Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

ГЮИК.502528.005 ПЗ

Системний підхід до автоматизації обліку хмарного сервісу
(тема)

Виконав:
Студент 2 курсу, групи *СПРм-19-2*
Спеціальність *122 – Комп'ютерні науки*
(код і повна назва напряму)
Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма Системне проектування
(повна назва освітньої програми)

Муренченко П.Є.
(прізвище, ініціали)

Керівник *проф.Вишняк М.Ю.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Гребеннік І. В.*
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)
Кафедра *Системотехніки*
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність *122 – Комп'ютерні науки*
(код і повна назва)
Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма Системне *проектування*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Муренченку Павлу Євгеновичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Системний підхід до автоматизації обліку хмарного сервісу »*
затверджена наказом по університету від « *23* » 03 20*21* р. № 389Ст
2. Термін подання студентом роботи до екзаменаційної комісії 24 *травня 2021 р.*
3. Вихідні дані до роботи *Застосувати системний підхід до процесу автоматизації системи обліку хмарного сервісу та розробити компоненти системи обліку хмарного сервісу. Система повинна являти собою проект клієнтського web-додатка з інтерфейсом доступу до бази даних. Перелік використовуваних програмних засобів: ОС Microsoft Windows 7,8,10, NotePad++, Python 3.7.0, Django 2.2.1, MySQL 5.7. Технічне забезпечення: IBM-сумісний ПК з МП Core 2 Duo та вище.*
4. Перелік питань, що потрібно опрацювати в роботі *Вступ. 1 Аналіз предметної області та теоретичне визначення системного підходу. 1.1 Аналіз хмарних сервісів та їх актуальності. 1.2 Поняття та сутність системного підходу. 1.3 Постановка завдання. 2 Застосування системного підходу до автоматизації обліку хмарного сервісу. 2.1 Аналіз роботи хмарного*

сервісу як підприємства 2.2 Визначення основних елементів системи. 2.3 Визначення внутрішніх елементів та взаємозв'язків автоматизованої системи. 3 Практична реалізація дослідженої системи. 3.1 Проектування системи. 3.2 Вибір мови програмування та обґрунтування вибору СУБД. 3.3 Розробка програмного забезпечення. 3.4 Розробка алгоритму роботи системи. Висновки..

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів, комп'ютерних ілюстрацій) ***5.1*** *Контекстна діаграма IDEF0 системи (1 аркуш формату А4).* ***5.2*** *IDEF0-діаграма декомпозиції (1 аркуш формату А4).* ***5.3*** *Діаграма варіантів використання. (1 аркуш формату А4).* ***5.4*** *Фізична модель бази даних системи (1 аркуш формату А4).* ***5.5*** *Діаграма класів (1 аркуш формату А4).* ***5.6*** *Схема даних БД системи (1 аркуш формату А4).* ***5.7*** *Алгоритм роботи системи (1 аркуш формату А4).*

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання, аналіз завдання, уточнення*	*23.03.2021*	
	плану роботи		
2	*Аналіз предметних області та теоритичне*	*25.03.2021*	
	визначення системного підходу		
3	*Постановка задачі та вибір методу її вирішення*	*03.04.2021*	
4	*Проведення досліджень системи обліку*	*18.04.2021*	
5	*Оформлення пояснювальної записки*	*27.04.2021*	
6	*Підготовка презентації*	*14.05.2021*	
7	*Подання закінченої роботи науковому керівникові*	*15.05.2021*	
8	*Подання роботи на рецензування*	*20.05.2021*	
9	*Попередній захист*	*20.05.2021*	
10	*Подання роботи до комісії*	*23.05.2021*	

Дата видачі завдання *23 березня 2021 р.*

Студент _____ *Муренченко П.Є.*

(підпис)

Керівник роботи _____ *проф.Вишняк М.Ю.*

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 67 аркушів, 29 рисунків, 2 таблиці, 3 додатків, 29 джерела; графічний матеріал – 9 аркушів.

АВТОМАТИЗАЦІЯ, СИСТЕМНИЙ ПІДХІД, ХМАРНИЙ СЕРВІС, ЕЛЕКТРОННА КОМЕРЦІЯ, КЛІЄНТ, БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗОЮ ДАНИХ, WEB-СЕРВІС, СЕРВЕР, ІГРОВІ ДОДАТКИ, WEB-ФОРМА, MYSQL, PYTHON, DJANGO

Об’єкт дослідження кваліфікаційної роботи є хмарні сервіси, бізнес-процеси яких потребують автоматизації.

Предмет дослідження – системний підхід, як методологія для автоматизації обліку хмарного сервісу.

Метою роботи полягає дослідженні системи обліку з метою автоматизації.

Методи дослідження – системний підхід, об’єктно-орієнтований аналіз, методи структурного моделювання реляційних баз даних і об’єктно-орієнтованого програмування на мові Python.

Результати роботи – розроблені моделі системи обліку хмарного сервісу, рекомендації до автоматизації та компоненти системи обліку хмарного сервісу, що включає базу даних, web-інтерфейс доступу до бази даних та функції надання різного рівня доступу користувачам.

Область застосування: забезпечення та автоматизація електронної комерції продажу послуг хмарного сервісу.

ABSTRACT

Explanatory note contains 67 sheets, 29 figures, 2 tables, 3 annexes, 29 sources; graphic material - 9 sheets.

AUTOMATION, SYSTEM APPROACH, CLOUD SERVICE, ELECTRONIC COMMERCE, CLIENT, DATABASE, DATABASE MANAGEMENT SYSTEM, DATA SERVICE, WEB SERVICE, PYTHON, DJANGO

The object of qualification research is cloud services, business processes of which require automation.

The subject of research is a systematic approach as a methodology for automating cloud service accounting.

The purpose of the work is to study the accounting system for automation.

Research methods - systems approach, object-oriented analysis, methods of structural modeling of relational databases and object-oriented programming in Python.

The results of the work - developed models of the cloud service accounting system, recommendations for automation and components of the user accounting system, which includes a database, web-interface for database access and functions for providing different levels of access to users.

Scope: providing and automating e-commerce sales of cloud services.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД – база даних;

СУБД – система управління базами даних;

E-commerce – Electronic Commerce, інформаційні системи електронної комерції;

CASE – Computer Aided Software Engineering, сукупність методів та засобів автоматизованого проектування інформаційних систем;

ERD – Entity Relationship Diagram, модель даних для опису концептуальні схеми в нотації «сутність» – «зв’язок»;

HTML – HyperText Markup Language, мова гипертекстової розмітки;

HTTP – HyperText Transfer Prorocol, протокол передачі гіпертекстових файлів;

CSS – Cascading Style Sheets, каскадні таблиці стилів;

MS – фірма Microsoft;

SQL – Structured Query Language, мова структурованих запитів;

URL – Uniform Resource Locator, уніфікований показник інформаційного ресурсу.

ВСТУП

Протягом 1960-х початкові концепції розподілу часу стали популярними за допомогою RJE (Remote Job Entry). До початку 1970-х рр. Рішення для спільного використання були доступні на таких платформах, як Multics (на апаратному забезпеченні GE), Cambridge CTSS та найперших портах UNIX (на апаратному забезпеченні DEC). Однак модель "центру обробки даних", де користувачі подавали завдання операторам для роботи на мейнфреймах IBM, була переважною.

У 90-х роках телекомунікаційні компанії, які раніше пропонували переважно виділені схеми передачі даних "точка-точка", почали пропонувати послуги віртуальної приватної мережі (VPN) із порівнянною якістю обслуговування, але з меншою вартістю. Змінюючи трафік, як вони вважали за потрібне, щоб збалансувати використання сервера, вони могли б ефективніше використовувати загальну пропускну здатність мережі. Вони почали використовувати хмарний символ для позначення точки розмежування між тим, за що відповідав постачальник, і тим, за що відповідали користувачі. Хмарні обчислення розширили цю межу, охоплюючи всі сервери, а також мережеву інфраструктуру. У міру того, як комп’ютери стали більш розповсюдженими, вчені та технологи досліджували способи зробити широкомасштабну обчислювальну потужність доступною для більшої кількості користувачів за допомогою розподілу часу. Вони експериментували з алгоритмами для оптимізації інфраструктури, платформи та додатків, щоб визначити пріоритети процесорів та підвищити ефективність для кінцевих користувачів.

На теперішній час ранок хмарних сервісів стрімко зростає. Багато організацій вже перенесли свої робочі навантаження в хмару. Згідно з останніми дослідженнями, міграція до публічної хмари зросла до 92% з 89% за останні три роки. І зараз понад 80% фірм, що мають 1000 або більше працівників, використовують одночасно кілька хмарних платформ. Очікується, що до 2024 року цей відсоток зросте до 90%. Більше того, протягом наступного року світові ринкові витрати на публічних платформах сягнуть приголомшливих 277 млрд. Доларів США, що на 73% більше, ніж у 2018 році.

Все більше і більше організацій та людей потребує наявність широкого списку хмарних сервісів.

Для функціонування хмарних сервісів як бізнес-проекту необхідні більш нові та ефективні системи обліку.

Компанії використовує облікові записи таких користувачів таких систем для автентифікації, відстеження, реєстрації та моніторингу своїх послуг. Дані записи можуть бути таких типів облікових записів користувачів: обліковий запис системи, суперкористувач, звичайний користувач та гостьовий.

Дані з таких систем обліку використовуються для планування, маркетингу та створення фінансової звітності.

Мета роботи полягає дослідженні системи обліку з метою автоматизації.

Об'єктом дослідження є хмарні сервіси.

Предмет дослідження – організаційна структура хмарного сервісу, яка вважається неавтоматизованою та неефективною.

