доступу та оновлення. Токен доступу зберігається у локальному сховищі на стороні клієнта та перевіряється на повноваження того, чи іншого запиту перед початком його виконання.

Токен оновлення використовується для створення нового токену доступу у разі закінчення його терміну дії. Коли користувач виходить з системи, токен на клієнтській стороні самознищується, тому безпосередньо немає необхідності у його взаємодії із сервером.

Усі токени для оновлення, які генеруються сервером, зберігаються у окремому розподіленому сховищі Redis для створення кешу та забезпечення довговічності їх зберігання (рис. 4.26, 4.27).

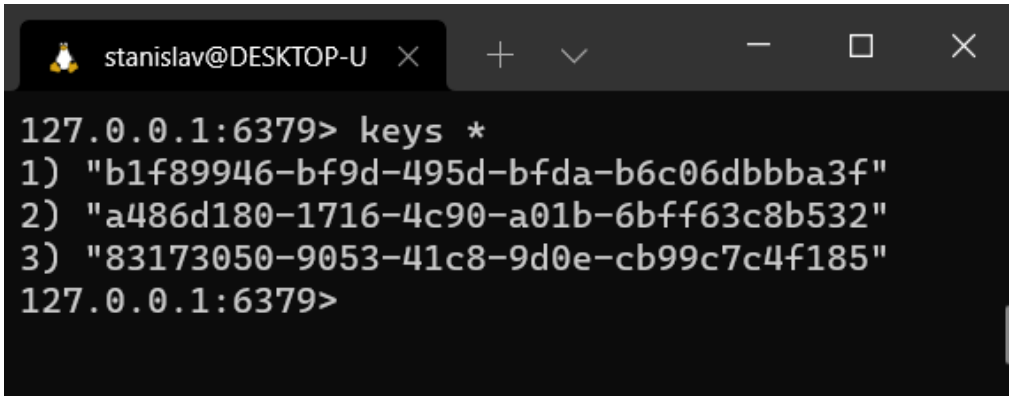

Рисунок 4.26 – Збережені сесії користувачів у розподіленому сховищі Redis

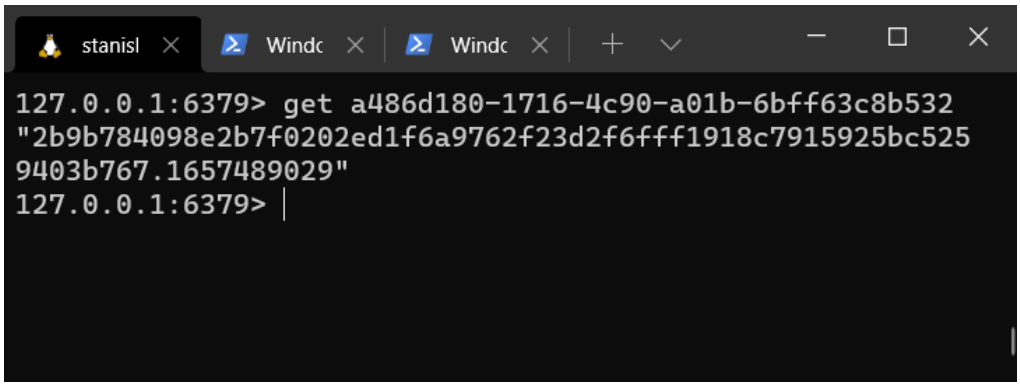

Рисунок 4.27 – Значення токену оновлення

5 СТВОРЕННЯ ХМАРНОЇ ІНФРАСТРУКТУРИ ТА РОЗГОРТАННЯ CLOUD NATIVE ЗАСТОСУНКУ ДЛЯ ДОШКІЛЬНИХ НАВЧАЛЬНИХ ЗАКЛАДІВ

5.1 Створення хмарної інфраструктури

Перед створенням хмарної інфраструктури слід створити обліковий запис на платформі постачальника хмарних послуг. Платформа DigitalOcean пропонує три варіанта отримання облікового запису користувачем: через персональну електронну пошту, через обліковий запис Google та через обліковий запис GitHub.

Після обрання одного із варіантів та його проходження, для повного створення облікового запису, слід також зазначити дані свого платіжного рахунку, для ідентифікації себе, як реального користувача, спроможного платити власні кошти при подальшому використанні хмарної платформи. Після того, як користувач вказав свої платіжні дані, постачальник списує деяку невелику частину коштів та практично відразу повертає їх замовнику. Як тільки транзакція проходить успішно – етап створення облікового запису пройдено.

За замовчуванням, для нових користувачів платформи надається пробна пропозиція на два місяці із лімітом використання у сто доларів на безоплатній основі. Ліміт, окрім обмеження у часі, немає ніякий інших обмежень, тобто може бути використаним на будь-які потреби замовника. Слід зауважити, що наявна також пропозиція використання ліміту уздовж дванадцяти місяців, за умови, якщо користувач зможе підтвердити, що на даний момент він є студентом та вчиться у закладі вищої освіти. Зазвичай, це можна підтвердити через наявність пакету послуг GitHub Education, який містить дану пропозицію, та підтверджується через електронну пошту свого вищого навчального закладу.

За допомогою створеного облікового запису можна розпочати створення хмарної інфраструктури. На початку слід створити VPC мережу для оголошення хмарного середовища зі спільно виділеними ресурсами, тобто усі новостворені ресурси будуть знаходитись у одній приватній мережі із

можливістю взаємодіяти між собою через приватні IP адреси. При створенні VPC мережі слід зазначати лише ім’я мережі та регіон її знаходження. На рис. 5.1 зображені усі групи регіонів, що пропонує постачальник.

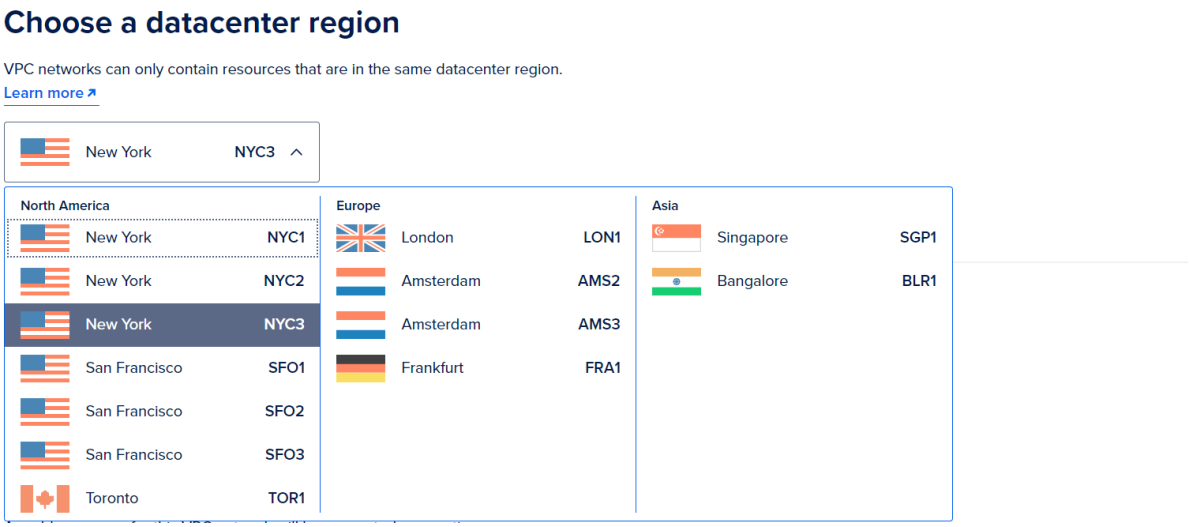

Рисунок 5.1 – Доступні регіони для створення VPC мережі

Серед усіх груп регіонів вирішено використовувати групу Europe та регіони AMS3, LON1, розташовані у ній, в силу їх відносної регіональної близькості порівняно з іншими регіонами, що впливає на стабільність та швидкість взаємодії на рівні зовнішньої мережі. На рис. 5.2 зображені створені мережі VPC.

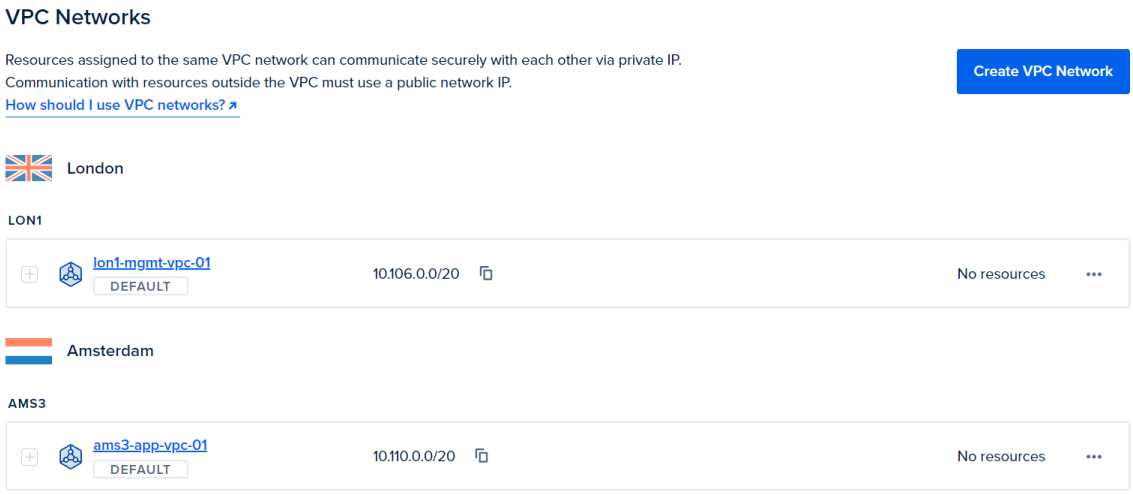

Рисунок 5.2 – Створені мережі VPC

Після створення мережі VPC, на її основі слід створити виділені віртуальні сервери для окремих компонентів застосунку, які будуть виконувати певну функцію. Кожен із них представляє повноцінну віртуальну машину на основі вибраної операційної системи, з повним контролем у встановленні, оновлені та розміщені програм, бібліотек тощо. Усі дії виконуються власноруч та ідеально підходять під потреби розміщення власного застосунку та іншого програмного забезпечення, наприклад, що відповідають за безперервну доставку. VPS, що необхідно створити:

1) виділена віртуальна машина для застосунку;

2) виділена віртуальна машина для керування інфраструктурою застосунку.

Платформа DigitalOcean пропонує створювати власні VPS через послугу «Droplet». На рис. 5.3 наведений список запропонованих операційних систем.

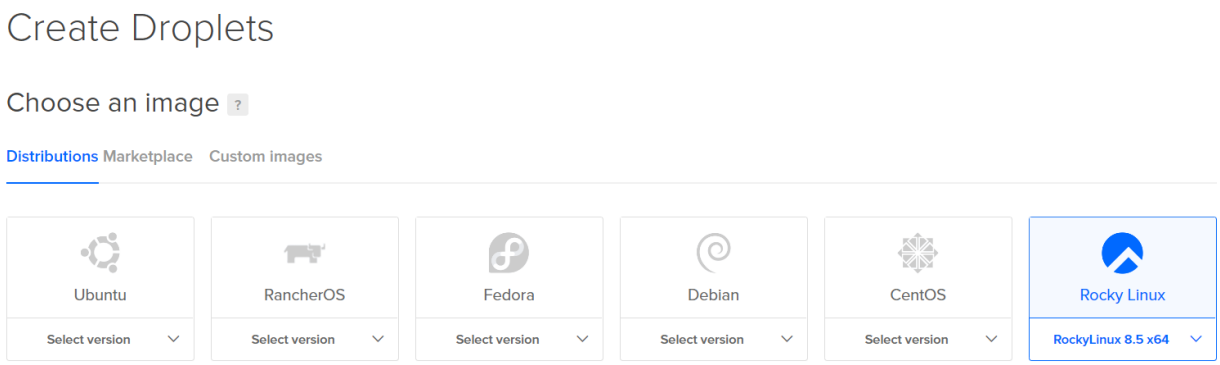

Рисунок 5.3 – Список доступних операційних систем

Далі треба визначити план послуги, від якого залежать загальні характеристики машини та її вартість. Запропоновані постачальником плани наведені на рис. 5.4.

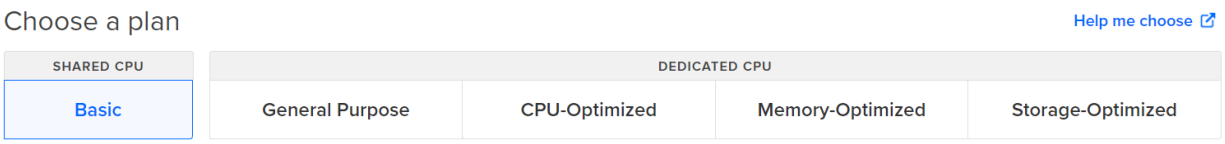

Рисунок 5.4 – Список доступних планів користування послугою «Droplet»

Серед усіх вищезазначених планів обраний базовий план, що пропонує оптимізовані виділені ресурси за відносно невеликі кошти. На рис. 5.5 зображені доступні опції плану «Basic».

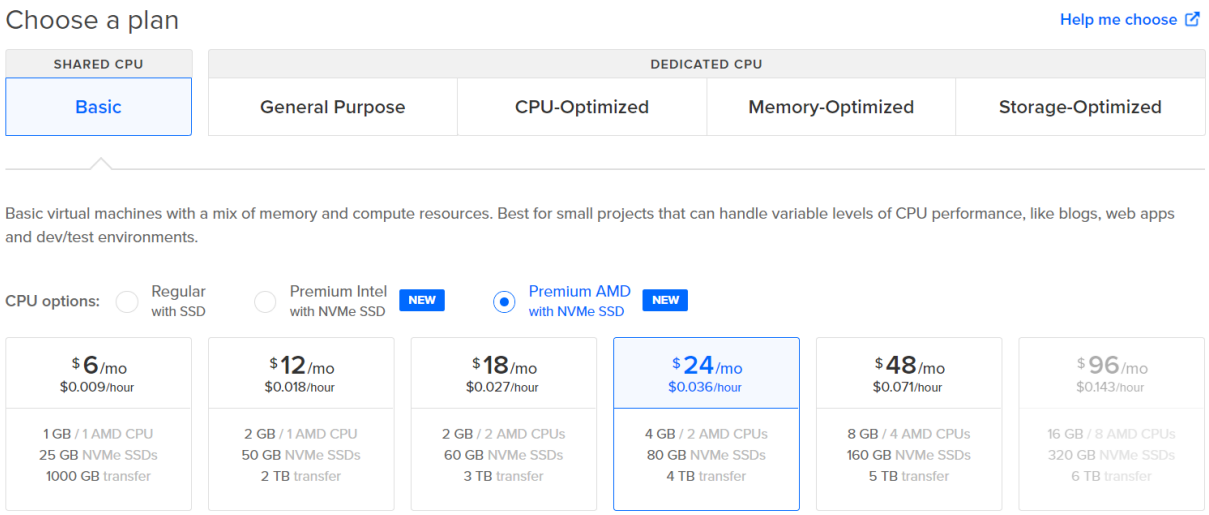

Рисунок 5.5 – Список доступних опцій у базовому плані користування послугою «Droplet»

Наступна секція «Choose a datacenter region» пропонує вибрати регіон датацентру та VPC мережі для розташування виділеної віртуальної машини, що створюється.

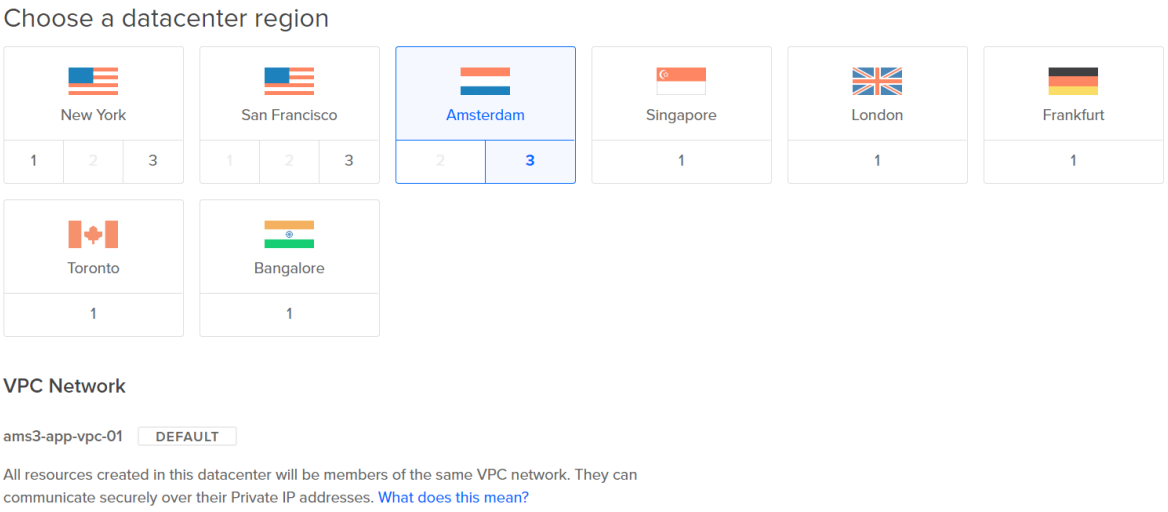

Рисунок 5.6 – Вибір регіону та мережі VPC

Секція «Authentication», зображена на рис. 5.7, пропонує обрати спосіб аутентифікації у виділеній віртуальній машині між ключами SSH та паролем користувача root.