Теоретична і практична значущість роботи полягає в тому, що розроблена модель організаційної структури хмарного сервісу може використовуватись до побудови та автоматизації системи обліку більшості підприємств, які надають послуги хмарних сервісів, а також доведена ефективність використання системного підходу до автоматизації може служити основою для майбутніх досліджень застосування такого підходу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНЕ ВИЗНАЧЕННЯ СИСТЕМНОГО ПІДХОДУ

1.1 Аналіз хмарних сервісів та їх актуальності

Хмарні обчислення - це модель надання зручного мережевого доступу до колективно використовуваному набору параметрів, що обчислювальних ресурсів (наприклад, мереж, серверів, сховищ даних, додатків сервісів), які користувач може оперативно задіяти під свої завдання і вивільняти при зведенні до мінімуму числа взаємодій з постачальником послуги або власних управлінських зусиль.[1]

На теперішній час технологія хмарних обчислень стає все більш популярною, так як з’явилась глобальна потреба в побудові хмарних сервісі для створення умов віддаленої роботи та надання віддаленого корпоративного доступу до ресурсів компанії. Також зросла і популярність хмарних сервісів для фізичних осіб, наприклад, сервісів самообслуговування, хмарних сховищ та ігрових додатків.

Серед постачальників хмарних обчислень найбільш популярними є такі базові типи хмарних сервісів:

– Infrastructure as a service(IaaS) – сервіс, який надає основну обчислювальну інфраструктуру: сховища даних, сервери, мережеві. IaaS забезпечує інфраструктуру, необхідну багатьом постачальникам хмарних послуг для управління інструментами SaaS, але не хочуть підтримувати себе. Він служить повною структурою ЦОД, усуваючи необхідність у ресурсомістких установках на місці.

Прикладами IaaS є веб-служби Amazon (AWS), Microsoft Azure та Google Compute Engine. Ці постачальники підтримують усі сервери зберігання даних та мережеве обладнання, а також можуть пропонувати балансування навантаження, брандмауери додатків тощо. Багато відомих постачальників SaaS працюють на платформах IaaS.[2]

– Platform as a service(PaaS) – хмарний сервіс, який надає клієнтові готову програмну платформу. За допомогою даного сервісу клієнти можуть встановлювати будь-які додатки та використовувати будь-які інструменти в інфраструктурі. У вказані моделі постачальник керує і обслуговує операційну систему, обробку даних, мережу, сервер та сховища. PaaS служить веб-середовищем, де розробники можуть створювати хмарні програми. PaaS забезпечує базу даних, операційну систему та мову програмування, які організації можуть використовувати для розробки хмарного програмного забезпечення без необхідності підтримувати основні елементи. [3]

– Software as a Service(SaaS) – модель ліцензування та надання програмного забезпечення, яка організовується централізовано на основі підписок. Найбільш загальновизнаний тип хмарних служб відомий як програмне забезпечення як послуга, або SaaS. Ця широка категорія охоплює різноманітні послуги, такі як зберігання та резервне копіювання файлів, веб-електронна пошта та інструменти управління проектами.

Прикладами постачальників хмарних послуг SaaS є Dropbox, G Suite, Microsoft Office 365, Slack та Citrix Content Collaboration. У даних додатках користувачі можуть отримувати доступ, ділитися, зберігати та захищати інформацію в "хмарі".

На теперішній час існують більш спеціалізовані сервіси хмарних обчислень.:

– Content as a service (CaaS) це модель сервісу[4], у якій постачальник послуг надає контент на вимогу користувачу сервісу через веб-служби ліцензування якої виникає після передоплати. Постачальник послуг розташовує сервіс централізовано у хмарі, презентує та надає зацікавленим користувачам контент через в будь-які додатки чи систему;

– Data as a service(DaaS) – це модель хмарного сервісу, за домопогою якої передаються та розповсюджуються файли даних будь-якого формату клієнтам через мережу[5]. Модель використовує хмарну базову технологію, яка підтримує веб-сервіси та сервісно-орієнтовану архітектуру. Інформація DaaS зберігається у хмарі та є доступною через різні пристрої. Сервіс також вивантажує хмарний провайдер даних для хмарного постачальника;

– Desktop as a service(тежDaaS) – це хмарна служба віртуалізації робочого столу, розміщена стороннім підприємством. Сторонній хмарний провайдер управляє усіма внутрішніми ресурсами, такими як пам’ять настільних комп’ютерів, обчислення та мережі, включаючи віртуальні хмарні машини, на яких працюють настільні операційні системи. Робочий стіл як постачальник послуг передає віртуальні робочі столи на пристрої кінцевих користувачів, дозволяючи в будь-який час і в будь-якому місці доступ до робочих столів та додатків. Послуги DaaS, як і більшість пропозицій хмарних послуг, базуються на підписці. Організації можуть також розгорнути інфраструктуру робочого столу в приватній хмарі в локальному центрі обробки даних;

– Function as a service(FaaS) – це безсерверний спосіб виконання модульних фрагментів коду. FaaS дозволяє розробникам писати та оновлювати фрагмент коду у режими реального часу, який потім може бути виконаний у відповідь на подію,

наприклад, коли користувач натискає на елемент у веб-додатку. Це полегшує масштабування коду та є економічно вигідним способом реалізації мікропослуг;

– Integration platform as a service(IPaaS) – це набір хмарних рішень, що дозволяють розробляти, виконувати та керувати широким спектром сценаріїв інтеграції, таких як інтеграція даних, інтеграція програм, системна інтеграція, хмарна інтеграція, архітектура, орієнтована на сервіс (SOA). інтеграція, інтеграція процесів, інтеграція Інтернету речей, управління API та сценарії інтеграції B2B. Він може підключити будь-яку комбінацію локальних систем або хмарних додатків всередині організації або декількох. Ця функціональність дозволяє iPaaS виступати як ідеальна гібридна платформа інтеграції (HIP). Хмарні платформи інтеграції настільки універсальні, тому все більше підприємств починає інвестувати в iPaaS. Багато постачальників iPaaS використовують свою платформу лише для того, щоб запропонувати інтегровані «роз'єми» для додатків SaaS, і навіть ті, що пропонують платформу корпоративної інтеграції як послугу (EiPaaS), як правило, продають лише ту технологію, яку ваші архітектори інтеграції можуть використовувати для розробки, розгортання та підтримувати свої інтеграційні рішення. Лише декілька постачальників надають повністю керовані послуги інтеграції, побудовані на хмарній платформі інтеграції[6];

– Mobile backend as a service(MBaaS) або Backend as a Service(BaaS) – модель хмарного сервісу, в якій розробники передають усі закулісні аспекти веб або мобільного додатка на аутсорсинг, щоб їм залишалося лише писати та підтримувати інтерфейс. Постачальники BaaS пропонують заздалегідь написане програмне забезпечення для дій, що відбуваються на серверах, таких як автентифікація користувачів, управління базами даних, віддалене оновлення та push-сповіщення (для мобільних додатків), а також хмарне зберігання та хостинг;

– Network as a service(NaaS) – хмарна модель, яка дозволяє користувачам легко управляти мережею та досягти результатів, яких вони очікують від неї, не володіючи, не будуючи та не підтримуючи власної інфраструктури;

NaaS може замінити апаратно-орієнтовані VPN, балансири навантаження, пристрої брандмауера та багатопротокольні комутації міток (MPLS). Користувачі можуть масштабуватися вгору та вниз у міру зміни попиту, швидко розгортати послуги та елімітувати витрати на обладнання;

– Security as a service(SECaaS або SaaS) це аутсорсинговий хмарний сервіс, в якій провайдер керує та керує безпекою клієнта. Найголовнішим, найпростішим прикладом безпеки як послуги є використання антивірусного програмного

забезпечення через Інтернет. Завдяки безпеці як послузі, рішення безпеки більше не постачаються локально, де IT-відділ клієнта встановлює програмне забезпечення для захисту від вірусів, програмне забезпечення для фільтрації спаму та інші засоби захисту на кожному комп'ютері або в мережі або на сервері на робочому місці, підтримуючи програмне забезпечення у оновленому стані. Даними задачами займається постачальник послуг;

– Data Base as a Service(DBaaS) – це послуга хмарних обчислень, яка дозволяє користувачам отримувати доступ до хмарної системи баз даних і використовувати її без придбання та налаштування власного обладнання, встановлення власного програмного забезпечення для баз даних чи управління ними. Хмарний провайдер піклується про все - від періодичного оновлення до резервного копіювання до забезпечення того, щоб система баз даних залишалася доступною та безпечною цілодобово;

– Information as a Service(теж IaaS) – це модель хмарних сервісів, яка забезпечує дані для своїх клієнтів. Дані подаються у корпоративному та зручному для користувача форматі. Доставка даних та інформації по суті працює за стандартною схемою, яка здатна ефективно представляти та розподіляти дані та інформацію;

– Integration as a Service(теж IaaS) є моделлю надання хмарних послуг для інтеграції. Integration-as-a-Service забезпечує інтеграційне рішення, яке забезпечує зв’язок із серверними системами, джерелами, файлами та операційними програмами завдяки реалізації чітко визначених інтерфейсів, веб-служб та дзвінків між додатками та джерелами даних. Це забезпечує користувачам більш вільно пов'язане середовище, захищене від складної взаємозалежності. Модель інтеграції як послуги надає можливість інтеграції в хмарі, що дозволяє обмінюватися даними між системами, а також сторонніми постачальниками в режимі реального часу;

– Management або Governance as a Service(MaaS або GaaS) дозволяє керувати будь-якою кількістю хмарних ресурсів із вказаними параметрами (virtualization, use of resources, topology);

– Process as a Service(теж PaaS) – хмарний сервіс, який дозволяє керувати зв’язком даних або послуг в одному бізнес-проекті, що розміщуються в одній або декількох хмарах;

– Storage as a Service(STaaS) – це хмарна модель, за якої компанія здає в оренду або здає в оренду свою інфраструктуру зберігання іншій компанії або приватним особам для зберігання даних. Невеликі компанії та приватні особи часто

вважають це зручною методологією для управління резервними копіями та забезпечення економії витрат на персонал, обладнання та фізичний простір. Компанію, що надає SaaS, можна назвати постачальником послуг зберігання даних. Зберігання як послуга може також називатися розміщеним сховищем. Microsoft OneDrive і Google Drive використовують послуги STaaS.