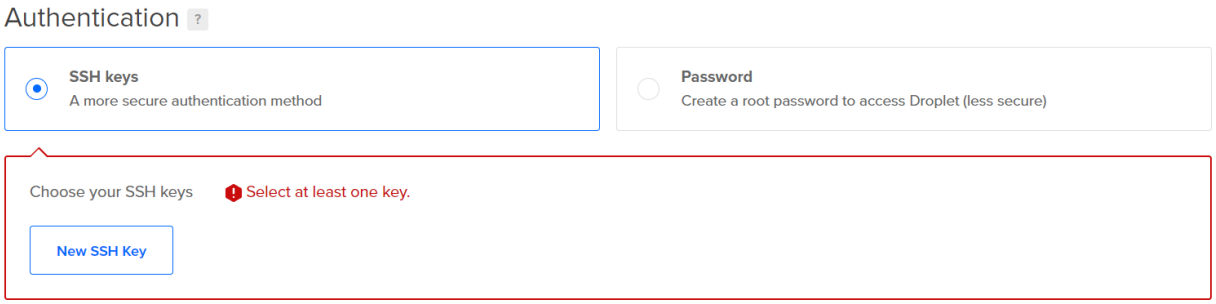

Рисунок 5.7 – Вибір методу аутентифікації

Для більш захищеного віддаленого доступу використовується метод ключів SSH, попередньо згенерувавши публічний та приватний ключ. Для цього використана утиліта OpenSSH та команда для генерації ключів:

```
ssh-keygen -t ed25519 -b 4096
```

Дана команда генерує пару 4096 бітних ключів за алгоритмом EdDSA. Після цього вміст створеного публічного ключа вноситься у поле публічного ключа даної секції, де йому задається назва, з подальшим зберіганням у платформі до наступних використань (рис. 5.8).

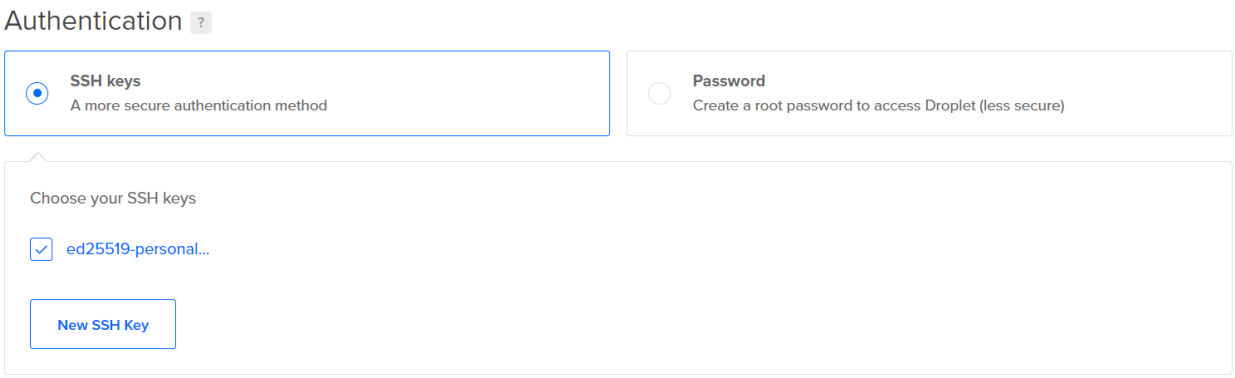

Рисунок 5.8 – Внесений публічний ключ

Секція «Select additional options» пропонує використання додаткових опцій, таких як підтримка створення регулярних резервних копій машини, моніторинг, підтримка протоколу IPv6, сценарії виконання при створенні віртуальної машини.

На даному етапі створення хмарної інфраструктури опції ігноруються. Після створення машини дані опції продовжують бути наявними у підключенні. Секцію «Select additional options» зображено на рис. 5.9.

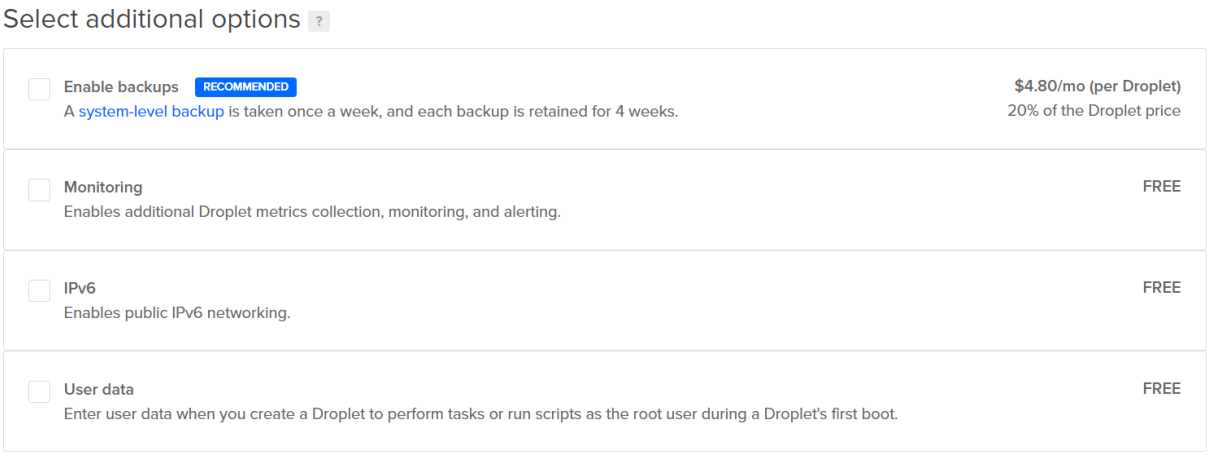

Рисунок 5.9 – Вибір додаткових опцій

Секція «Finalize and creates» є останньою секцією при створенні виділеної віртуальної машини, у даній секції замовник обирає кількість машин, що створюються, назву хосту, тег та приналежність до проекту (рис. 5.10).

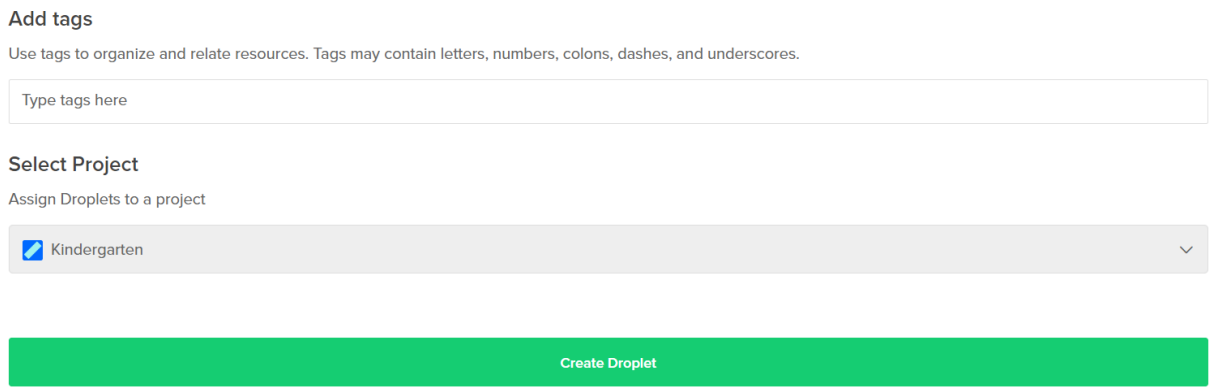

Рисунок 5.10 – Створення виділеної віртуальної машини

У результаті створено два виділених віртуальних сервери, що зображені на рис. 5.11 та 5.12.

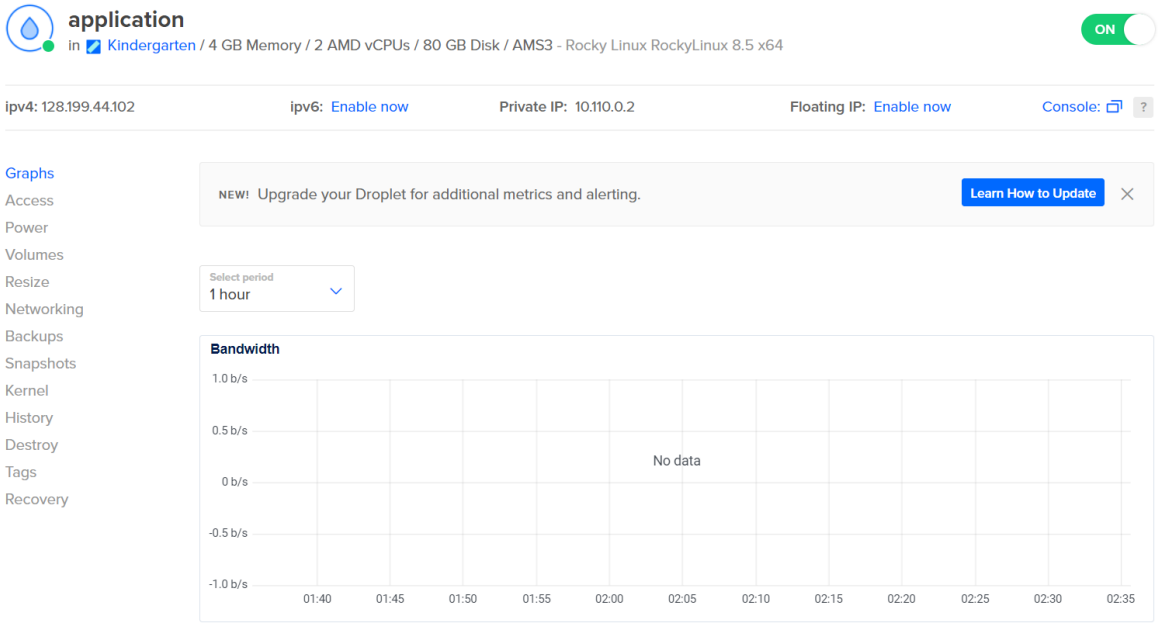

Рисунок 5.11 – Створений віртуальний виділений сервер застосунку

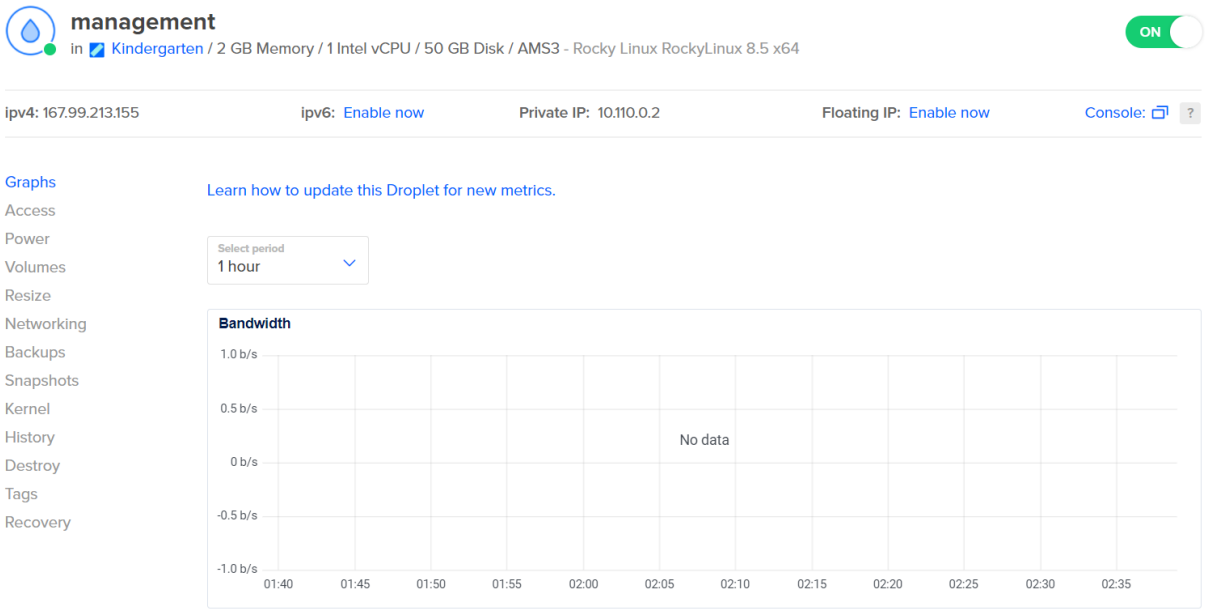

Рисунок 5.12 – Створений виділений віртуальний сервер для керування інфраструктурою застосунку

Характеристики створених віртуальних машин наведені у табл. 5.1 і 5.2.

Таблиця 5.1 – Характеристики виділених ресурсів для віртуального серверу застосунку

Параметр	Значення
Процесор	4 віртуальних ядра на базі AMD
Операційна система	Rocky Linux 8.5 x64
Оперативна пам’ять	4Gb
Ємність диску	80Gb NVME SSD
Ліміт на трафік у мережі	4Tb
Мережа VPC	ams3-app-vpc-01
Назва хосту	application

Таблиця 5.2 – Характеристики виділених ресурсів для віртуального серверу керування інфраструктурою застосунку

Параметр	Значення
Процесор	2 віртуальних ядра на базі Intel
Операційна система	Rocky Linux 8.5 x64
Оперативна пам’ять	2Gb
Ємність диску	50Gb NVME SSD
Ліміт на трафік у мережі	2Tb
Мережа VPC	ams3-app-vpc-01
Назва хосту	management

У ході створення хмарної інфраструктури виникає необхідність у використанні контейнерів для деяких компонентів застосунку, а саме для інтерфейсів прикладного програмування.

Для зберігання контейнерів використовується репозиторій контейнерів, що надається послугою «Container registry». На рис. 5.13 зображений створений репозиторій контейнерів.

Рисунок 5.13 – Створений репозиторій контейнерів

Для зберігання контенту інформаційної системи (зображень, статичних файлів, зовнішніх конфігураційних файлів, документів тощо) використовується зовнішнє хмарне сховище, що надається послугою «Spaces».

Процес створення зовнішнього хмарного сховища показаний на рис. 5.14.

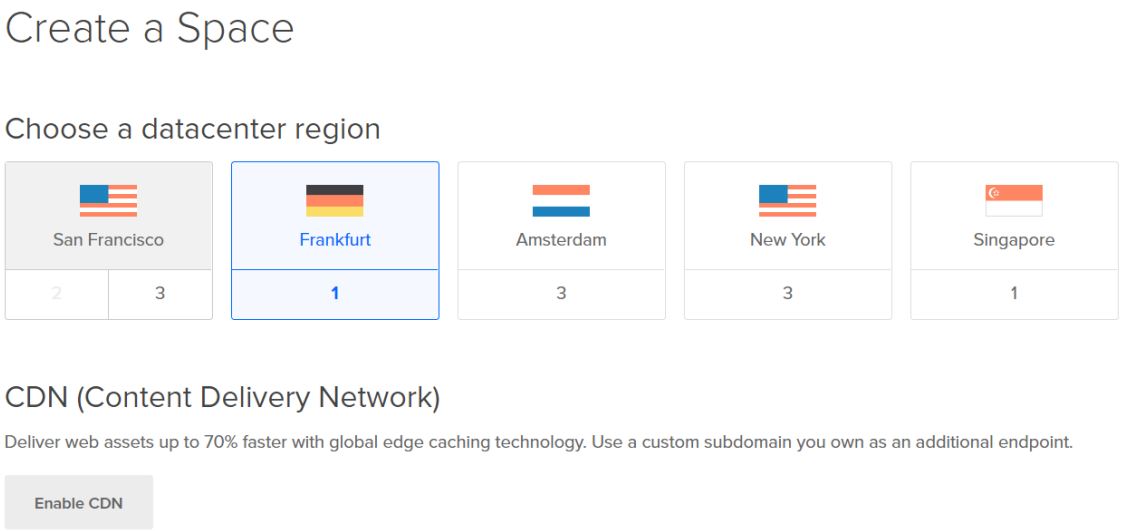

Рисунок 5.14 – Створення зовнішнього хмарного сховища

У секції «Create a Space» слід зазначити регіон розташування сховища та необхідність опції використання CDN, що оптимізує надсилання контенту та його кешування.

Наступне, що необхідно вказати – це права доступу до сховища: з обмеженням, чи без. У першому варіанті, переглядати список вмісту сховища можуть лише користувачі з певними ключами доступу, у другому – будь-хто може переглядати список, але слід зауважити, що цей варіант не надає повних

прав до перегляду вмісту будь-яких файлів, на котрих встановлено певні додаткові обмеження.

Процес надання прав доступу до сховища зображений на рис. 5.15.

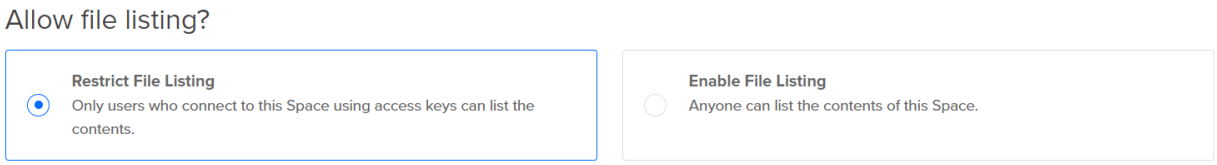

Рисунок 5.15 – Вибір типу доступу до сховища

При створенні сховища обраний регіон FRA1 з використанням послуги CDN.

За замовчуванням, права доступу до сховища не мають обмежень для зовнішніх користувачів, що надає системі можливість доставляти контент без зайвих ускладнень доступу. Слід зазначити, що це стосується лише не конфіденційних файлів. До піддиректорії сховища, що містить документи, окремо надані та встановлені певні права доступу, що не дають змогу переглядати вміст цих документів.

На рис. 5.16 зображено створене сховище та його віртуальні піддиректорії.

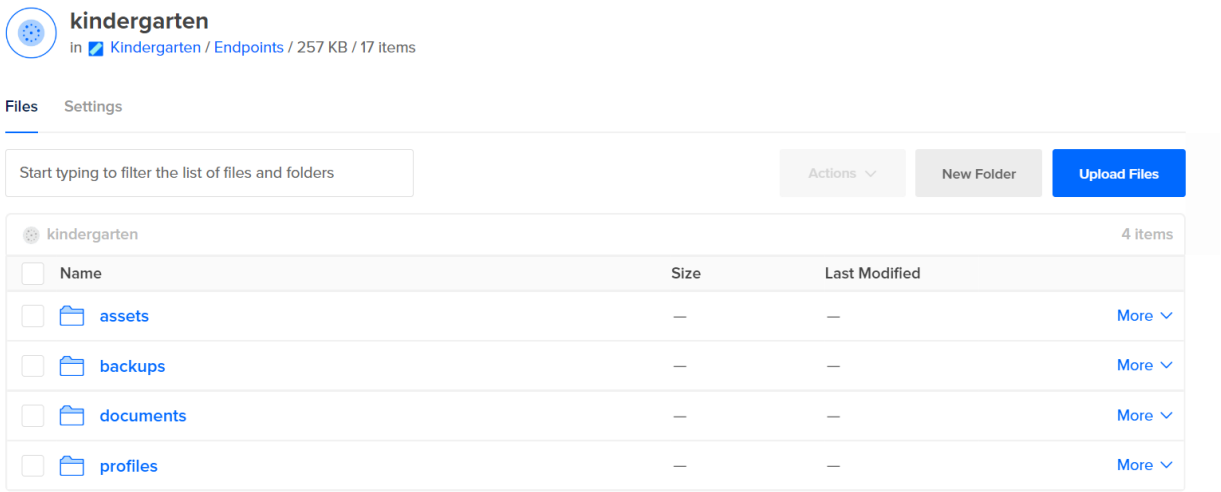

Рисунок 5.16 – Структура створеного сховища

Також у створеному сховищі виділені чотири логічні піддиректорії для вмісту певних типів файлів застосунку:

– assets – для вмісту статичних файлів, які необхідні для роботи презентаційного рівня застосунку;

– backups – для вмісту резервних копій компонентів застосунку;

– documents – для вмісту документів та інших файлів, що належать певним користувачам;

– profiles – для вмісту специфічних файлів користувачів, таких як світлини профілю тощо.

Для зберігання даних інформаційної моделі системи, кешування системи та зберігання ідентифікаторів сесій користувачів використовуються кластери баз даних PostgreSQL та Redis. Для створення відповідних кластерів використовується послуга «Database», що зображено на рис. 5.17.

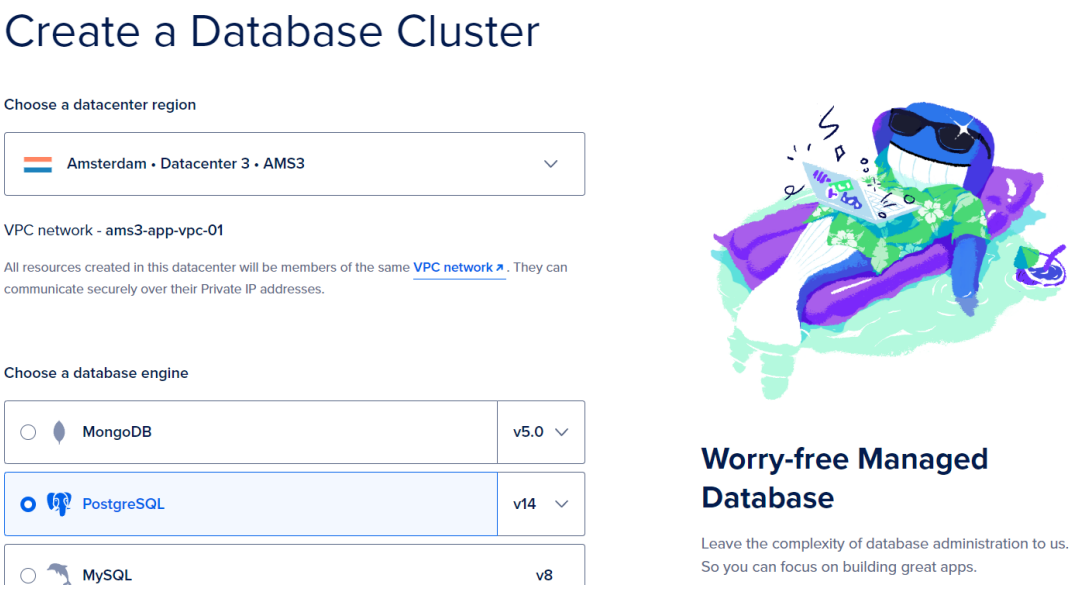

Рисунок 5.17 – Створення кластеру баз даних

Секція «Create a Database Cluster» пропонує вибрати регіон розташування кластеру та движок бази даних. Далі потрібно зазначати план користування кластером, вибір якого зображений на рис. 5.18.

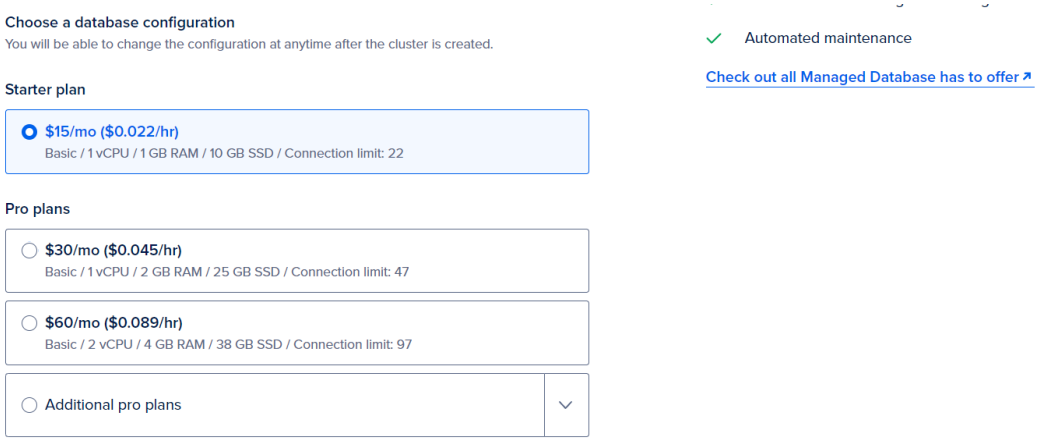

Рисунок 5.18 – Вибір плану користування кластером баз даних

Для створення обох кластерів обраний план «Starter plan», характеристика виділених ресурсів якого вказана у табл. 5.3.

Таблиця 5.3 – Характеристика виділених ресурсів для кластера баз даних

Параметр	Значення
Процесор	1 віртуальне ядро
Оперативна пам’ять	1Gb
Ємність диску	10Gb SSD
Ліміт у активних підключеннях	22

Список створених кластерів баз даних зображений на рис. 5.19.

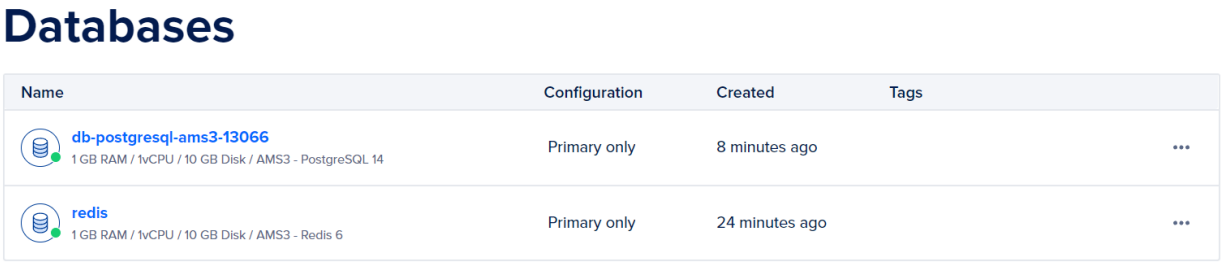

Рисунок 5.19 – Список створених кластерів баз даних

В створених кластерах баз даних у панелі управління кластером можна відслідковувати журнал подій, коли та які запити надійшли до них (рис. 5.20).

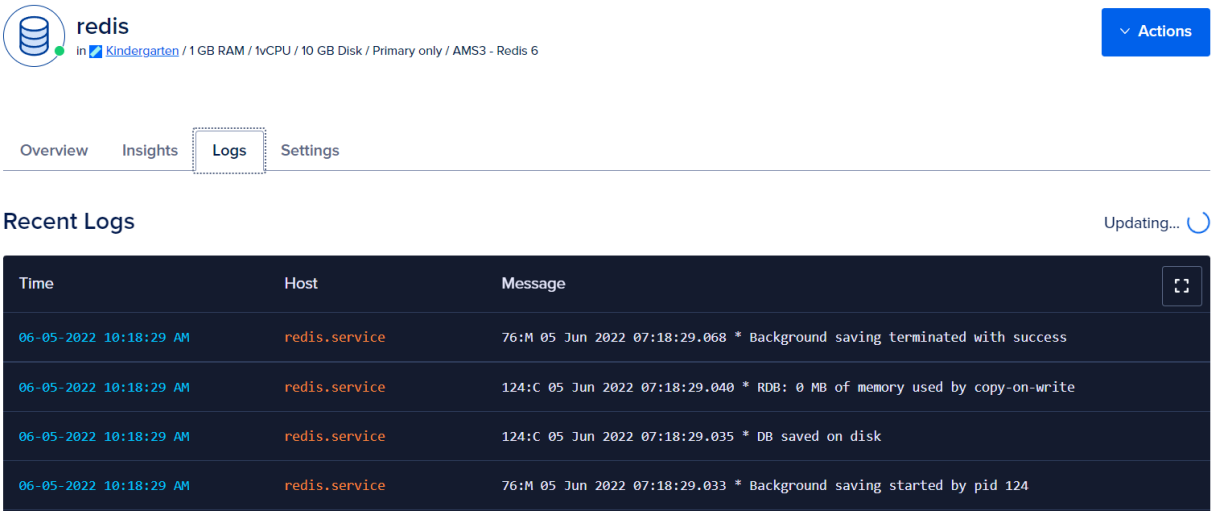

Рисунок 5.20 – Журнал подій у кластері Redis

Після створення основних хмарних ресурсів слід створити вхідну точку для клієнтів, котра буде слугувати публічним маршрутизатором у приватній мережі застосунку з підтримкою криптографічних протоколів, таких як SSL. У ролі вхідної точки слугує балансир навантаження, який можна створити, скориставшись послугою «Load Balancer».

Створення балансиру навантаження наведено на рис. 5.21.

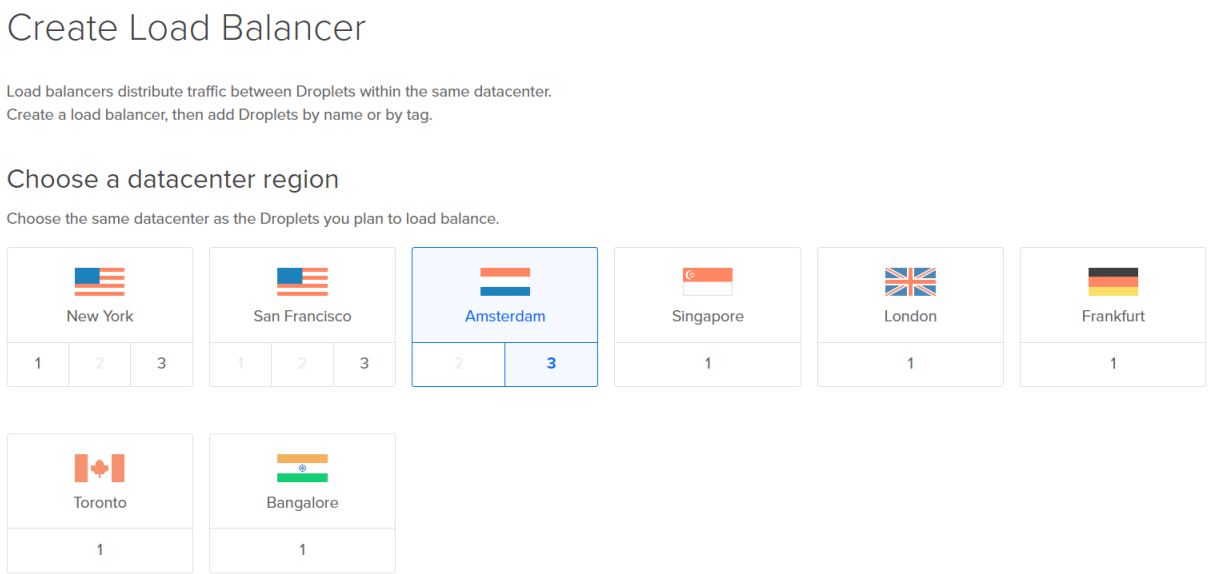

Рисунок 5.21 – Створення балансиру навантаження

Секція «Create Load Balancer», як і в попередніх випадках, перед створенням пропонує обрати регіон розташування балансиру навантаження. У якості регіону обраний AMS3 для спільної взаємодії з інфраструктурою у одній VPC (рис. 5.22).

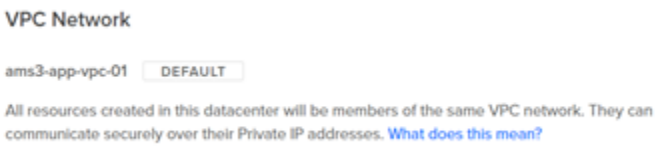

Рисунок 5.22 – Мережа VPC балансиру навантаження

Наступна секція «Scaling configuration». У ній потрібно вказати кількість вузлів для масштабу балансиру навантаження. Вибір кількості доступних вузлів наведений на рис 5.23.