– Testing as a Service(TaaS) є хмарною моделлю аутсорсингу, при якій тестування програмного забезпечення здійснюється стороннім постачальником послуг, а не працівниками організації. У TaaS тестування проводиться постачальником послуг, який спеціалізується на моделюванні реальних середовищ тестування та пошуку помилок у програмному продукті..

– Disaster Recovery as a Service(DRaaS) – це модель хмарних обчислень, яка дозволяє організації створювати резервні копії своїх даних та IT-інфраструктури у сторонніх середовищах хмарних обчислень та забезпечувати всю організацію DR за допомогою рішення SaaS, щоб відновити доступ та функціональність до IT-інфраструктури після аварії. Модель надання послуг означає, що сама організація не повинна володіти всіма ресурсами або керувати всім управлінням для ліквідації наслідків аварій, а натомість покладатися на постачальника послуг.

– Backup as a Service(BaaS) - це підхід до резервного копіювання даних, який передбачає придбання послуг резервного копіювання та відновлення у постачальника послуг онлайн-резервного копіювання даних. Замість того, щоб виконувати резервне копіювання за допомогою централізованого локального IT-відділу, BaaS підключає системи до приватної, публічної або гібридної хмари, керованої зовнішнім провайдером. Резервним копіюванням як послугою легше керувати, ніж іншими послугами поза межами сайту. Замість того, щоб турбуватися про обертання стрічок чи жорстких дисків та керування ними за межами майданчика, адміністратори сховища даних можуть розвантажити обслуговування та управління провайдером.

BaaS може використовуватися, коли організація переросла застарілу резервну копію сховища, і їй доведеться пройти дороге оновлення або бракує ресурсів для локального резервного копіювання високого рівня. Аутсорсинг резервного копіювання та відновлення для постачальника може також забезпечити доступ до даних або їх можна відновити з віддаленого місця в разі відключення або відмови.[7]

– Monitoring as a Service(MaaS) моніторинг як сервіс – це сучасна модель доставки хмарних послуг, що передаються підрядниками, насамперед на бізнес-

платформах, які використовують Інтернет для ведення бізнесу. Моніторинг безпеки передбачає захист підприємства чи державного клієнта від кіберзагроз. Команда безпеки відіграє вирішальну роль у забезпеченні та підтримці конфіденційності, цілісності та доступності IT-активів. Однак обмеження часу та ресурсів обмежує операції з безпеки та їх ефективність для більшості компаній. Це вимагає постійної пильності щодо інфраструктури безпеки та важливих інформаційних активів;

– Hardware as a Service(HaaS) – хмарний сервіс, що передбачає надання апаратної інфраструктури в хмарі. Клієнт орендує не тільки потрібну йому інфраструктуру, але і її обслуговування з боку провайдера;

– Communications as a Service(CaaS) – це аутсорсингове рішення для корпоративних комунікацій, яке можна орендувати у одного постачальника. Такі комунікації можуть включати передачу голосу через IP (VoIP або Інтернет-телефонія), обмін миттєвими повідомленнями, додатки для співпраці та відеоконференції за допомогою стаціонарних та мобільних пристроїв;

– Container as a Service(CaaS) – це хмарний сервіс, який дозволяє розробникам програмного забезпечення та IT-відділам завантажувати, впорядковувати, запускати, масштабувати та керувати контейнерами за допомогою віртуалізації на основі контейнерів. CaaS відрізняється від PaaS, оскільки він покладається на використання контейнерів;

– Resourse as a Service(RaaS) - це економічна модель хмарних обчислень, яка дозволяє провайдерам продавати окремі ресурси (такі як центральний процесор, пам’ять та ресурси вводу-виводу) протягом декількох секунд за раз. У хмарі RaaS клієнти можуть придбати саме ті ресурси, які їм потрібні, коли вони їм потрібні;

– Customer Relationship Management as a Service(CRMaaS) – це хмарне CRM забезпечення, де всі програми та дані зберігаються або розміщуються на власних серверах постачальників CRM у центрі обробки даних та отримують доступ через Інтернет-браузер. Хмарний CRM, також відомий як CRM на основі хмар, CRM для програмного забезпечення як сервіс (SaaS), Інтернет-CRM та Web CRM;

– Bookkeeping as a Service(BaaS) – хмарний сервіс, який надає послуги віртуального сервісу бухгалтерії;

– Game as a Service(GaaS) – хмарний сервіс, який надає користувачу доступ до ігрових додатків. GaaS - це засіб монетизації відеоігор або після їх первинного продажу, або для підтримки безкоштовної моделі. Ігри, випущені за моделлю GaaS, зазвичай отримують довгий або невизначений потік монетизованого нового вмісту з часом, щоб заохотити гравців продовжувати платити за підтримку гри. Ігри GaaS

часто можна транслювати прямо з хмари на пристрій користувача, що дозволяє отримати до них доступ з будь-якого місця та в будь-який час. Це також дозволяє покращити крос-платформну функціональність, що є важливою частиною збереження конкурентоспроможності в ігровій галузі. Ця стратегія дозволяє оновлювати ігри щотижня, щомісяця, а іноді навіть щодня, щоб залучити користувачів.[8]

Велика кількість провайдерів розширюють з кожним роком кількість хмарних послуг в Україні. Такі послуги надають ряд українських компаній Воля, DeNovo, Tucha, Vega, які забезпечують підтримку ряду хмарних сервісів: IaaS; STaaS; BaaS; RaaS; CRMaaS; CaaS; BaaS.

Класифікація за типом середовища хмарного сервісу:

Публічні хмарні сервіси це сервіси де послуги, які постачальник надає численним клієнтам через Інтернет, називаються загальнодоступними хмарними послугами. Зазначені вище приклади SaaS, IaaS та PaaS надають загальнодоступні хмарні послуги. Найбільшою перевагою використання публічних хмарних служб є можливість широко розподіляти ресурси, що дозволяє організаціям пропонувати працівникам більше можливостей, ніж це могло б бути можливо окремо.

Приватні хмарні сервіси це сервіси де послуги, які постачальник не робить загальнодоступними для корпоративних користувачів або абонентів, називаються приватними хмарними послугами. За допомогою приватної моделі хмарних служб програми та дані стають доступними через власну внутрішню інфраструктуру організації. Платформа та програмне забезпечення обслуговують одну компанію і не доступні для зовнішніх користувачів. Компанії, які працюють із високочутливими даними, такими як галузі охорони здоров'я та банківської діяльності, часто використовують приватні хмари, щоб використовувати вдосконалені протоколи безпеки та розширювати ресурси у віртуалізованому середовищі за потреби.

У гібридному хмарному середовищі приватне хмарне рішення поєднується із загальнодоступними хмарними послугами. Ця схема часто використовується, коли організації потрібно зберігати конфіденційні дані у приватній хмарі, але хоче, щоб працівники мали доступ до програм та ресурсів у загальнодоступній хмарі для повсякденного спілкування та співпраці. Запатентоване програмне забезпечення використовується для забезпечення зв’язку між хмарними службами, часто за допомогою однієї консолі управління IT.[9]

Виходячи з короткого опису хмарних сервісів можна зробити висновок, що тема хмарних обчислень є достатньо актуальною на теперішній час, так як з кожним

роком зростає запит та варіативність хмарних сервісів. Хмарні обчислення тісно пов’язані з бізнесом та сучасними технологіями, які охоплюють більшість сфер нашого життя.

В рамках роботи буде розглянуто хмарний сервіс GaaS як сервіс, який стрімко розвивається та потребує рішень для прискорення ефективності його роботи з організаційної точки зору, даний тип сервісу більш орієнтований на масовий сегмент ринку(на фізичних осіб) але разом з тим тісно пов'язаний з бізнесом. Неефективність такого типу постачальників послуг призводить до втрат не тільки власної організації, але й призводить до втрат інших організацій, таких видавництва та студії розробників ігор.

Не існує загального консенсусу щодо визначення бізнес-моделі. Тим не менше, навіть якщо елементи варіюються від одного визначення до іншого, більшість представників, що використовуються в літературі, включають чотири взаємопов'язані компоненти: опис пропозиції вартості бізнесу (послуга), технологія (платформа обслуговування), ланцюжок створення вартості та джерело доходів (модель доходу). Ці компоненти відповідають моделі STOF , слідуючи визначенню бізнес-моделі як "плану надання послуги, що описує визначення послуги та заплановану вартість для цільової групи, джерела доходу та забезпечуючи архітектуру щодо надання послуг, включаючи опис необхідних ресурсів, а також організаційні та фінансові домовленості між залученими суб'єктами підприємницької діяльності, включаючи опис їх ролі та розподіл витрат та доходів на суб'єктів бізнесу ". Чотири виміри бізнес-моделі будуть визначені:

- сервіс: опис передбачуваної вартості, поставленої вартості, очікуваної вартості, сприйнятої вартості

- технологія: опис технічної архітектури, сервісних платформ, пристроїв, додатків

- організація: опис суб'єктів , ролі, взаємодії, стратегії та цілі, ціннісні заходи.

- фінанси: опис джерел інвестицій, джерел витрат, джерел доходів, джерел ризику та ціноутворення

У міру зближення технологій бізнес-моделі стають складнішими для визначення. Ця складність полягає у змінах учасників ланцюга створення вартості та їх взаємодії. Наприклад, цифровий розподіл позбавляє потреби продавця, замінюючи його постачальником послуг та хостингом платформи. Як і для будь-якої іншої системи, пов’язаної з комп’ютером, досягнення технологій, зроблені бізнес-моделями, розвиваються у міру дозрівання технології. Будучи технологією, що

зростає, мало проведено емпіричних досліджень щодо бізнес-моделей SaaS. Поява онлайн-і мобільних ігор змінює структуру ринку відеоігор, дозволяючи нові способи ведення бізнесу.

1.2 Поняття та сутність системного підходу

Системний підхід заснований на узагальненні того, що все взаємопов'язане та взаємозалежне. Система складається з пов'язаного та залежного елемента, який, взаємодіючи, утворює єдине ціле. Система - це просто сукупність або поєднання речей або частин, що утворюють складне ціле.