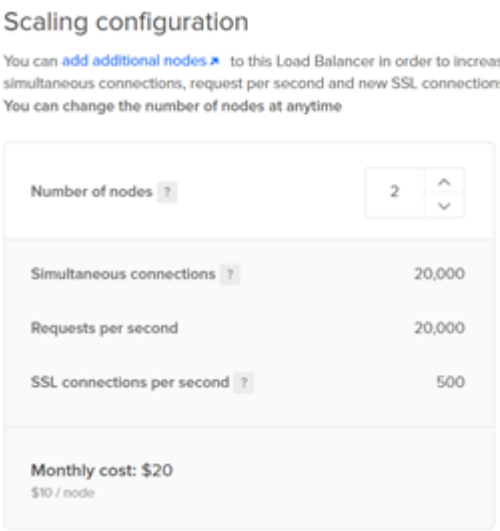

Рисунок 5.23 – Вибір кількості вузлів у балансирі навантаження

Секція «Connect Droplets» пропонує обрати хмарні ресурси «Droplet» по їх назві, або тегу, які треба включити до балансування навантаження та які повинні бути доступними користувачам через нього.

У секції «Forwarding rules» треба вказати протоколи та мережеві порти, які необхідно прокидати між виділеними машинам та балансиром. Тобто, іншими словами, що саме повинно бути доступним через балансир навантаження. Обидві секції зображені на рис. 5.24.

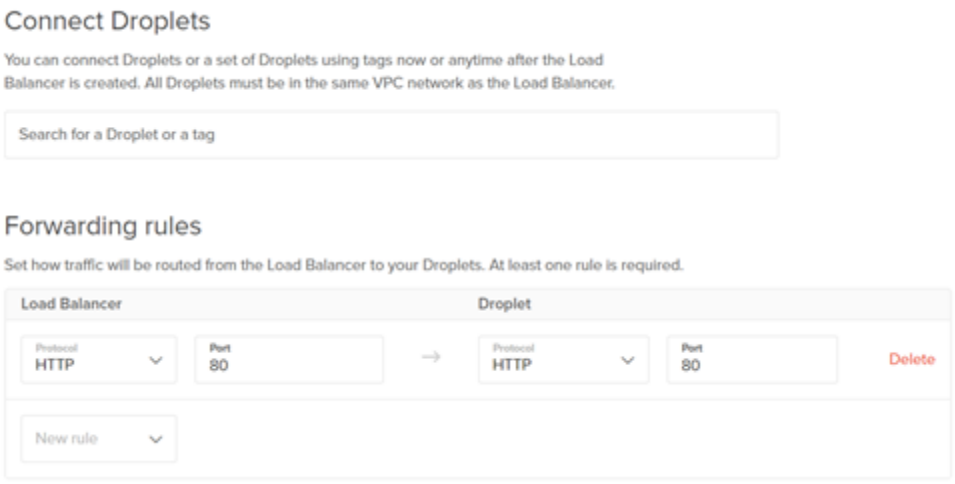

Рисунок 5.24 – Секції «Connect Droplets» та «Forwarding rules»

Секція «Advanced settings» містить опції для розширеної конфігурації балансиру навантаження при його створенні. У цій секції можна вказати:

– мережевий порт для перевірки стану роботи застосунку через інтервальні запити (у даній опції наявна можливість змінити інтервал, кінцеву точку, час очікування відповіді, часовий поріг у стані роботи, або збою);

– тривалість сесій та включення підтримки липких (sticky) сесій;

– включення підтримки перенаправлення HTTP у HTTPS;

– включення протоколу проксування;

– включення підтримки Backend Keepalive для спроби підтримувати постійне з’єднання із працюючим застосунком.

На рис. 5.25 вказані обрані розширенні конфігурації.

У останній секції «Finalize and create» вказуються назва балансиру та його приналежність до проекту (рис. 5.26).

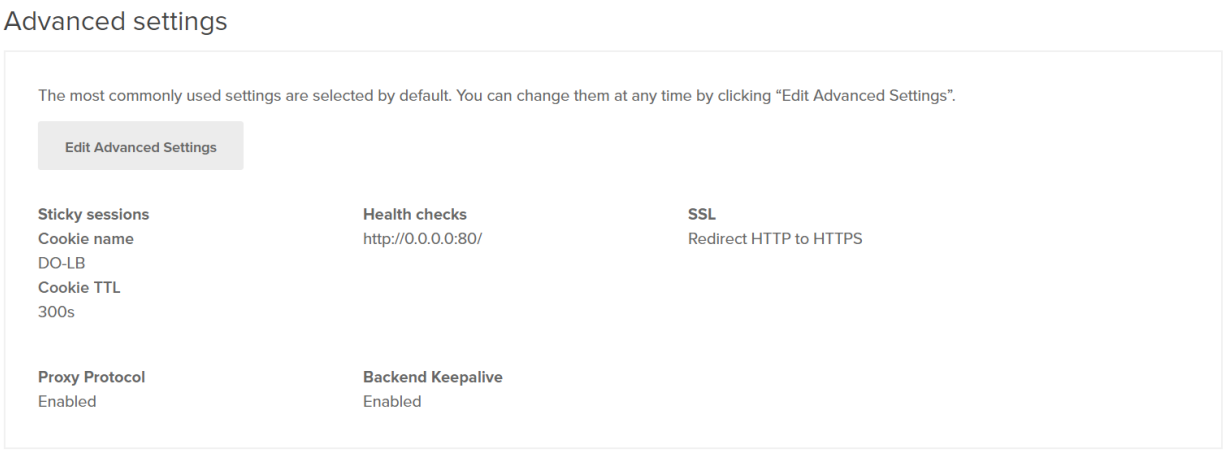

Рисунок 5.25 – Розширена конфігурація балансиру навантаження

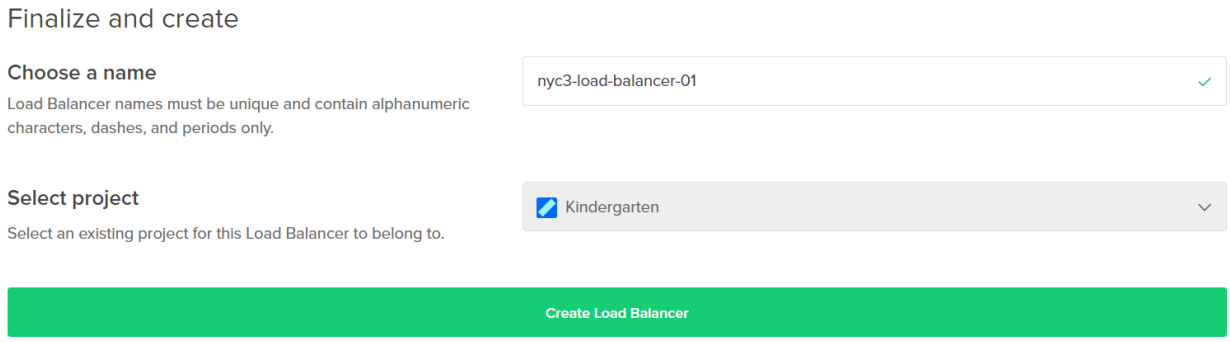

Рисунок 5.26 – Секція «Finalize and create»

5.2 Розгортання застосунку у хмарі

Для розгортання застосунку у хмарі використовується CI/CD інструмент TeamCity на виділеному віртуальному сервері керування інфраструктурою.

TeamCity слід розділяти на дві частини: сервер та агенти. Сервер TeamCity використовується для централізованого керування та формування конфігурацій задач, таких як побудова, тестування та розгортання застосунку. Агенти використовуються як одиниця обробки задач, що надходять від сервера. Кожен агент, зазвичай, представляє окрему машину та може одночасно виконувати лише одну задачу. Використовуючи контейнери, можна розгортати декілька агентів на одній машині, розділяючи їх один від одного.

Для створення агенту використовується наступна команда:

```
docker run -e SERVER_URL="<адреса до серверу>" \
-v <директорія до конфігураційного файла
>:/data/teamcity_agent/conf \ jetbrains/teamcity-agent
```

При створенні серверу TeamCity слід вказати тип бази даних для збереження інформації історії виконаних задач та даних, пов'язаних із користувачами. За замовчуванням, для користування обрано внутрішню базу HSQLDB (рис. 5.27).

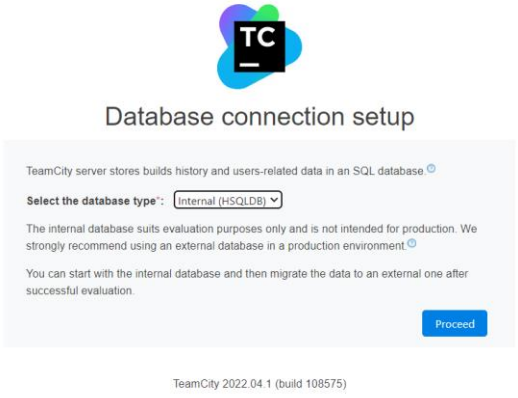

Рисунок 5.27 – Обрання типу баз даних для TeamCity

Наступне, що потрібно зробити після вибору типу бази даних для TeamCity – створити власні адміністраторські облікові данні. Як тільки обліковий запис нового користувача створено, TeamCity пропонує створити новий проект. При створенні проекту, надається дві опції – використання автоматичних налаштувань на основі віддаленого репозиторію системи управління версіями, або ж ручне налаштування. У даній роботі використовується ручне налаштування через те, що репозиторій на базі Git міститься локально у файловій системі агента, без необхідності у додатковій взаємодії із іншими репозиторіями. У секції створення проекту вказується назва, ідентифікатор проекту та його опис (рис 5.28).

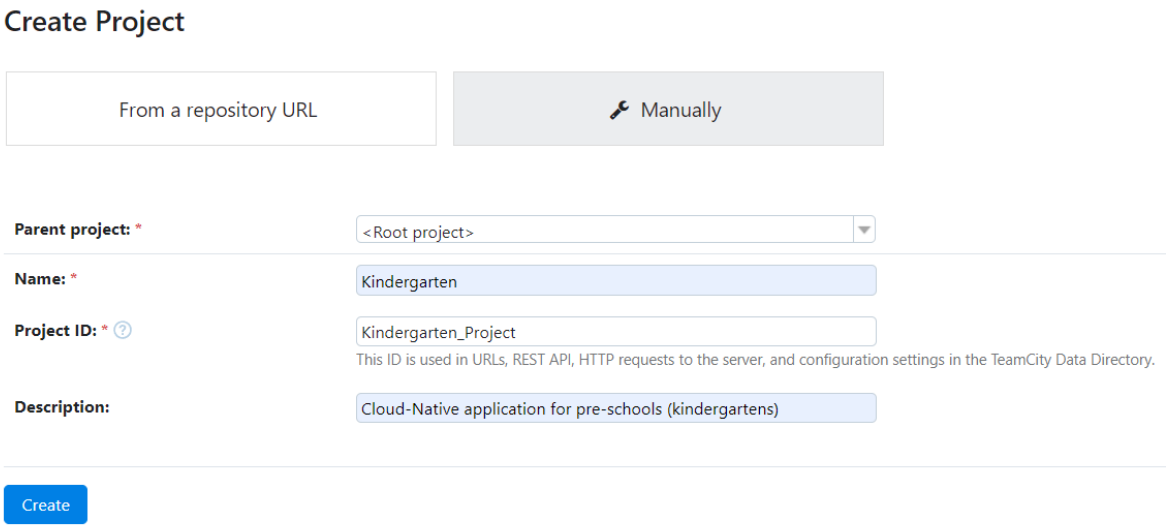

Рисунок 5.28 – Створення проекту у TeamCity

Після того як проект створено, слід налаштувати конфігурацію побудови застосунку (рис. 5.29).

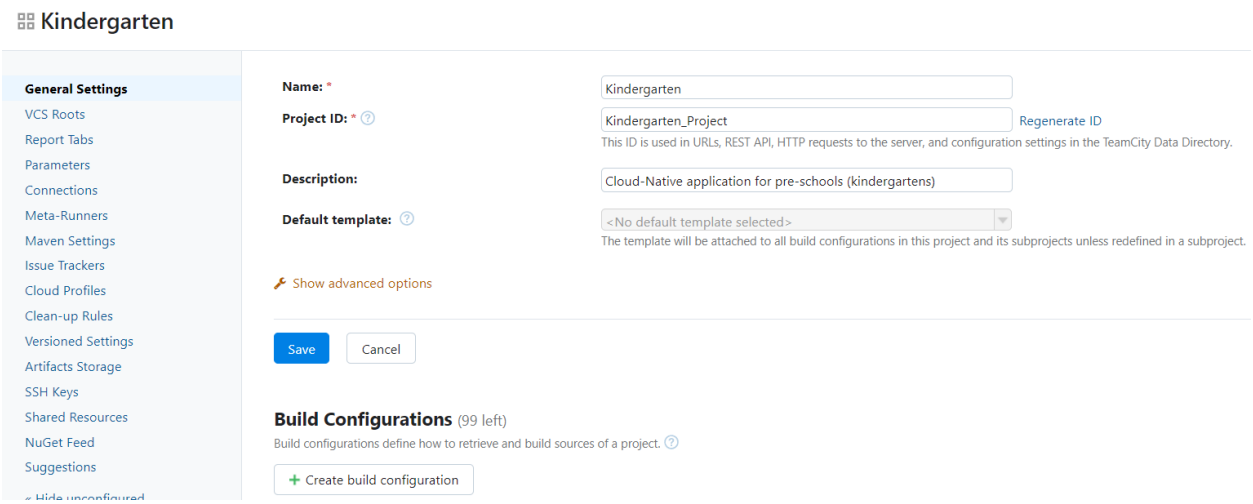

Рисунок 5.29 – Створений проект та його конфігурація

У секції «Create Build Configuration» вказується вибір новоствореного проекту, назва конфігурації, ідентифікатор конфігурації та опис (рис. 5.30).

У створеній конфігурації побудови слід описати сценарії, що будуть виконуватимуться агентами, кроки конфігурації, які йдуть послідовно, або за умови при виконанні. Для створення кроку конфігурації треба натиснути на кнопку «Add build steps» (рис. 5.31).

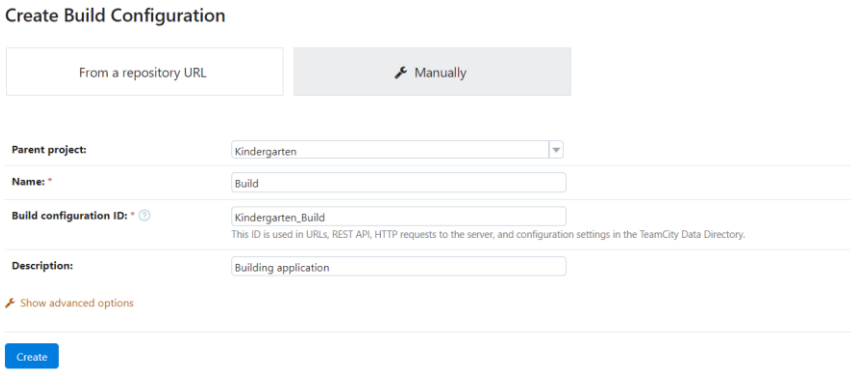

Рисунок 5.30 – Створення Build Configuration

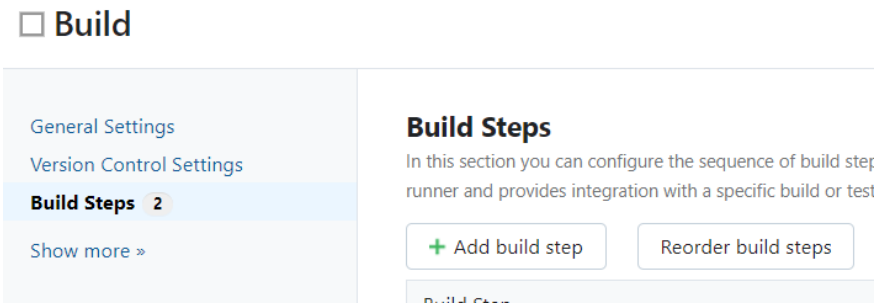

Рисунок 5.31 – Створення кроку конфігурації

При переході до створення кроку конфігурації користувач у контекстному меню повинен обрати тип інструменту для написання його опису дії (рис. 5.32).