Однією з найважливіших його характеристик є те, що вона складається з ієрархії підсистем. Це частини, що утворюють основну систему тощо. Наприклад, світ можна вважати системою, в якій різні національні економіки є підсистемами.

У свою чергу, кожна національна економіка складається з різних галузей, кожна галузь складається з фірм, і, звичайно, фірму можна вважати системою, що складається з підсистем суді, таких як виробництво, маркетинг, фінанси, бухгалтерський облік тощо. [10]

Особливості системного підходу:

– Система складається з взаємодіючих елементів. Це сукупність взаємопов'язаних та взаємозалежних частин, розташованих таким чином, що утворює єдине ціле.

– Різні підсистеми слід вивчати в їх взаємозв'язках, а не ізольовано одна від одної.

– Організаційна система має межі, що визначають, які частини є внутрішніми, а які зовнішніми.

– Система не існує у вакуумі. Він отримує інформацію, матеріал та енергію від інших систем як вхідні дані. Ці входи проходять процес трансформації в системі і залишають систему як вихід для інших систем.

– Організація - це динамічна система, оскільки вона реагує на своє оточення. Він вразливий до змін у своєму середовищі.

У системному підході увага приділяється загальній ефективності системи, а не ефективності підсистем. Враховується взаємозалежність підсистем. Ідея систем може бути застосована на організаційному рівні. При застосуванні концепцій системи беруться до уваги організації, а не лише цілі та результати діяльності різних підрозділів (підсистем).

Системний підхід розглядається як загальними, так і спеціалізованими системами. Загальний системний підхід до управління в основному стосується формальних організацій, а концепції стосуються техніки соціології, психології та філософії. Конкретна система управління включає аналіз організаційної структури, інформації, механізму планування та контролю, проектування робочих місць тощо.

Як вже обговорювалося раніше, системний підхід має величезні можливості: «Точка зору на систему може дати поштовх до уніфікації теорії управління. За визначеннями, він може розглядати різні підходи, такі як процес кількісного та поведінкового, як підсистеми в загальній теорії управління. Таким чином, системний підхід може досягти успіху там, де процесний підхід не зміг вивести управління з теорії джунглів ".[11]

Теорія систем корисна для управління, оскільки вона спрямована на досягнення цілей і розглядає організацію як відкриту систему. Честер Барнард був першою людиною, яка застосувала системний підхід у сфері управління.

Він вважає, що виконавча влада повинна керуватися, дотримуючись балансу між конфліктуючими силами та подіями. Високий порядок відповідального керівництва робить керівників ефективними. Х. Саймон розглядав організацію як складну систему процесу прийняття рішень.

Оцінка системного підходу:

Системний підхід допомагає у вивченні функцій складних організацій і був використаний як основа для нових видів організацій, таких як організація управління проектами. Можна виявити взаємозв'язки в різних функціях, таких як планування, організація, керівництво та контроль. Цей підхід має перевагу над іншими підходами, оскільки він дуже близький до реальності.

Системний підхід до створення інформаційної системи – це комплексне вивчення об'єкта автоматизації як одного цілого з представленням частин його як цілеспрямованих систем і вивчення цих систем та взаємозв'язків між ними. При системному підході об'єкт розглядається як сукупність взаємопов'язаних елементів однієї складної динамічної системи, яка перебуває в стані постійних змін під впливом багатьох внутрішніх і зовнішніх факторів, пов'язаних процесами перетворення вхідної інформації у вихідну внаслідок вирішення задач автоматизації.

1.3 Постановка завдання

До кваліфікаційної роботи ставляться наступні задачі:

– проведення дослідження й аналіз хмарного сервісу як підприємства;

– дослідити стан автоматизації системи обліку хмарного сервісу;

– визначити внутрішнє та зовнішні взаємозв’язки;

– розробити автоматизовану систему обліку хмарного сервісу на основі системного підходу.

–

–

–

2 ЗАСТОСУВАННЯ СИСТЕМНОГО ПІДХОДУ ДО АВТОМАТИЗАЦІЇ ОБЛІКУ ХМАРНОГО СЕРВІСУ

2.1 Аналіз роботи хмарного сервісу як підприємства

Розгляне в якості представника хмарного сервісу компанію «Magic Game Company».

Дана компанія надає послуги оренди обладнання для роботи ігрових додатків у певних приміщеннях та за допомогою хмарного сервісу. Вказані послуги це основний бізнес цієї компанії. цільовою аудиторією якої є люди різного статі та віку, у яких не має потужного обладнання для запуску сучасних ігрових додатків.

Клієнти комп’ютерного клубу оплачуть послуги у менеджера залу, який в надає номер комп’ютера. Менеджер залу зі свого комп’ютер розблоковує орендований комп’ютер на вказаний час. Клієнт займає місце за вказаним комп’ютером та виконує необхідні йому дії.

Після робочого дня персонал технічного відділу проводять технічну профілактику обладнання. Складаються звіти про стан обладнання та надаються менеджеру відділу у папері. Дані замовлень передаються до бухгалтерського відділу у вигляді паперових документів.

Для реалізації хмарного сервісу компанія створила певне приміщення, встановила обладнання, тобто комп’ютери різних рівнів потужності, та купила програмне забезпечення, що дозволяє передавати зображення та керування користувачу.

Клієнти цього сервісу можуть оплатити послугу в одному з комп’ютерних клубів компанії. Коли клієнт оплачує послугу - він отримує програмне забезпечення та необхідні налаштування.

Модератор обслуговує клієнтів та відповідає на дзвінки клієнтів, у яких склались технічні проблеми з роботою сервісу. Він заповнює відповідні заявки та надає в технічний відділ.

Технічний відділ обслуговує заявки про технічні проблеми. Кожний тиждень проводиться технічна профілактика обладнання та складаються звіти про стан обладнання.

Організаційна структура компанії представлена на рис. 1.1.

У компанії є наступні керуючі особи:

– власник здійснює загальне керівництво підприємством, стверджує всі документи підприємства, укладає договори з партнерами, займається розміщенням вільних коштів та несе юридичну відповідальність за діяльність компанії, виконує планування роботи компанії.

– модератор здійснює обробку замовлень та відповідає за актуальне наповнення інформаційних систем сервісу.

– головний бухгалтер керує відділом бухгалтерського обліку та очолює всі фінансові операції: нарахування зарплати, розрахунок і перерахування податків, обробляє і архівує всю фінансову документацію, складає бюджет підприємства, формує графік платежів, готує зовнішню фінансову звітність.

– менеджер відділу кадрів веде облік кадрів, є основним учасником прийому на роботу і звільнення.

– менеджер відділу маркетингу керує відділом маркетингу та бере участь у стратегічному плануванні компанії. Також проводить заходи щодо просування послуг, пошук можливих партнерів та користувачів. Надає наради щодо розширення кількості послуг сервісу.

– системний адміністратор для кожного відділу очолює команду технічного обслуговування та проводить заходи щодо підтримання стабільності систем та поліпшення якості послуг.

Зовнішні особи та партнери:

– постачальник обладнання забезпечує підприємство необхідними комплектуючими та обладнанням.

– видавництва та студії розробки комп’ютерних ігор розміщує ігрових додатків у сервісі.

– банки-партнери – банки, які забезпечують проведення фінансових операцій для внутрішніх цілей компанії і отримання оплати від клієнтів.