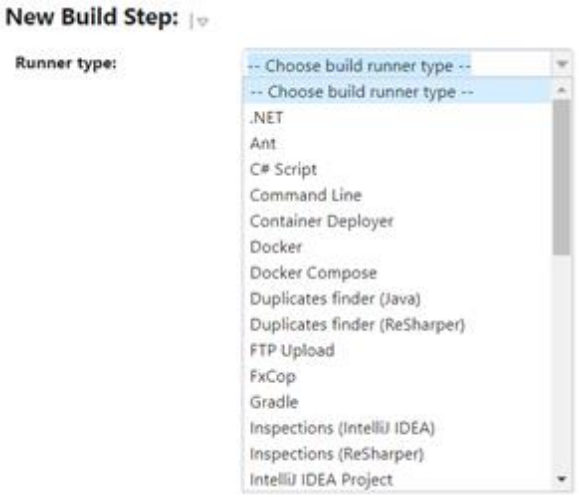

Рисунок 5.32 – Контекстне меню вибору інструменту для кроку конфігурації

У контекстному меню обраний тип інструменту Docker для опису сценарію побудови образу застосунку (рис 5.33). У цьому кроці вказуються назва кроку, правило виконання та тип команди Docker. Оскільки цей крок використовується для побудови образу, тому обрано команду build.

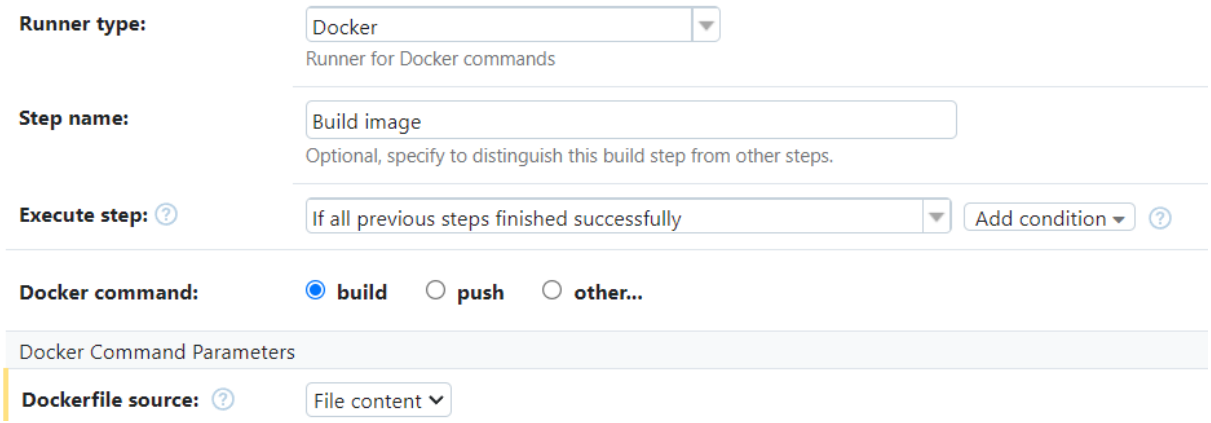

Рисунок 5.33 – Конфігурація Docker

Під параметром джерела Dockerfile можна вказати файл, адресу до нього у мережі, або вставити його вміст у текстове поле. Для вказання Dockerfile використовується його вміст у текстовому полі, який наведений у лістингу 5.1.

```
FROM golang:1.18.3 AS go-builder
WORKDIR /build
COPY go.sum go.mod ./
RUN go mod download
COPY backend
ENV GOOS=linux GOARCH=amd64 CGO_ENABLED=0
RUN go build -ldflags="-w -s" -o api .
FROM gcr.io/distroless/base
COPY --from=go-builder
     ["/build/api", "/build/.env", "/"]
EXPOSE 5000
ENTRYPOINT ["/api"]
```

Лістинг 5.1 – Dockerfile для побудови зображення API застосунку

Розширенні налаштування кроку конфігурації показані на рис. 5.34.

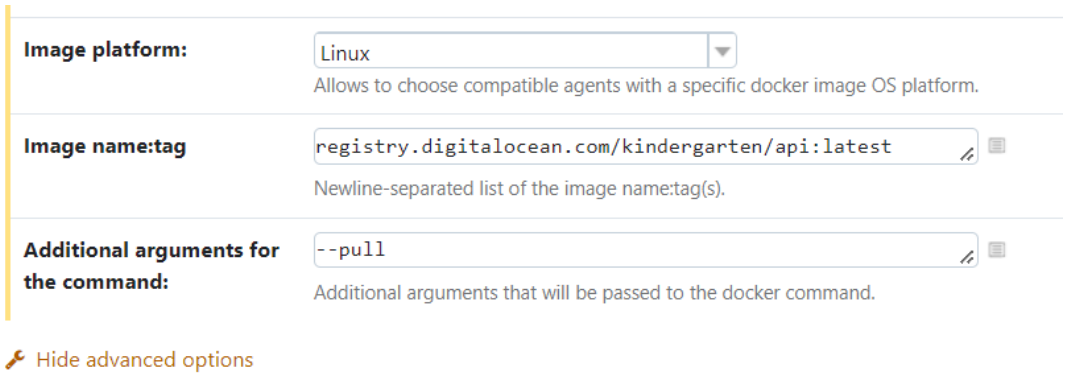

Рисунок 5.34 – Розширенні налаштування кроку конфігурації

Як назва контейнера використовується адреса до власного репозиторію контейнерів, що створений у DigitalOcean. Збудований контейнер використовується у наступному кроці для відправки до цього репозиторію за командою push (рис 5.35).

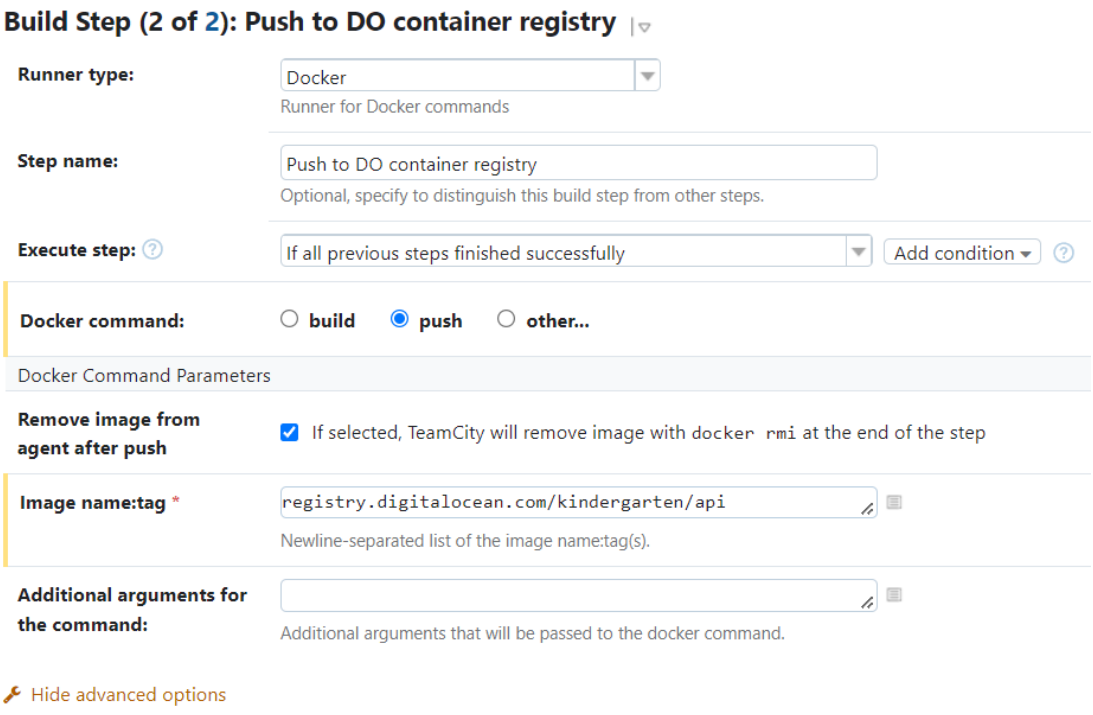

Рисунок 5.35 – Конфігурація відправки контейнеру до репозиторію

Перед відправкою виконано команду з авторизацією до репозиторію через токен API `docker login registry.digitalocean.com`

Інші конфігурації задач TeamCity налаштовуються подібним чином, для побудови частини застосунку, відповідальної за користувальницький інтерфейс, використовується Dockerfile з вмістом, наведеним у лістингу 5.2.

```
FROM python:latest
COPY frontend /
EXPOSE 8000
CMD python -m http.server 8000
```

Лістинг 5.2 – Dockerfile для побудови образу користувальницького інтерфейсу

Збудовані контейнери запускаються віддалено на виділеному сервері застосунків через задачі розгортання, або через командний рядок (лістинг 5.3).

```
podman run -d --rm -p 8000:8000
registry.digitalocean.com/kindergarten/ui
podman run -d --rm -p 5000:5000
registry.digitalocean.com/kindergarten/api
```

Лістинг 5.3 – Запуск контейнерів через командний рядок

Для перенаправлення контейнерів на загальний HTTP порт застосовується веб-сервер Caddy. Його конфігураційний файл показаний у лістингу 5.4.

```
http://localhost:80 {
   reverse_proxy localhost:8080
   reverse_proxy /api/* localhost:5000
   encode gzip
   log {
     output file /var/log/access.log {
       roll_size 1gb
       roll_keep 5
       roll_keep_for 720h }
     format filter {
       wrap console
       fields {request>headers>Authorization delete}
     }   }   }
```

Лістинг 5.4 – Конфігурація веб серверу Caddy

Приклад взаємодії клієнта та хмарного застосунку наведений на рис. 5.36. На ньому підкреслено адресу запитуваного файлу користувача та відповідь цього запиту.

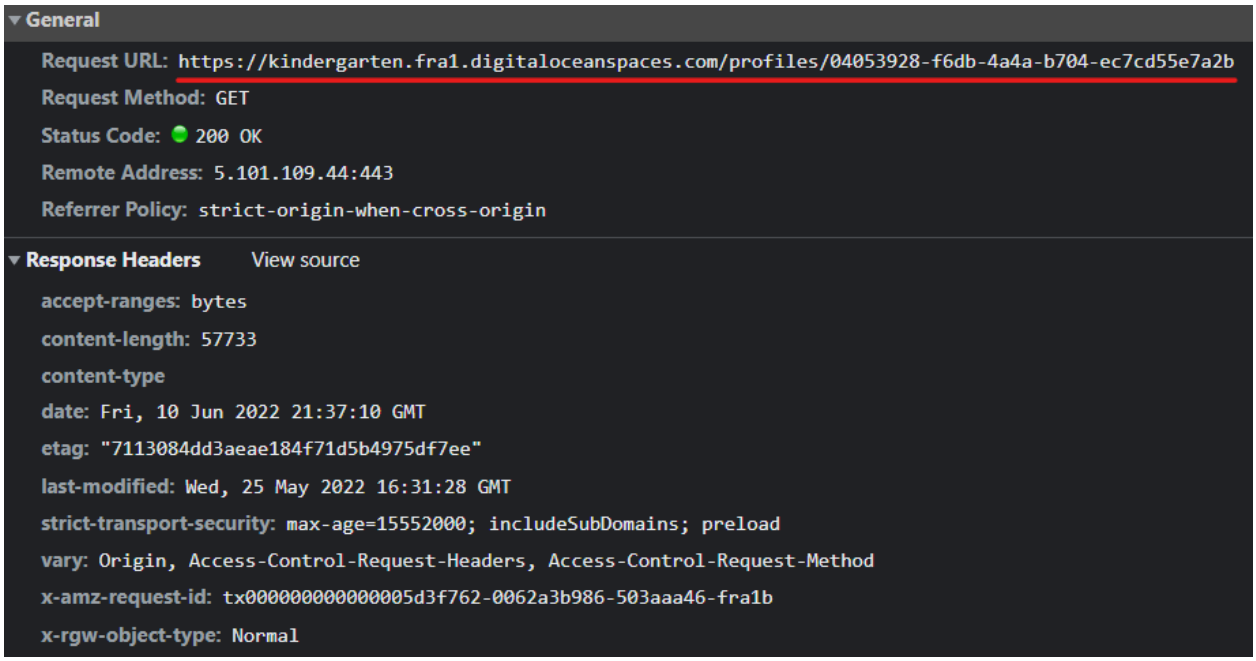

Рисунок 5.36 – Взаємодія клієнта зі зовнішнім хмарним сховищем

На рис. 5.37 підкреслений прикріплений токен доступу, що використовується при надсиланні запиту.

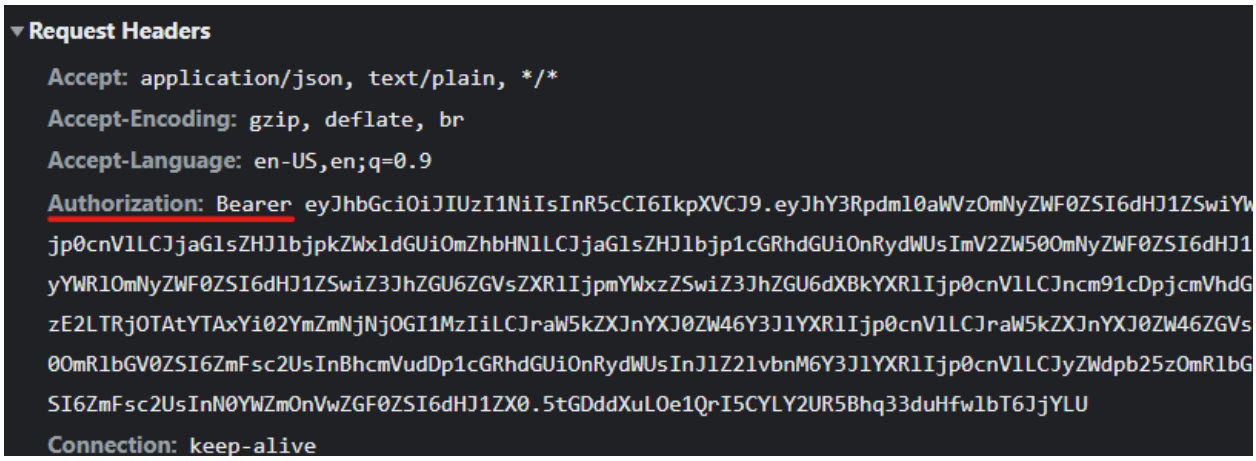

Рисунок 5.37 – Згенерований токен доступу у заголовку HTTP запиту

ВИСНОВКИ

У ході виконання дипломної роботи розглянута предметна область «Заклади дошкільної освіти» та визначені користувачі застосунку (батьки та їх діти, співробітники, адміністратор), сформульовані їх функціональні задачі.

При дослідженні існуючих застосунків для дошкільних навчальних закладів, що дозволяють проводити педагогічний моніторинг і забезпечувати взаємодію дітей, батьків та педагогічних працівників, проведений аналіз систем Ellis та Brightwheel. Розглянута їх функціональність, виокремлені недоліки, зокрема орієнтованість на певні заклади, підтримка регіонального законодавства, а також плата за використання. Виявлено також, що наразі в Україні подібних інформаційних систем взагалі не існує, тому актуальність розробки є беззаперечною.

Сформовані вимоги до проектованого застосунку, спроектована інформаційна модель предметної області «Дошкільні навчальні заклади».

Розглянута доцільність застосування хмарних технологій для створення застосунку, обґрунтований вибір мікросервісної архітектури та архітектурного стилю REST, що застосовані при розробці, реалізовані такі концепції підходу cloud native, як DevOps, CI/CD та контейнеризація.

Спроектована модель хмарної інфраструктури застосунку, що реалізована засобами, наданими постачальником хмарних послуг DigitalOcean. Створені наступні компоненти моделі:

– VPC мережа;

– два виділені віртуальні сервери – один для застосунку та другий для керування інфраструктурою застосунку;

– хмарний кластер СКБД PostgreSQL;

– хмарний кластер розподіленого сховища Redis;

– репозиторій контейнерів;

– зовнішнє хмарне сховище;

– балансир навантаження.

Для створення та розгортання застосунку у хмарі обраний наступний стек технологій: мови програмування Golang та Javascript, фреймворки Fiber та Vue.js, бібліотека компонентів Vuetify, утиліта Axios, система керування базами даних PostgreSQL, розподілене сховище Redis, веб сервер Caddy.

Застосунок будується та розгортається через систему побудови застосунків та неперервної інтеграції Teamcity з використанням контейнеризації за допомогою інструментарію Podman.

Для забезпечення захисту інформаційної системи використаний стандарт JWT токенів та розмежування повноважень користувачів на рівні СКБД. Задля підвищення захисту хмарної інфраструктурі її складові розміщуються у приватній мережі.

Застосування хмарних технологій при створенні застосунку для дошкільних навчальних закладів має такі основні переваги:

– значне зниження витрат дошкільних навчальних закладів на придбання, підтримку і модернізацію програмного забезпечення та комп'ютерної техніки;

– можливість отримувати доступ до застосунку в будь-який час, в будь-якому місці за допомогою комп'ютера або мобільного пристрою, підключеного до мережі Internet;

– створення майданчика для взаємодії педагогічного персоналу та батьків.

Подальший розвиток даного проекту можливий у напрямку покращення інтерфейсу застосунку, наприклад, надання інформації про успіхи дитини у вигляді графіків, діаграм тощо.

Актуальність роботи обговорена під час дев'ятнадцятої Всеукраїнської конференції студентів та молодих науковців «Інформатика, інформаційні системи та технології», тези доповіді опубліковано [4].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ключові показники якості дошкільної освіти. [Електронний ресурс] Режим доступу: https://mon.gov.ua/storage/app/media/doshkilna/2021/01/25/Klyuchovi%20pokaznyky%20yakosti%20doshkilnoyi%20osvity.pdf
2. ELLIS – e-kindergarten – Education Estonia [Електронний ресурс] – Режим доступу: https://www.educationestonia.org/organisation/eliis
3. About brightwheel [Електронний ресурс] – Режим доступу: https://mybrightwheel.com/about
4. Дон С.С, Розновець О.І Cloud Native застосунок для дошкільних навчальних закладів/ Інформатика, інформаційні системи та технології: тези доповідей дев'ятнадцятої всеукраїнської конференції студентів і молодих науковців. // Одеса, 29 квітня 2022 р. – Одеса, 2022.– с.109-110
5. Service-Oriented Architecture [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/service-oriented-architecture
6. What Are Microservices? How Microservices Architecture Works [Електронний ресурс] – Режим доступу: https://middleware.io/blog/microservices-architecture
7. Microservices Communication Design Patterns [Електронний ресурс] – Режим доступу: https://learncsdesign.medium.com/microservices-communication-design-patterns-906ea111c0d0
8. Мікросервісна Архітектура Для Початківців. Частина I [Електронний ресурс] – Режим доступу: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one
9. Мікросервісна архітектура [Електронний ресурс] – Режим доступу: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d
10. Які є конвенції в REST API та для чого їх дотримуватись [Електронний ресурс] – Режим доступу: https://dou.ua/forums/topic/34550
11. Що таке Cloud native додатки і чому вони важливі? [Електронний ресурс] – Режим доступу: https://denovo.ua/blog/cloud-native-apps

12. Understanding DevOps [Електронний ресурс] – Режим доступу: https://www.redhat.com/en/topics/devops
13. What Is CI/CD and How Does It Work? [Електронний ресурс] – Режим доступу: https://www.synopsys.com/glossary/what-is-cicd.html
14. Containerization Explained [Електронний ресурс] – Режим доступу: https://www.ibm.com/cloud/learn/containerization
15. IaaS versus PaaS versus SaaS [Електронний ресурс] – Режим доступу: https://www.ibm.com/cloud/learn/iaas-paas-saas
16. Cloud Computing Providers Comparison | Cloud Computing Companies Comparison. [Електронний ресурс] – Режим доступу: https://www.cloudorado.com/cloud_providers_comparison.jsp
17. Why Golang Is Widely Used in the DevOps and Cloud Native Space? [Електронний ресурс] – Режим доступу: https://thechief.io/c/editorial/why-golang-is-widely-used-in-the-devops-and-cloud-native-space
18. SPA vs MPA: Which One is Better For You? [Електронний ресурс] – Режим доступу: https://yojji.io/blog/spa-vs-mpa
19. The Good and the Bad of Vue.js Framework Programming [Електронний ресурс] – Режим доступу: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js
20. What is PostgreSQL? [Електронний ресурс] – Режим доступу: https://aws.amazon.com/rds/postgresql/what-is-postgresql
21. JSON Web Token Introduction [Електронний ресурс] – Режим доступу: https://jwt.io/introduction
22. What is Podman? – Podman Documentation [Електронний ресурс] – Режим доступу: https://docs.podman.io/en/latest
23. CI/CD Tools Comparison: Jenkins, TeamCity, Bamboo, Travis CI, and More [Електронний ресурс] – Режим доступу: https://www.altexsoft.com/blog/engineering/cicd-tools-comparison

ДОДАТОК А

Задачі користувачів cloud native застосунку для дошкільних навчальних закладів

Таблиця А.1 – Задачі користувачів cloud native застосунку для дошкільних навчальних закладів

Номер	Задача	Вхідні дані	Вихідні дані
Користувач			
К1	Перегляд свого профілю	Користувач	ПІБ, стать, світлина, дата народження, електронна пошта
К2	Редагування облікових даних	Облікові дані користувача (логін, пароль), користувач	Нові облікові дані
К3	Перегляд інформації пов’язаного закладу	Користувач	Назва, коротка назва, код ЄДЕБО, контактні дані (електронна пошта, телефон, сайт, адреса), статус роботи, тип власності
К4	Перегляд інформації про заплановані події	Заклад	Заголовок події, світлина, опис події, дата проведення
К5	Перегляд інформації про власних дітей	Користувач	ПІБ, стать, світлина, дата народження, дата вступу, дата закінчення
К6	Перегляд електронних звітів	Користувач, дитина	Дата, вид діяльності, оцінка
Співробітник			
С1	Редагування облікових даних	Користувач, електронна пошта, псевдонім, пароль	Нові облікові дані
С2	Перегляд свого профілю	Користувач	ПІБ, електронна пошта
С3	Перегляд інформації пов’язаного закладу	Користувач	Назва, коротка назва, код ЄДЕБО, контактні дані (електронна пошта, телефон, сайт, адреса)

Продовження таблиці А.1

Номер	Задача	Вхідні дані	Вихідні дані
С4	Перегляд інформації про заплановані події	Заклад	Заголовок, світлина, опис, дата проведення події
С5	Фільтрування інформації про заплановані події	Деталі про подію	Список подій (назва, світлина, опис, дата проведення)
С6	Створення інформації про заплановану подію	Заголовок події, дата проведення, світлина, опис події, заклад	Нова подія
С7	Редагування інформації про заплановані події	Заголовок події, дата проведення, світлина, опис події, подія	Оновлена інформація про подію
С8	Видалення інформації про заплановані події	Подія	Відсутні
С9	Перегляд інформації про дітей у своїй групі	Користувач	Список дітей: ПІБ, стать, світлина, дата народження, дата входження, дата закінчення, додаткова інформація
С10	Перегляд електронних звітів діяльності дітей	Дитина	Дата, вид діяльності, оцінка
С11	Створення електронних звітів діяльності дитини	Дитина, вид діяльності, оцінка, дата, користувач	Новий електронний звіт діяльності дитини
С12	Редагування електронних звітів діяльності дитини	Електронний звіт, дитина, вид діяльності, оцінка, дата	Оновлений електронний звіт
С13	Перегляд списку документів	Користувач	Список назв та описів документів
С14	Створення документів	Файл, опис документу	Новий документ

Продовження таблиці А.1

Номер	Задача	Вхідні дані	Вихідні дані
С15	Редагування документів	Файл, опис документу	Оновлений документ
С16	Видалення документів	Документ	Відсутні
Адміністратор			
А1	Перегляд списку користувачів	Відсутні	Список користувачів: ПІБ та логін, заклад
А2	Перегляд певних користувачів	Користувач	ПІБ, стать, світлина, дата народження, електронна пошта, логін, заклад
А3	Створення користувача	Облікові дані (логін, пароль), категорія користувача	Новий користувач
А4	Редагування користувача	Користувач	Оновлена інформація користувача
А5	Видалення користувача	Користувач	Відсутні
А6	Перегляд закладів дошкільної освіти	Відсутні	Назва, коротка назва, код ЄДЕБО, контактні дані (електронна пошта, телефон, сайт, адреса), статус роботи, тип власності
А7	Фільтрування закладів дошкільної освіти	Деталі про заклад дошкільної освіти, параметри (назва, адреса, телефон, сайт, адреса, статус роботи, код ЄДЕБО)	Список закладів: назва, коротка назва, код ЄДЕБО, контактні дані (електронна пошта, телефон, сайт, адреса), статус роботи
А8	Створення інформації про заклад дошкільної освіти	Назва, коротка назва, код ЄДЕБО, контактні дані (електронна пошта, телефон, сайт, адреса), статус роботи, тип власності	Новий заклад дошкільної освіти
А9	Редагування інформації про заклад дошкільної освіти	Назва, коротка назва, код ЄДЕБО, статус роботи, тип власності, контактні дані, адреса	Оновлена інформація про заклад дошкільної освіти

Продовження таблиці А.1

Номер	Задача	Вхідні дані	Вихідні дані
А10	Видалення закладу дошкільної освіти	Заклад дошкільної освіти	Відсутні
А11	Фільтрування подій	Дата	Список подій (дата, назва, опис, заклад)
А12	Створення інформації про подію	Заголовок події, дата проведення, світлина, опис події	Нова подія
А13	Редагування інформації про подію	Заголовок події, дата проведення, світлина, опис події	Оновлена інформація про подію
А14	Видалення події	Подія	Відсутні
А15	Внесення нового населеного пункту	Область, район, назва населеного пункту	Новий населений пункт
А16	Редагування інформації про населений пункт	Область, район, назва населеного пункту	Оновлена інформація про населений пункт
А17	Видалення населеного пункту	Населений пункт	Відсутні
А18	Перегляд співробітників	Відсутні	Список співробітників
А19	Фільтрування списку співробітників	Параметри співробітників (логін, ПІБ)	Відфільтрований список співробітників (логін, ПІБ, електронна пошта, заклад)
А20	Створення інформації про нового співробітника	ФІО, стать, дата народження, посада, приналежність до певного закладу	Новий співробітник
А21	Редагування інформації про співробітника	ФІО, стать, дата народження, посада, приналежність до певного закладу	Оновлена інформація про співробітника
А22	Видалення співробітника	Співробітник	Відсутні

ДОДАТОК Б

ER-діаграма бази даних cloud native застосунку для дошкільних навчальних закладів

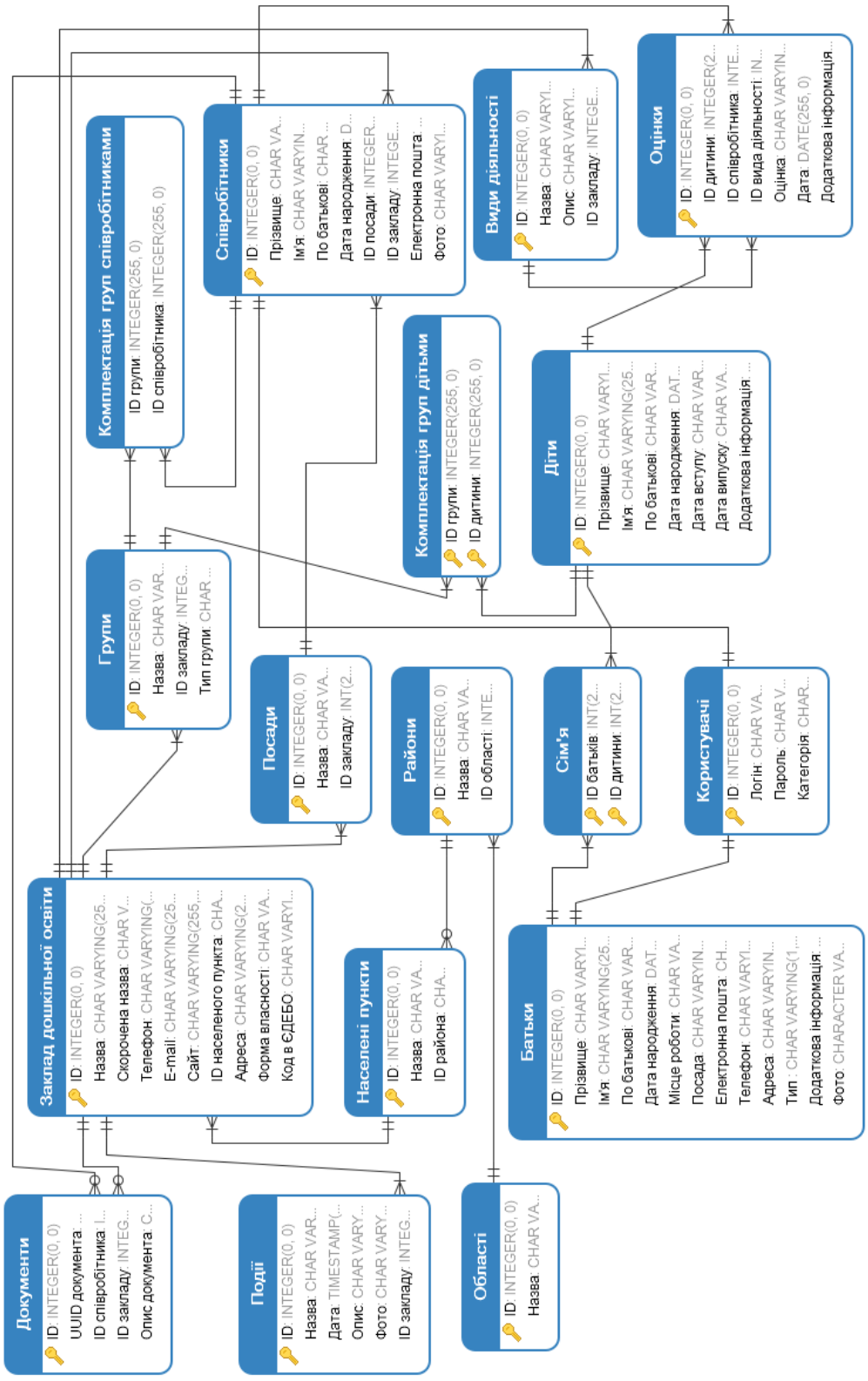

Рисунок Б.1 – ER-діаграма бази даних cloud native застосунку для дошкільних навчальних закладів

ДОДАТОК В

SQL запити для створення бази даних

```
CREATE ROLE "ADMIN" WITH LOGIN ENCRYPTED PASSWORD '{ПАРОЛЬ}';
CREATE ROLE "STAFF" WITH LOGIN ENCRYPTED PASSWORD '{ПАРОЛЬ}';
CREATE ROLE "DEFAULT" WITH LOGIN ENCRYPTED PASSWORD '{ПАРОЛЬ}';
```

Лістинг В.1 – Створення категорій користувачів

```
CREATE TYPE "GRADE" AS ENUM ('1', '2', '3', '4', '5');
CREATE TYPE "CATEGORY" AS ENUM ('ADMIN', 'STAFF', 'USER');
CREATE TYPE "GENDER" AS ENUM ('M', 'F', 'O', 'B');
CREATE TYPE "PARENT_TYPE" AS ENUM ('БАТЬКО', 'МАТИ');
CREATE TYPE "OWNERSHIP_TYPE" AS ENUM ('ПРИВАТНА', 'КОМУНАЛЬНА',
'МУНІЦИПАЛЬНА', 'ДЕРЖАВНА', 'КОЛЕКТИВНА');
```

Лістинг В.2 – Створення перелічувальних типів

```
CREATE TABLE "USERS" (
  ID UUID DEFAULT GEN_RANDOM_UUID() PRIMARY KEY,
  PROFILE_ID UUID,
  EMAIL VARCHAR(127) NOT NULL CHECK(EMAIL ~ '^[A-ZA-Z0-
9._%-]+@[A-ZA-Z0-9.-]+[.][A-ZA-Z]+$'),
  USERNAME VARCHAR(32) NOT NULL,
  PASSWORD VARCHAR(255) NOT NULL CHECK(LENGTH(PASSWORD) >= 8),
  CATEGORY CATEGORY NOT NULL,
  PICTURE UUID DEFAULT('E25D11F3-0557-445A-9A71-B19771B8E848'),
  CREATED_AT TIMESTAMP WITH TIME ZONE DEFAULT NOW() NOT NULL,
  UPDATED_AT TIMESTAMP WITH TIME ZONE DEFAULT NOW() NOT NULL,
  UNIQUE(PROFILE_ID, USERNAME, EMAIL)
);
```

Лістинг В.3 – Створення таблиці «Користувачі»

```
CREATE TABLE "KINDERGARTENS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  SHORT_NAME VARCHAR(255),
  INSTITUTION_CODE INT NOT NULL,
  STATUS BOOL NOT NULL DEFAULT(TRUE),
  OWNERSHIP OWNERSHIP_TYPE,
```