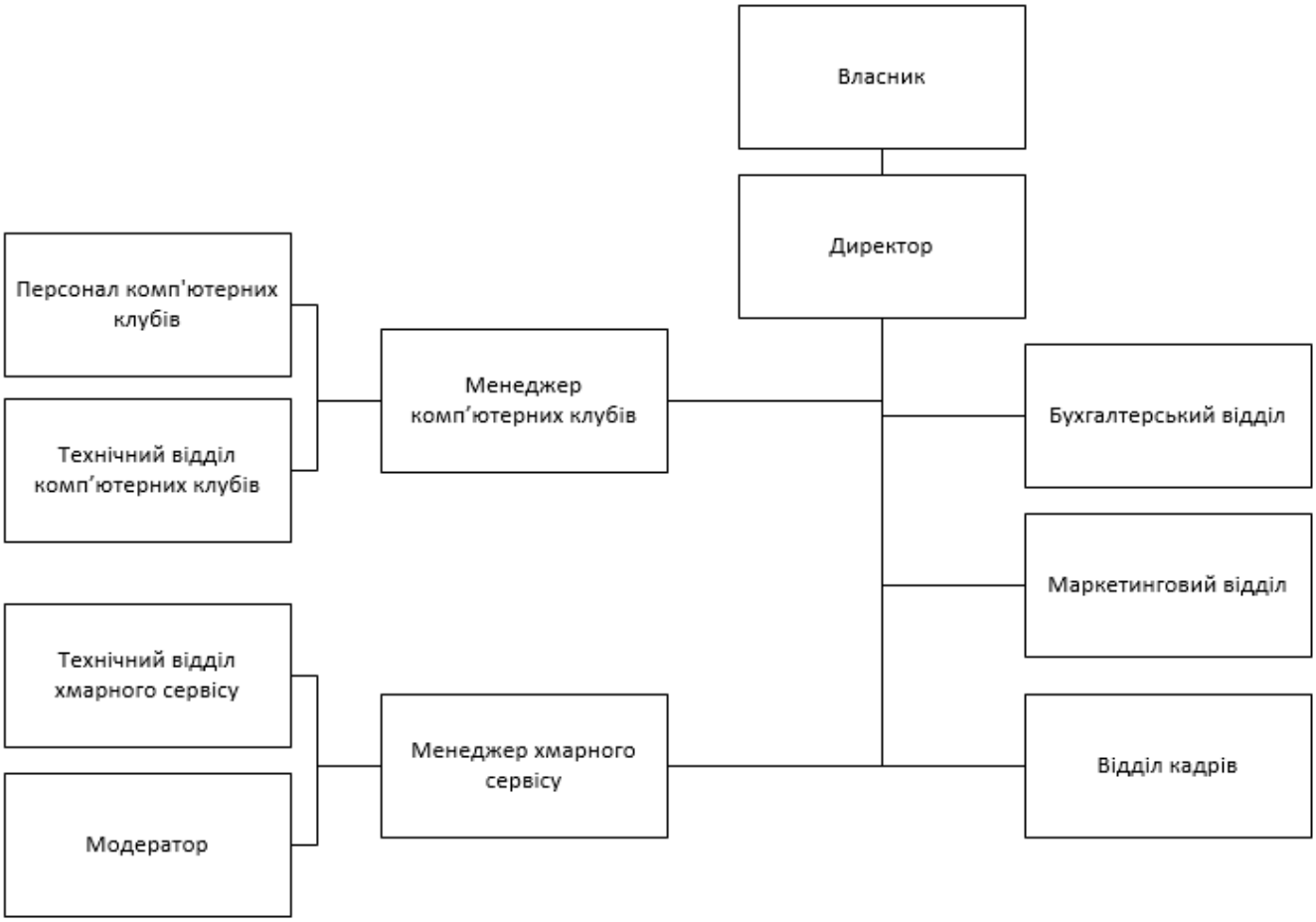

Рисунок 2.1 – Організаційна структура компанії

2.2 Визначення основних елементів системи

Система являє собою сукупність взаємопов'язаних елементів, які об'єднані єдиною метою та функціональна цілісність, та при цьому властивість самої системи не зводиться до суми властивостей елементів. [12]

Виходячи з опису підприємства хмарного сервісу, метою автоматизації є зв'язуюча ланка між клієнтами, персоналом компанії та її продуктом, і це облік користувачів хмарного сервісу. Тому основним об'єктом дослідження та розробки є автоматизована система обліку хмарного сервісу.

Мета дослідження полягає у виявленні властивостей та зв'язків основних бізнес-процесів, сформуванні функції та побудові автоматизованої системи обліку хмарного сервісу, з надання доступу до ігрових додатків.

Для цього потрібно визначити що являє собою дана система.

Облік – це функція реєстрації, збір, передача та обробка поточних повідомлень у підрозділах об'єкта управління з виконання плану.

Автоматизована інформаційна система являє собою інформаційно-технічну систему, яка включає в себе засоби автоматизації одного або декількох видів діяльності робітників певного бізнес-процесу.

Основним бізнес процесом в рамках хмарного сервісу є надання послуг хмарного сервісу у вигляді надання доступу до обладнання та ігрових додатків, за орендну плату.

В даному бізнес-процесі взаємодіють наступні актори:

– клієнт – особа яка користується послугами.

– потенційний клієнт – особа, яка ознайомлюється з продуктами хмарного сервісу але ще не зробила замовлень;

– контент-модератор – співробітник, який підтримує інформаційну наповненість;

– зовнішня розрахункова система;

– співробітники фінансового відділу;

– система надання доступу до обладнання;

– співробітники маркетингового відділу.

Тому у даній системі вказані актори повинні мати можливість користуватись автоматизованими функціями.

Зовнішнє середовище даної системи представлено у вигляді інфраструктури GaaS, яку зображено на рисунку 2.2.

Зовнішнє середовище представляється як набір конкретних меж об'єктів, що впливають на діяльність системи. Середовищє є сукупністю об'єктів, модифікація яких впливає на систему, а також об'єктів, властивості яких змінюються впродовж роботи системи.

Опис системи є описом її структурних і функціональних властивостей. Формалізований опис потребує розробки чіткої структури показників системи.

Із зовнішнього середовища видно що система обліку є зв’язуючою ланкою між технічним обладнанням та платформами клієнта, так як виконує основну функцію авторизації та надання доступу до обладнання.

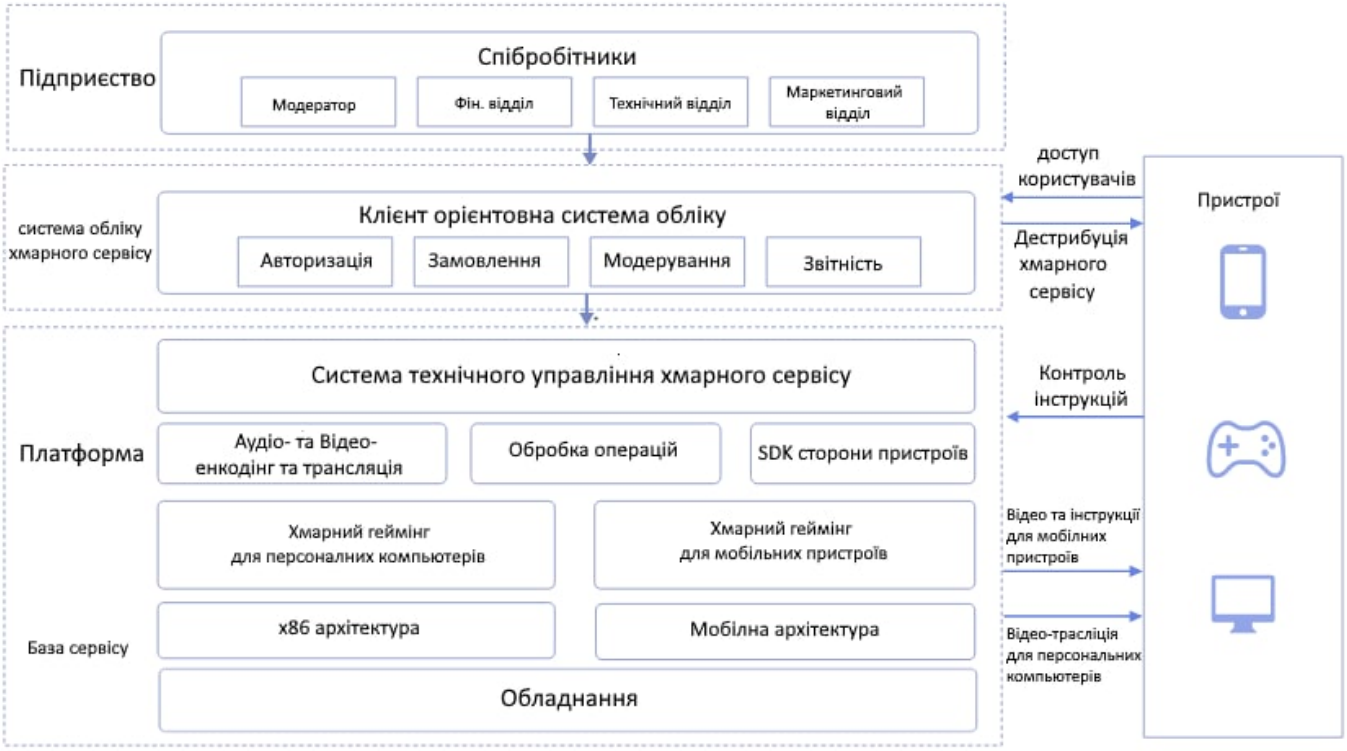

Рисунок 2.2 – Зовнішнє середовище системи обліку хмарного сервісу

Типова взаємодія між хмарним провайдером і клієнтом працює наступним чином: клієнт підключається до «хмарного ринку» через веб-інтерфейс і вибирає тип і обсяг необхідних їй ресурсів (наприклад, деякі віртуальні сервери з заданою кількістю ядер центрального процесора, пам'яттю та диском простір). Ресурси розподіляються з великого пулу, який фізично розміщений у якомусь bigdatacenter, керованому хмарним постачальником. Після створення екземпляра користувач отримує доступ до ресурсів через мережу. Додаткові ресурси можна придбати пізніше, наприклад, щоб впоратися зі збільшенням навантаження, і звільнити, коли вони більше не потрібні. Клієнт сплачує ціну, яка залежить від типу та кількості запитуваних ресурсів (наприклад, швидкості ядер процесора, обсягу пам'яті, дискового простору) та тривалості їх використання.). У хмарі Software as a Service (SaaS) система надає послуги додатків, що працюють у хмарі. "Програми Google" - це приклад широко використовуваної хмари SaaS. На відміну від цього, можливості, що надаються хмарою aPlatform as Service (PaaS), складаються з мов програмування, інструментів та середовища хостингу для додатків, розроблених замовником.

Основним фактором досягнення масштабованості інфраструктури GaaS є можливість розподілу робочого навантаження на хмарні ресурси. Це відносно легко, якщо робоче навантаження складається із виконання незалежних екземплярів гри,

які можна виконати на будь-якому доступному ресурсі, незалежно від того, де запущені інші екземпляри. Це той випадок, коли гра не дозволяє іншим гравцям взаємодіяти. Речі стають складними, якщо екземпляри не є незалежними, як у випадку системи MMOG, де всі гравці взаємодіють з одним і тим же віртуальним світом. У цьому випадку ігровий движок повинен підтримувати великий загальний стан, що дозволяє гравцям "бачити" ефекти реакцій, що виконуються іншими гравцями, що працюють у тому ж віртуальному місці. Це досягається розподілом віртуального світу між кількома зонами, кожна з яких обробляється окремим набором хмарних ресурсів. Враховуючи, що зв'язок між екземплярами ресурсів може спричинити значні затримки, важливо, щоб взаємодія між сусідніми зонами була зведена до мінімуму. Наприклад, кожен розділ може містити колекцію "островів", щоб усі взаємодії відбувалися в колекції, тоді як гравці можуть стрибати з одного "острова" "В інший. Залежно від (віртуальної) моделі мобільності кожного гравця, деякі ділянки ігрового поля можуть бути переповненими, а інші можуть стати менш заселеними. Для того, щоб впоратися з цією мінливістю, кожен контролер зони фізично розміщується на ресурсах, що надаються, і експлуатується за допомогою хмарної інфраструктури. Взагалі хмарний провайдер - це окрема організація, яка надає обчислювальні ресурси та ресурси зберігання даних ігровому оператору за моделлю оплати праці. Це означає, що оператор гри може будь-коли запитувати додаткові сервери та/або додатковий простір для зберігання та випускати їх, коли це більше не потрібно. Таким чином, ігровий оператор може вимагати більше ресурсів, коли навантаження на зону збільшується, щоб утримати сприйнятий гравцями час відгуку нижче заданого максимального порогу. Коли робоче навантаження зменшується, оператор гри може звільнити надлишки ресурсів, щоб зменшити витрати.[13]

Виходячи з даного опису можна перейти до визначення внутрішніх підсистем та функцій системи обліку хмарного сервісу.