Лістинг В.4 – Створення таблиці «Заклади дошкільної освіти»

```
  EMAIL VARCHAR(255) DEFAULT('') CHECK(EMAIL ~ '^[A-ZA-Z0-
9._%-]+@[A-ZA-Z0-9.-]+[.][A-ZA-Z]+$' OR EMAIL ~ ''),
  TEL_NO VARCHAR(255) DEFAULT(''),
  WEBSITE VARCHAR(255) DEFAULT('') CHECK (WEBSITE ~ '.*?\.[A-
Z]+\/' OR WEBSITE ~ ''),
  LOCALITY_ID INT NOT NULL,
  ADDRESS VARCHAR(255) NOT NULL,
  UNIQUE(ADDRESS, EMAIL, WEBSITE, TEL_NO, INSTITUTION_CODE)
);
```

Лістинг В.4 , лист 2

```
CREATE TABLE "GROUPS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  KINDERGARTEN_ID INT NOT NULL,
  GROUP_TYPE VARCHAR(255)
);
```

Лістинг В.5 – Створення таблиці «Групи»

```
CREATE TABLE "STAFF_GROUPS" (
  GROUP_ID INT NOT NULL,
  STAFF_ID INT NOT NULL
);
```

Лістинг В.6 – Створення таблиці «Групи співробітників»

```
CREATE TABLE "LOCALITY" (
  ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  DISTRICT_ID INT NOT NULL,
  UNIQUE(NAME)
);
```

Лістинг В.7 – Створення таблиці «Населенні пункти»

```
CREATE TABLE "POSITIONS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  KINDERGARTEN_ID INT
);
```

Лістинг В.8 – Створення таблиці «Посади»

```
CREATE TABLE "STAFFS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  SURNAME VARCHAR(255) NOT NULL,
  NAME VARCHAR(255) NOT NULL,
  PATRONYMIC VARCHAR(255) NOT NULL,
  GENDER GENDER NOT NULL,
  DOB DATE NOT NULL,
  POSITION_ID INT NOT NULL,
  KINDERGARTEN_ID INT NOT NULL,
  PROFILE_ID UUID DEFAULT GEN_RANDOM_UUID(),
  UNIQUE(PROFILE_ID) );
```

Лістинг В.9 – Створення таблиці «Співробітники»

```
CREATE TABLE "REGIONS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  UNIQUE(NAME)
);
```

Лістинг В.10 – Створення таблиці «Області»

```
CREATE TABLE "DISTRICTS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  REGION_ID INT NOT NULL,
  UNIQUE(NAME));
```

Лістинг В.11 – Створення таблиці «Райони»

```
CREATE TABLE "CHILDREN_GROUPS" (
  GROUP_ID INT NOT NULL,
  CHILDREN_ID INT NOT NULL );
```

Лістинг В.12 – Створення таблиці «Групи дітей»

```
CREATE TABLE "ACTIVITIES" (
  ID SERIAL NOT NULL PRIMARY KEY,
  NAME VARCHAR(255) NOT NULL,
  DESCRIPTION VARCHAR(510) NOT NULL,
  KINDERGARTEN_ID INT );
```

Лістинг В.13 – Створення таблиці «Діяльності»

```
CREATE TABLE "PARENTS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  SURNAME VARCHAR(255) NOT NULL,
  NAME VARCHAR(255) NOT NULL,
  PATRONYMIC VARCHAR(255) NOT NULL,
  GENDER GENDER NOT NULL,
  DOB DATE NOT NULL,
  POSITION VARCHAR(255) NOT NULL,
  TEL_NO VARCHAR(255) DEFAULT('') CHECK(TEL_NO ~
     '^[\+]?[()]?[0-9]{3}[)]?[-\S\.]?[0-9]{3}
     [-\S\.]?[0-9]{4,6}$' OR TEL_NO ~ ''),
  ADDRESS VARCHAR(255) NOT NULL,
  PARENT_TYPE PARENT_TYPE NOT NULL,
  PROFILE_ID UUID DEFAULT GEN_RANDOM_UUID(),
  ADDITIONAL_INFORMATION VARCHAR(255),
  UNIQUE(PROFILE_ID)
 );
```

Лістинг В.14 – Створення таблиці «Батьки»

```
CREATE TABLE "FAMILY" (
  PARENT_ID INT NOT NULL,
  CHILDREN_ID INT NOT NULL
);
```

Лістинг В.15 – Створення таблиці «Сім’я»

```
CREATE TABLE "CHILDRENS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  SURNAME VARCHAR(255) NOT NULL,
  NAME VARCHAR(255) NOT NULL,
  PATRONYMIC VARCHAR(255) NOT NULL,
  GENDER GENDER NOT NULL,
  DOB DATE NOT NULL,
  ENTRY_DATE DATE NOT NULL,
  END_DATE DATE,
  PICTURE VARCHAR(255),
  ADDITIONAL_INFORMATION VARCHAR(255)
);
```

Лістинг В.16 – Створення таблиці «Діти»

```
CREATE TABLE "GRADES" (
  ID SERIAL NOT NULL PRIMARY KEY,
  CHILDREN_ID INT,
  STAFF_ID INT,
  ACTIVITY_ID INT,
  GRADE GRADE,
  DATE DATE
);
```

Лістинг В.17 – Створення таблиці «Оцінки»

```
CREATE TABLE "EVENTS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  TITLE VARCHAR(255) NOT NULL,
  DESCRIPTION VARCHAR(1020),
  PICTURE VARCHAR(255),
  KINDERGARTEN_ID INT,
  DATE DATE NOT NULL
);
```

Лістинг В.18 – Створення таблиці «Події»

```
CREATE TABLE "DOCUMENTS" (
  ID SERIAL NOT NULL PRIMARY KEY,
  DOCUMENT_ID UUID NOT NULL,
  STAFF_ID INT,
  KINDERGARTEN_ID INT,
  DESCRIPTION VARCHAR(255),
  UNIQUE(DOCUMENT_ID)
);
```

Лістинг В.19 – Створення таблиці «Документи»

```
ALTER TABLE ONLY KINDERGARTENS
     ADD CONSTRAINT LOCALITY_ID_FKEY FOREIGN KEY (LOCALITY_ID)
REFERENCES LOCALITY(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY GROUPS
     ADD CONSTRAINT KINDERGARTEN_ID_FKEY FOREIGN KEY
(KINDERGARTEN_ID) REFERENCES KINDERGARTENS(ID) ON UPDATE CASCADE
ON DELETE RESTRICT;

ALTER TABLE ONLY STAFF_GROUPS
```

Лістинг В.20 – Створення зовнішніх ключів

```
     ADD CONSTRAINT GROUP_ID_FKEY FOREIGN KEY (GROUP_ID)
REFERENCES GROUPS(ID) ON UPDATE CASCADE ON DELETE RESTRICT,
     ADD CONSTRAINT STAFF_ID_FKEY FOREIGN KEY (GROUP_ID)
REFERENCES STAFFS(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY LOCALITY
     ADD CONSTRAINT DISTRICTS_ID_FKEY FOREIGN KEY (DISTRICT_ID)
REFERENCES DISTRICTS(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY POSITIONS
     ADD CONSTRAINT KINDERGARTEN_ID_FKEY FOREIGN KEY
(KINDERGARTEN_ID) REFERENCES KINDERGARTENS(ID) ON UPDATE CASCADE
ON DELETE RESTRICT;

ALTER TABLE ONLY STAFFS
     ADD CONSTRAINT POSITION_ID_FKEY FOREIGN KEY (POSITION_ID)
REFERENCES POSITIONS(ID) ON UPDATE CASCADE ON DELETE RESTRICT,
     ADD CONSTRAINT KINDERGARTEN_ID_FKEY FOREIGN KEY
(KINDERGARTEN_ID) REFERENCES KINDERGARTENS(ID) ON UPDATE CASCADE
ON DELETE RESTRICT;

ALTER TABLE ONLY DISTRICTS
     ADD CONSTRAINT REGION_ID_FKEY FOREIGN KEY (REGION_ID)
REFERENCES REGIONS(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY CHILDREN_GROUPS
     ADD CONSTRAINT GROUP_ID_FKEY FOREIGN KEY (GROUP_ID)
REFERENCES GROUPS(ID) ON UPDATE CASCADE ON DELETE RESTRICT,
     ADD CONSTRAINT CHILDREN_ID_FKEY FOREIGN KEY (CHILDREN_ID)
REFERENCES CHILDRENS(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY ACTIVITIES
     ADD CONSTRAINT KINDERGARTEN_ID_FKEY FOREIGN KEY
(KINDERGARTEN_ID) REFERENCES KINDERGARTENS(ID) ON UPDATE CASCADE
ON DELETE RESTRICT;

ALTER TABLE ONLY FAMILY
    ADD CONSTRAINT CHILDREN_ID_FKEY FOREIGN KEY (CHILDREN_ID)
REFERENCES CHILDRENS(ID) ON UPDATE CASCADE ON DELETE RESTRICT,
     ADD CONSTRAINT PARENT_ID_FKEY FOREIGN KEY (PARENT_ID)
REFERENCES PARENTS(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY GRADES
```

Лістинг В.20 , лист 2

```
    ADD CONSTRAINT ACTIVITY_ID_FKEY FOREIGN KEY (ACTIVITY_ID)
REFERENCES ACTIVITIES(ID) ON UPDATE CASCADE ON DELETE RESTRICT,
     ADD CONSTRAINT CHILDREN_ID_FKEY FOREIGN KEY (CHILDREN_ID)
REFERENCES CHILDRENS(ID) ON UPDATE CASCADE ON DELETE RESTRICT,
     ADD CONSTRAINT STAFF_ID_FKEY FOREIGN KEY (STAFF_ID)
REFERENCES STAFFS(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY DOCUMENTS
    ADD CONSTRAINT KINDERGARTEN_ID_FKEY FOREIGN KEY
(KINDERGARTEN_ID) REFERENCES KINDERGARTENS(ID) ON UPDATE CASCADE
ON DELETE RESTRICT,
     ADD CONSTRAINT STAFF_ID_FKEY FOREIGN KEY (STAFF_ID)
REFERENCES STAFFS(ID) ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE ONLY EVENTS
    ADD CONSTRAINT KINDERGARTEN_ID_FKEY FOREIGN KEY
(KINDERGARTEN_ID) REFERENCES KINDERGARTENS(ID) ON UPDATE CASCADE
ON DELETE RESTRICT;
```

Лістинг В.20 , лист 3

```
CREATE VIEW FAMILY_VIEW
AS
SELECT
       P.PROFILE_ID,
       P.ID AS "PARENT - ID",
       P.SURNAME AS "PARENT - SURNAME",
       P.NAME AS "PARENT - NAME",
       P.PATRONYMIC AS "PARENT - PATRONYMIC",
       P.PARENT_TYPE AS "PARENT - TYPE",
       CH.ID AS "CHILDREN - ID",
       CH.SURNAME AS "CHILDREN - SURNAME",
       CH.NAME AS "CHILDREN - NAME",
       CH.PATRONYMIC AS "CHILDREN - PATRONYMIC",
       CH.ENTRY_DATE AS "CHILDREN - ENTRY DATE",
       CH.END_DATE AS "CHILDREN - END DATE"
FROM FAMILY F
  INNER JOIN PARENTS P ON P.ID = F.PARENT_ID
  INNER JOIN CHILDRENS CH ON CH.ID = F.CHILDREN_ID
GROUP BY P.ID, P.SURNAME, P.NAME, P.PATRONYMIC, P.PARENT_TYPE,
P.PROFILE_ID, CH.ID, CH.SURNAME, CH.NAME, CH.PATRONYMIC
ORDER BY  P.PROFILE_ID;
```

Лістинг В.21 – Створення представлення повної інформації про сім’ю

```
CREATE VIEW CHILDRENS_BY_PARENTS_PROFILE_VIEW AS
SELECT
       CH.ID,
       CH.SURNAME,
       CH.NAME,
       CH.PATRONYMIC,
       CH.GENDER,
       CH.ENTRY_DATE,
       CH.END_DATE,
       CH.ADDITIONAL_INFORMATION,
       P.PROFILE_ID
FROM FAMILY F INNER JOIN PARENTS P ON P.ID = F.PARENT_ID
       LEFT JOIN CHILDRENS CH ON CH.ID = F.CHILDREN_ID
GROUP BY P.PROFILE_ID, CH.ID, CH.SURNAME, CH.NAME,
CH.PATRONYMIC, CH.ENTRY_DATE, CH.END_DATE,
CH.ADDITIONAL_INFORMATION
ORDER BY CH.ID;
```

Лістинг В.22 – Створення представлення для виведення усіх дітей у сім’ї за унікальним ідентифікатором профілю одного із батьків

```
CREATE VIEW STAFF_USER_VIEW AS
SELECT
       U.ID,
       U.PROFILE_ID,
       U.PICTURE,
       U.EMAIL,
       U.USERNAME,
       S.SURNAME,
       S.NAME,
       S.PATRONYMIC,
       S.DOB,
       S.POSITION_ID,
       S.KINDERGARTEN_ID
FROM USERS U INNER JOIN STAFFS S ON U.PROFILE_ID = S.PROFILE_ID
GROUP BY
       U.ID, U.PROFILE_ID, U.PICTURE, U.EMAIL, U.USERNAME,
       S.ID, S.SURNAME, S.NAME, S.PATRONYMIC, S.DOB,
S.POSITION_ID, S.KINDERGARTEN_ID
ORDER BY
       U.ID;
```

Лістинг В.23 – Створення представлення повної інформації про співробітників

```
CREATE EXTENSION IF NOT EXISTS PGCRYPTO;
CREATE EXTENSION IF NOT EXISTS "UUID-OSSP";
```

Лістинг В.24 – Підключення розширень для роботи з криптографією

```
SET CLIENT_ENCODING = 'UTF8';
```

Лістинг В.25 – Встановлення кодування UTF-8

```
CREATE OR REPLACE FUNCTION ENCRYPT_PASSWORD() RETURNS TRIGGER AS
$FUNC$
BEGIN
 NEW.PASSWORD := CRYPT(NEW.PASSWORD, GEN_SALT('BF'));
 RETURN NEW;
END
$FUNC$ LANGUAGE PLPGSQL;
```

Лістинг В.26 – Створення функції шифрування паролів

```
CREATE TRIGGER ENCRYPT_USER_DATA
BEFORE INSERT OR UPDATE ON USERS
FOR EACH ROW
EXECUTE PROCEDURE ENCRYPT_PASSWORD();
CREATE OR REPLACE FUNCTION CREATE_CLIENT() RETURNS TRIGGER AS
$FUNC$
BEGIN
 NEW.EMAIL := CRYPT(NEW.CC_NUM, GEN_SALT('BF'));
 RETURN NEW;
END
$FUNC$ LANGUAGE PLPGSQL;
```

Лістинг В.27 – Створення тригеру шифрування паролів

```
GRANT SELECT ON ACTIVITIES TO "DEFAULT";
GRANT SELECT ON CHILDREN_GROUPS TO "DEFAULT";
GRANT SELECT,UPDATE ON CHILDRENS TO "DEFAULT";
GRANT SELECT ON DISTRICTS TO "DEFAULT";
GRANT SELECT ON EVENTS TO "DEFAULT";
GRANT SELECT,UPDATE ON FAMILY TO "DEFAULT";
GRANT SELECT ON GRADES TO "DEFAULT";
GRANT SELECT ON GROUPS TO "DEFAULT";
GRANT SELECT ON KINDERGARTENS TO "DEFAULT";
```

Лістинг В.28 – Надання прав доступу батькам

```
GRANT SELECT ON LOCALITY TO "DEFAULT";
GRANT SELECT,UPDATE ON PARENTS TO "DEFAULT";
GRANT SELECT ON POSITIONS TO "DEFAULT";
GRANT SELECT ON REGIONS TO "DEFAULT";
GRANT SELECT ON STAFF_GROUPS TO "DEFAULT";
GRANT SELECT ON STAFFS TO "DEFAULT";
GRANT SELECT,UPDATE ON USERS TO "DEFAULT";
GRANT SELECT ON CHILDRENS_BY_PARENTS_PROFILE_VIEW TO "DEFAULT";
GRANT SELECT ON FAMILY_VIEW TO "DEFAULT";
GRANT SELECT ON STAFF_USER_VIEW TO "DEFAULT";
```