2.3 Визначення внутрішніх елементів та взаємозв’язків автоматизованої системи

В даний час автоматизація діяльності компанії зазвичай виконується на основі регламентації і моделювання бізнес-процесів компанії. З цією метою розроблено ряд описових мов, нотацій, і програмних засобів на їх основі, покликаних

автоматизувати процес побудови даних моделей. Найбільш популярними і зручними є: UML (Unified Model Language) та BPM (Business Process Modeling) нотації.[14]

Для визначення внутрішніх елементів використовується IDEF0, що відображає функції системи. Стандарт IDEF0 визначає методологію функціонального моделювання.[15]

На головну діаграму розміщується блок, який відображає головну функцію системи (Рисунок 2.3). Головна функція системи – Система обліку хмарного сервісу ігрових додатків.

На вході даної діаграми зображуються дані, що потребує система для виконання бізнес-функції. На виході зображується дані, що надаються зовнішнім користувачам системи. Також на діаграмі зображені інструкції, які необхідні для правильного управління функціями, та механізми, що являють собою ресурси, що забезпечують виконання основної бізнес-функції.

Згідно цієї діаграми, входами у систему є: запити на надання інформації та дані реєстрації та авторизації. Виходами, насамперед, є: інформація про наявність послуг та фінансова інформація для подальшої оплати у системі банку-партнеру.

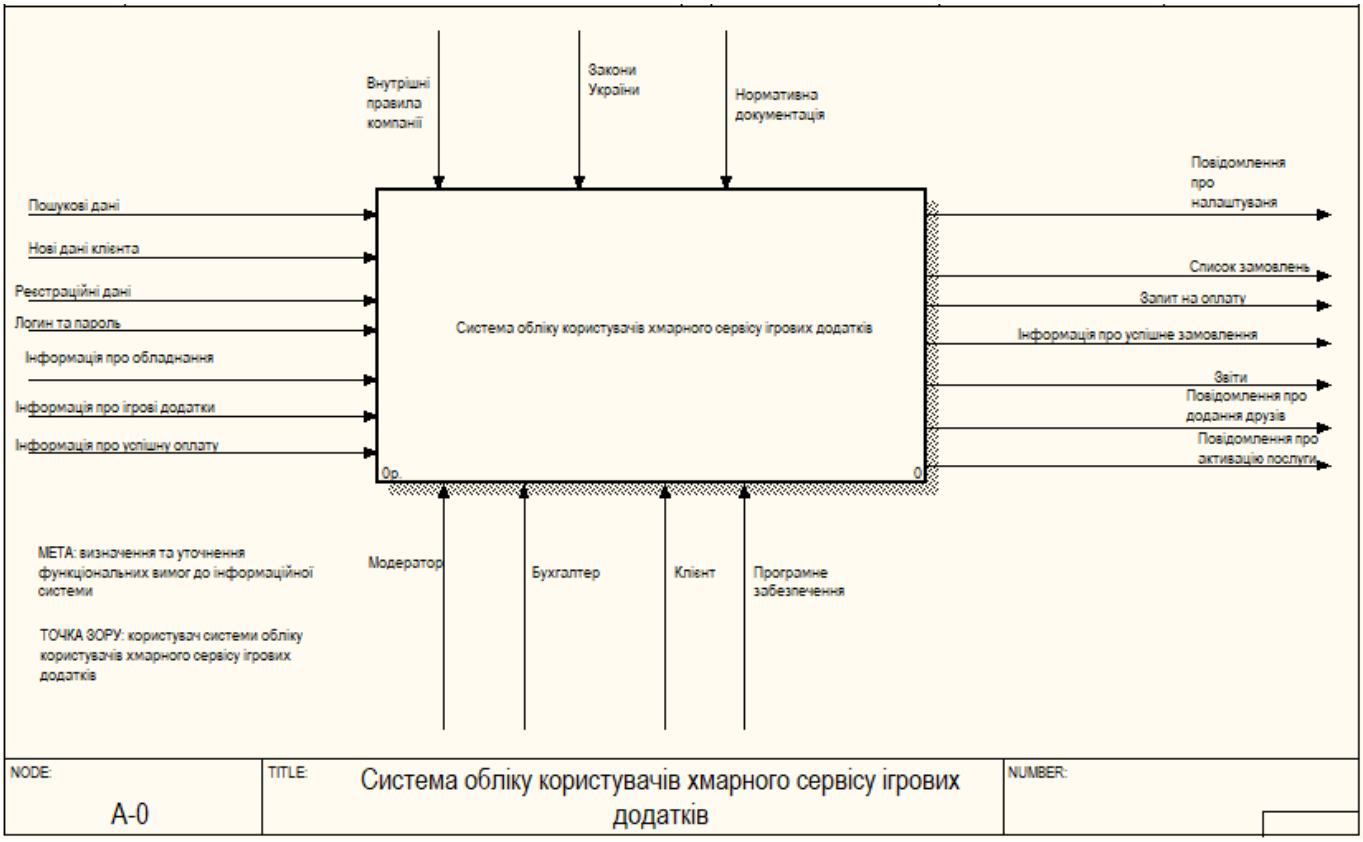

Рисунок 2.3 –Діаграма системи обліку хмарного сервісу

Далі використовуємо метод декомпозиції(Рисунок 2.4). Тобто система розкладається на підсистеми та вказується зв’язки між цими підсистемами

Система обліку хмарного сервісу ігрових додатків розкладається на наступні функції:

– реєстрація та авторизація користувача;

– керування обліковим записом;

– модерація контенту;

– замовлення послуги;

– обслуговування замовлень.

На даній діаграмі показано внутрішні зв’язки у системі між підсистемами. Так підсистема реєстрації та авторизації, приймає дані користувача, такі як пароль та логін, та передають дані облікового запису вже авторизованого користувача усім іншим підсистемам.

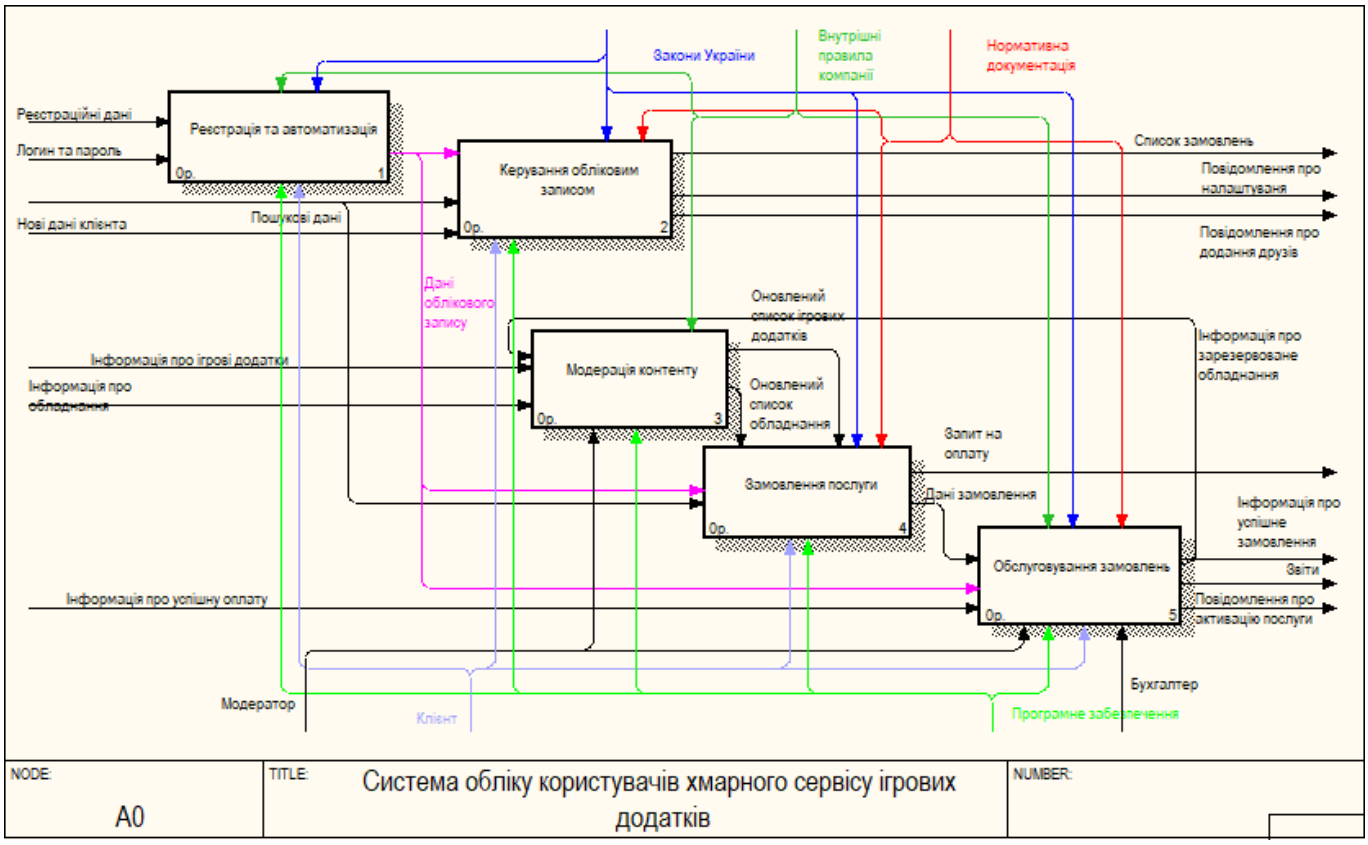

Рисунок 2.4 – Діаграма декомпозиції основної діаграми

Далі виконується декомпозиія другого рівня, у якому рокладається кожна підсистема. Такий рівень декомпозиції вважається достанім для подібних систем, тому далі розглядається лише діаграми даного рівня.

На рис.2.5 представлена декомпозиція підсистеми «Реєстрація та авторизація». Ця підсистема виконує функції реєстрації нових користувачів системи та їхньої авторизації. У функції авторизації перевіряється коректність вводу облікових даних користувача та надається певний рівень доступу в рамках хмарного сервісу.

Між даними функціями відсутній зв’язок та виконуються ці функції окремо.

Вхідними даними функції авторизації є логін та пароль користувача. Функція перевіряє наявність користувача у системі за вказаним логіном, перевіряє пароль та логін, надає роль. Дані авторизованого користувача передаються усім наступним функціям.

Функція реєстрація на вході отримує введені користувачем реєстраційні дані, виконує перевірку на коректність та фіксує у базу даних.

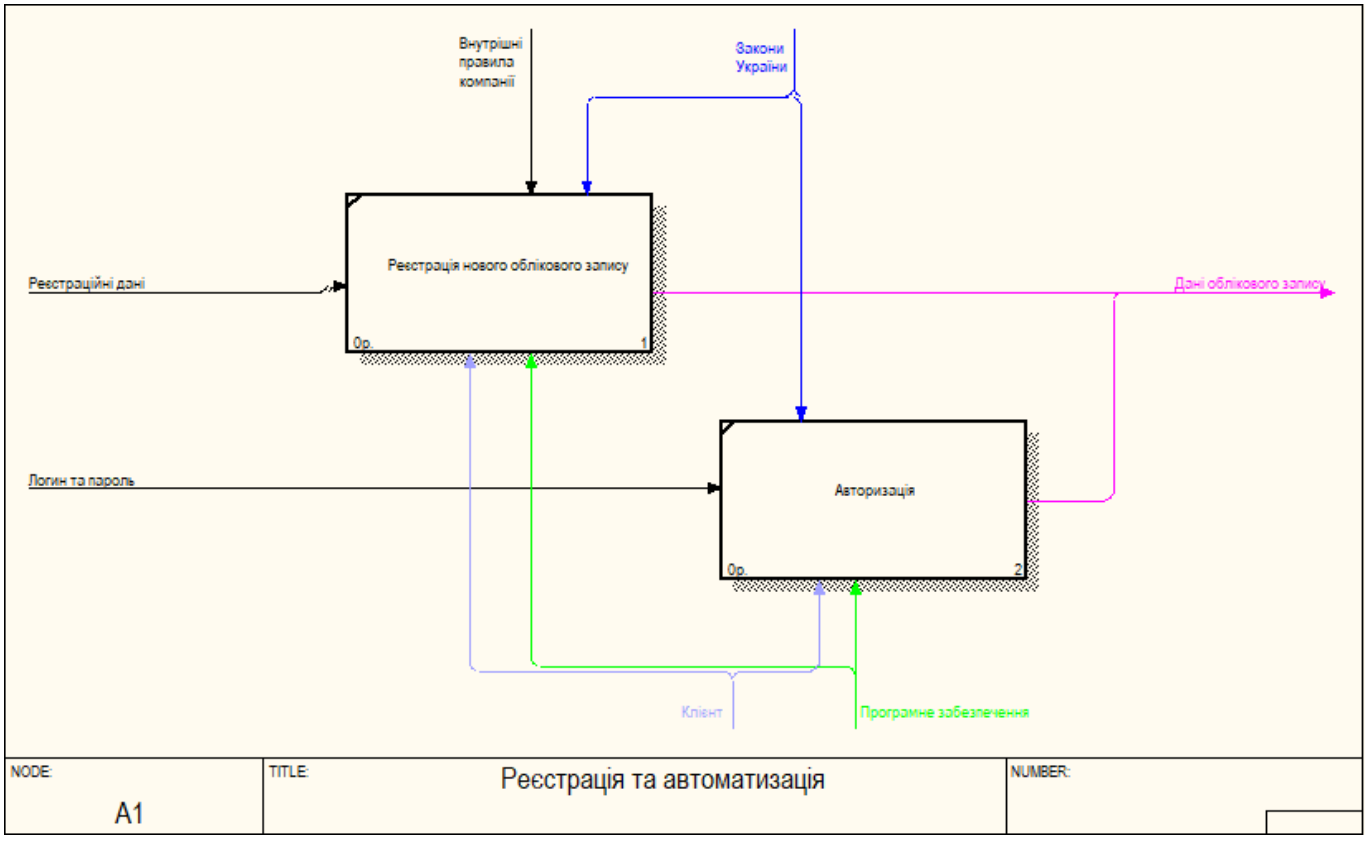

Рисунок 2.5 – Діаграма декомпозиції підсистеми «Реєстрація та авторизація»

На рис. 2.6 представлена декомпозиція підсистеми «Керування обліковим записом» системи. Призначенням даної функції є надання можливості користувачам налаштовувати профіль. Перелік можливостей: вказувати додаткові дані, призупиняти/активувати замовлені послуги, зміна облікових даних, додавання інших користувачів до свого списку.

Функція зміна даних профіля на вході приймає дані облікового запису користувача та нові дані, які користувач введе у відповідній форми. Функція виконує перевірку ввведених даних на коректність та внесення вказаних даних до бази даних.

Керування послугами виконується відповідною функцією надає можливість користувача зманювати статус послуги з активного на призупений та навпаки, а також надає можливість їх подовження. При коректної взаємодії з вказаною функцією, користувач отримує повідомлення про успішність.

Функція перегляд замовлень надає можливості переглядання виконані замовлення та отримання інформацію про активне обладнання з метою надання змоги замовлення. Функція отримує дані користувача та виконує пошук його замовлень. На виході надається список замовлень з інформацією про орендовані сервери.

Пошук друзів виконується за запитом користувача здійснює пошук серед всіх клієнтів системи. Дії клієнта у відповідній формі призводить до замін у списку друзів. На виході функція виконується тригер повідомлення про зміну списку та додавання друзів.

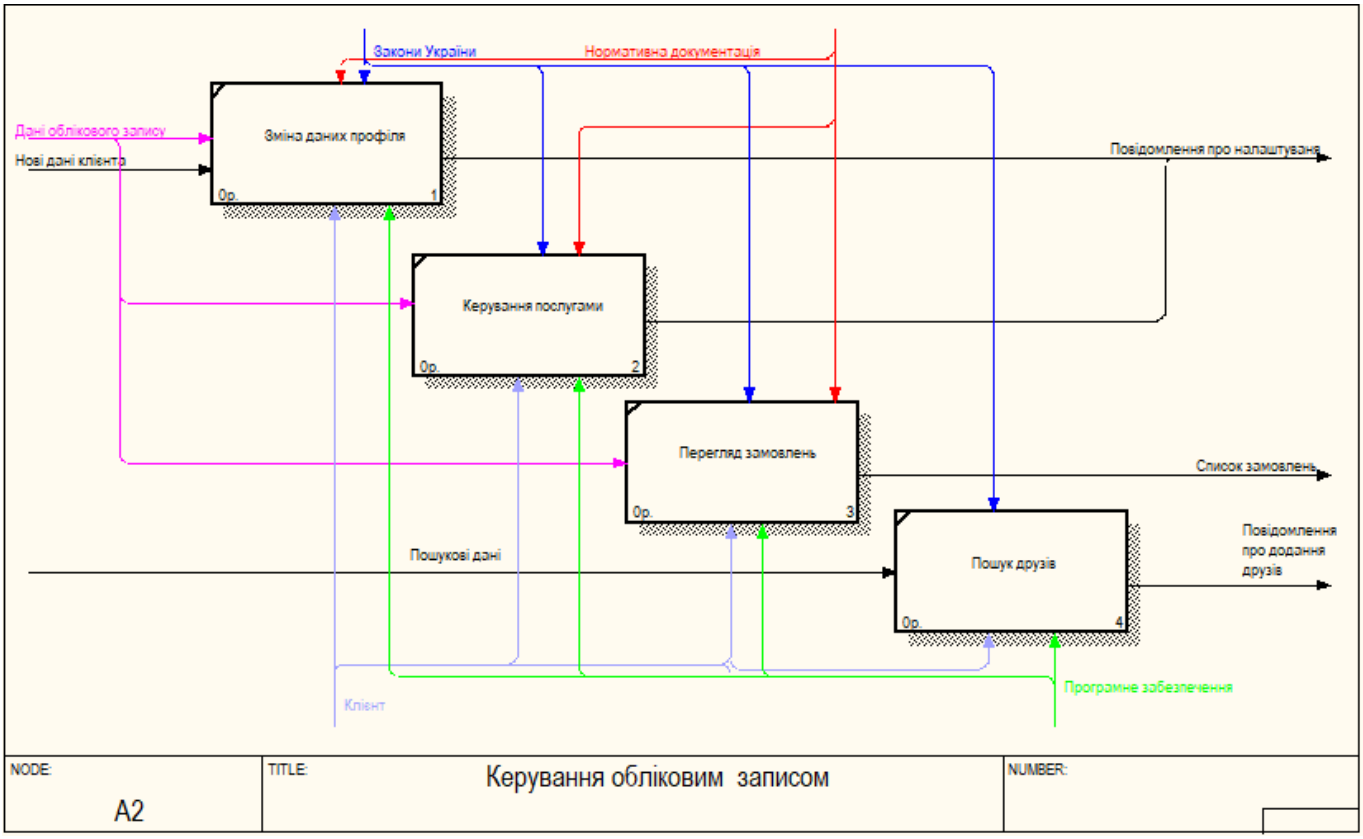

Рисунок 2.6 – Діаграма декомпозиції підсистеми «Керування обліковим записом»

На рис. 2.7 представлена декомпозиція підсистеми «Модерація контенту». Підсистем відповідає за проведення модерації та актуалізації інформації у системі. Виконується актуалізація наступних даних: дані сервера, дані ігрових додатків та дані обладнання.