Лістинг В.28, лист 2

```
GRANT INSERT,SELECT,UPDATE ON ACTIVITIES TO "STAFF";
GRANT SELECT ON CHILDREN_GROUPS TO "STAFF";
GRANT INSERT,SELECT,UPDATE ON CHILDRENS TO "STAFF";
GRANT SELECT ON DISTRICTS TO "STAFF";
GRANT INSERT,SELECT,UPDATE,DELETE ON DOCUMENTS  TO "STAFF";
GRANT INSERT,SELECT,UPDATE,DELETE ON EVENTS TO "STAFF";
GRANT INSERT,SELECT,UPDATE ON FAMILY TO "STAFF";
GRANT INSERT,SELECT,UPDATE,DELETE ON GRADES TO "STAFF";
GRANT INSERT,SELECT,UPDATE ON GROUPS TO "STAFF";
GRANT SELECT,UPDATE ON KINDERGARTENS TO "STAFF";
GRANT SELECT ON LOCALITY TO "STAFF";
GRANT INSERT,SELECT,UPDATE ON PARENTS TO "STAFF";
GRANT INSERT,SELECT,UPDATE ON POSITIONS TO "STAFF";
GRANT SELECT ON REGIONS TO "STAFF";
GRANT SELECT,UPDATE ON STAFF_GROUPS TO "STAFF";
GRANT SELECT ON STAFFS TO "STAFF";
GRANT SELECT,UPDATE ON USERS TO "STAFF";
GRANT SELECT ON CHILDRENS_BY_PARENTS_PROFILE_VIEW TO "STAFF";
GRANT SELECT ON FAMILY_VIEW TO "STAFF";
GRANT SELECT ON STAFF_USER_VIEW TO "STAFF";
```

Лістинг В.29 – Надання прав доступу співробітникам

```
GRANT ALL ON ACTIVITIES TO "ADMIN";
GRANT ALL ON CHILDREN_GROUPS TO "ADMIN";
GRANT ALL ON CHILDRENS TO "ADMIN";
GRANT ALL ON DISTRICTS TO "ADMIN";
GRANT ALL ON DOCUMENTS TO "ADMIN";
GRANT ALL ON EVENTS TO "ADMIN";
```

Лістинг В.30 – Надання прав доступу адміністратору

```
GRANT ALL ON FAMILY TO "ADMIN";
GRANT ALL ON GRADES TO "ADMIN";
GRANT ALL ON GROUPS TO "ADMIN";
GRANT ALL ON KINDERGARTENS TO "ADMIN";
GRANT ALL ON LOCALITY TO "ADMIN";
GRANT ALL ON PARENTS TO "ADMIN";
GRANT ALL ON POSITIONS TO "ADMIN";
GRANT ALL ON REGIONS TO "ADMIN";
GRANT ALL ON STAFF_GROUPS TO "ADMIN";
GRANT ALL ON STAFFS TO "ADMIN";
GRANT ALL ON USERS TO "ADMIN";
GRANT SELECT ON CHILDRENS_BY_PARENTS_PROFILE_VIEW TO "ADMIN";
GRANT SELECT ON FAMILY_VIEW TO "ADMIN";
GRANT SELECT ON STAFF_USER_VIEW TO "ADMIN";
```

Лістинг В.30, лист 2

ДОДАТОК Г

Кінцеві точки інтерфейсу програмування застосунку

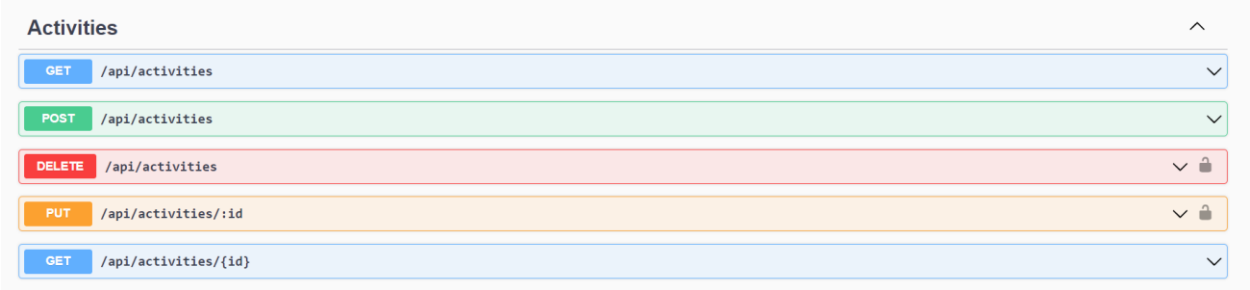

Рисунок Г.1 – Сутність «Діяльність»

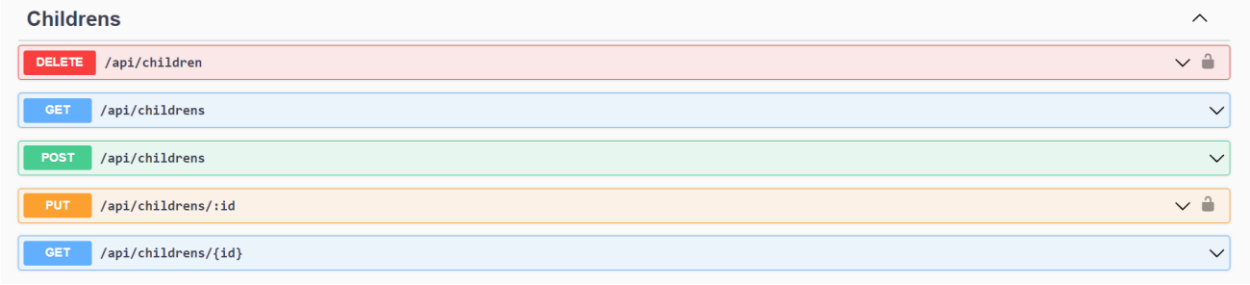

Рисунок Г.2 – Сутність «Діти»

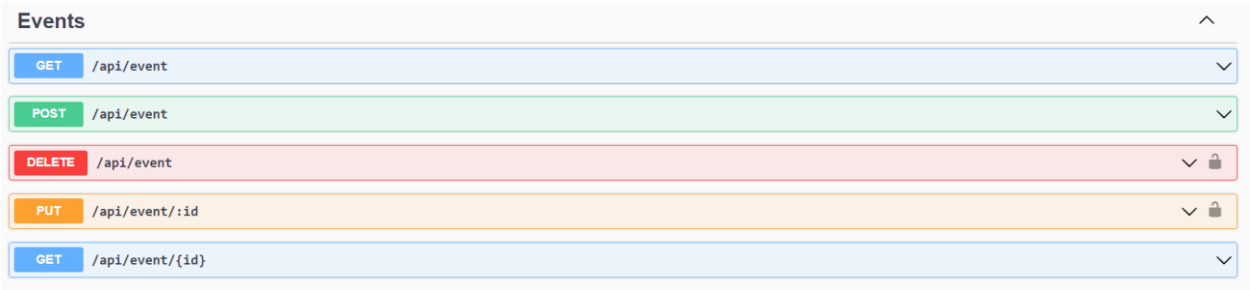

Рисунок Г.3 – Сутність «Події»

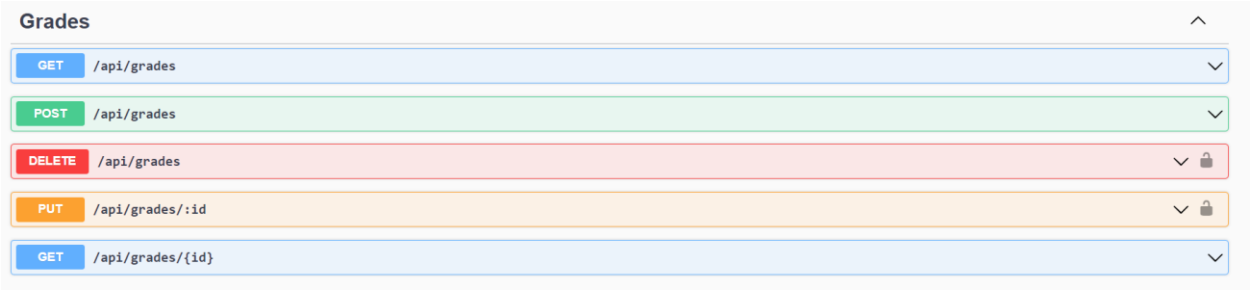

Рисунок Г.4 – Сутність «Оцінки»

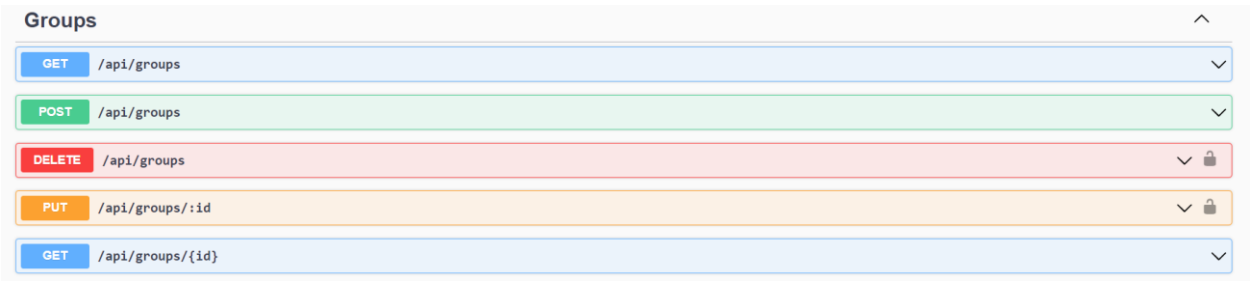

Рисунок Г.5 – Сутність «Групи»

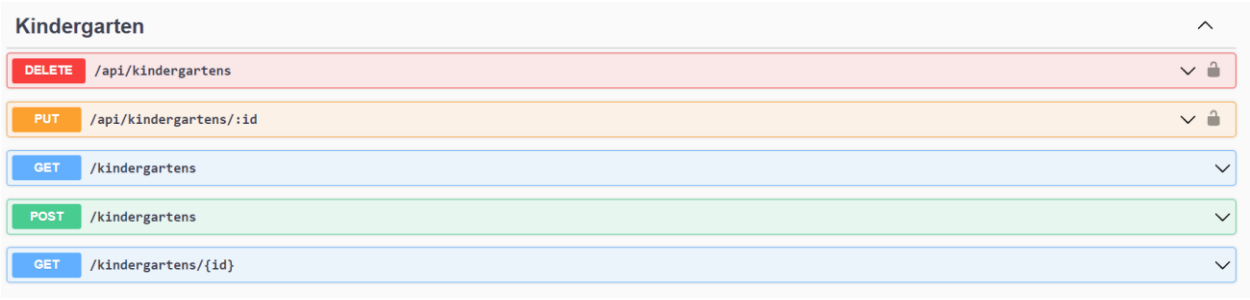

Рисунок Г.6 – Сутність «Заклади дошкільної освіти»

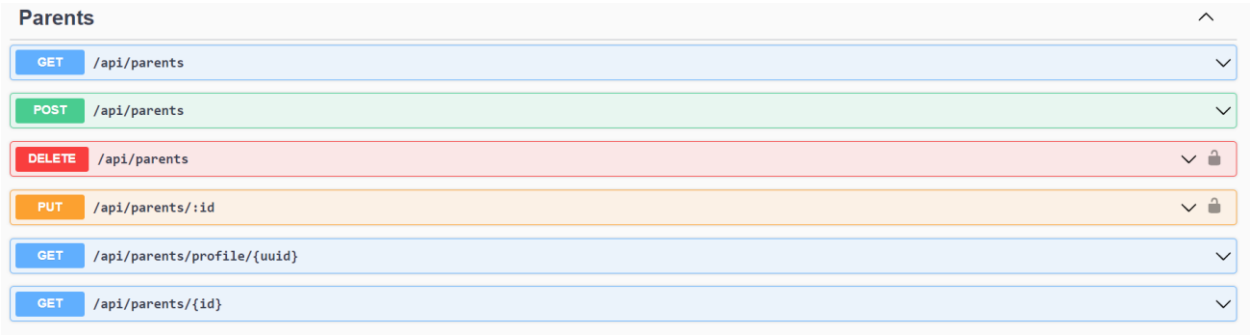

Рисунок Г.7 – Сутність «Батьки»

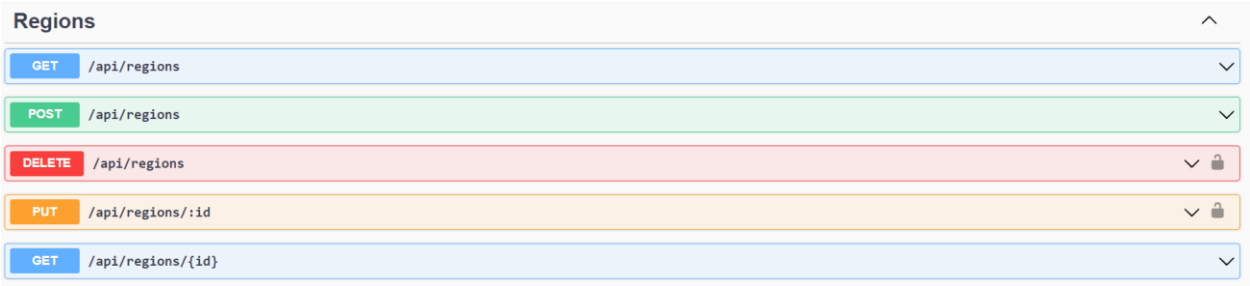

Рисунок Г.8 – Сутність «Області»

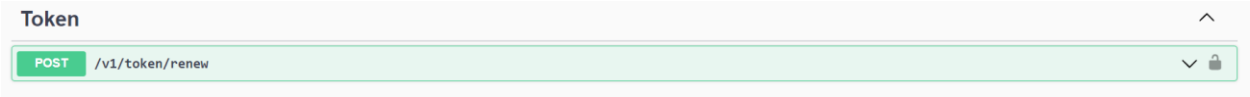

Рисунок Г.9 – Сутність «Токен»

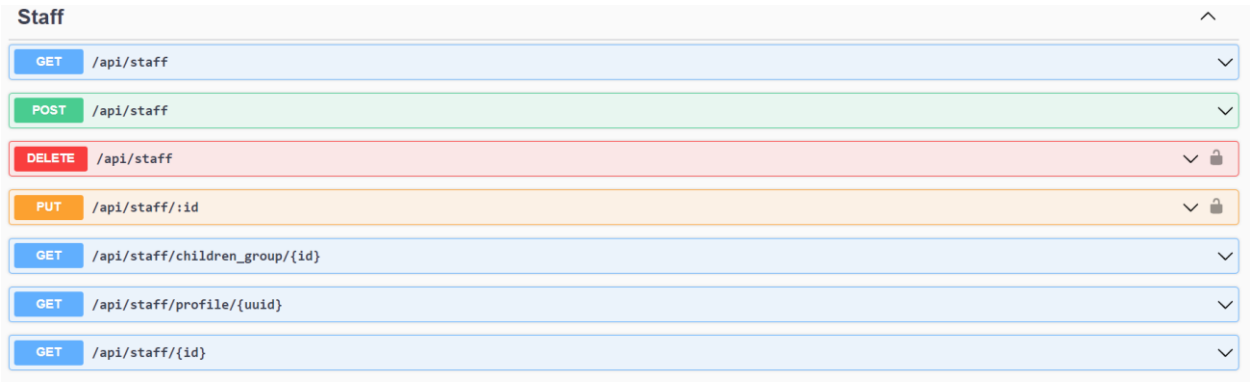

Рисунок Г.10 – Сутність «Співробітники»

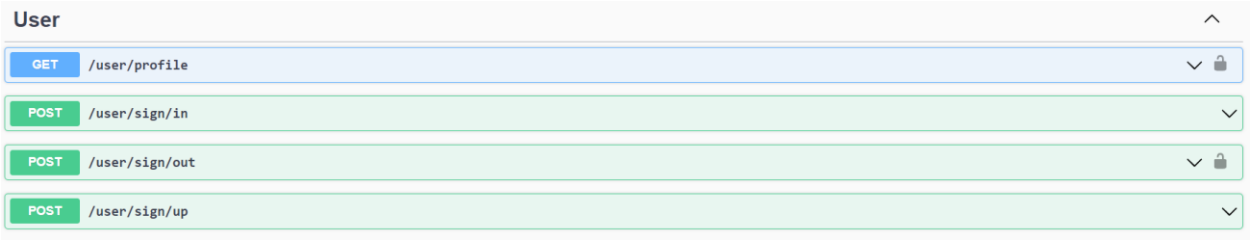

Рисунок Г.11 – Сутність «Користувачі»

Olga Roznovets

Подписано цифровой подписью: Olga Roznovets
Дата: 2022.06.15 19:13:24 +03'00'