Функція зміни інформації сервера виконує редагування вже існуючої інформації про сервери, ігрові додатки та послуги. Функція надає змогу модератору змінювати статус обладнання серверів та змінювати інформацію про розташування ігрових додатків. Після редагування повертає оновлений список додатків або серверів.

Функція додавання нової інформації дозволяє створювати нові сервери, ігрові додатки та послуги. Перевіряє коректність полів та зберігає у базу даних. Після успішного збереження даних повертає оновлений список додатків або серверів.

Функція видалення застарілої інформації дає можливість видалити вказаний запис про сервер, ігровий додаток або послугу. Перед видаленням запитує підтвердження дій. Після успішного видалення даних повертає оновлений список додатків або серверів.

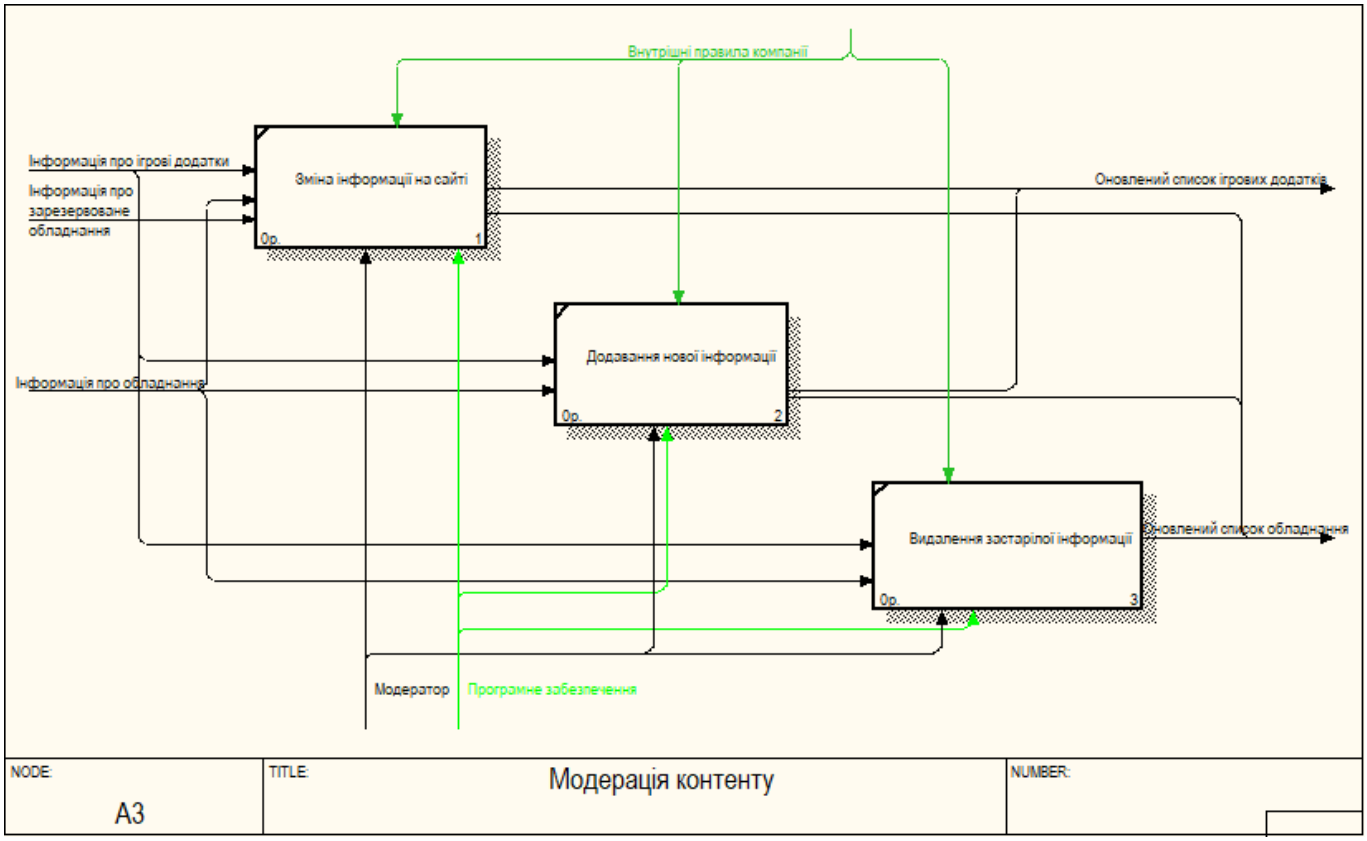

Рисунок 2.7 – Діаграма декомпозиції підсистеми «Модерація контенту»

На рис. 2.8 представлена декомпозиція підсистеми «Замовлення послуги» системи. Підсистема виконує один за найважливіших функцій, а саме надавання можливості замовлення.

Вибір послуги має надавати клієнту список доступних послуг. Вибір користувача має передаватись до наступної функції відповідно до того що обрав користувач.

Функція вибору конфігурації залежно від обраної послуги та актуального списку обладнання має надавати користувачу для вибору список актуальних конфігурацій серверів. Клієнт отримує функціонал для перегляду інформацію про вказаний сервер та перевірки наявності ігрових додатків, у яких клієнт зацікавлений. Вибір конфігурації визиває функцію вибору терміну дії та передає дані про обрану конфігурацію.

Останньою функцією є «Вибір терміну дії», яка отримує на вхід попередні дані. Дана функція формує замовлення та дає змогу клієнту вибрати тривалість дії послуги, після чого формує замовлення разом з заповненням фінансової інформації, зберігає у базу даних та відправляє квитанцію оплати до системи електронних розрахунків, де клієнт завершає транзакцію.

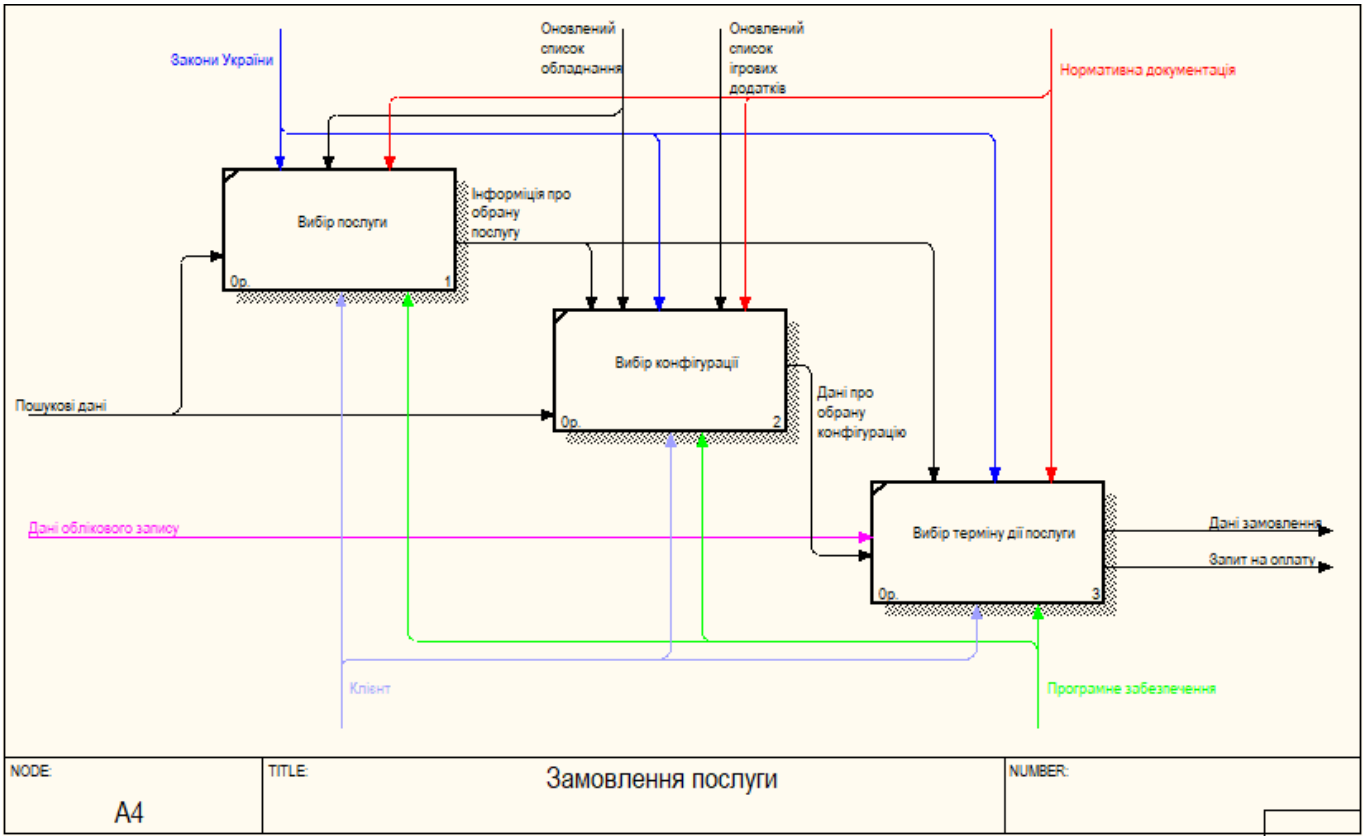

Рисунок 2.8 – Діаграма декомпозиції підсистеми «Замовлення послуги»

На рис. 2.9 представлена декомпозиція підсистеми «Обслуговування замовлення» системи. Підсистема відповідає останньому етапу формування замовлення, в якому система отримує підтвердження про успішну транзакцію та переходить до надання послуг користувачу. В рамках даної підсистеми можна мінімізувати контроль модератора щодо підтвердження транзакції та автоматизувати процеси резервування та активації обладнання.

Підтвердження замовлення є не автоматизованою функцією яку слід автоматизувати. На вхід отримує дані замовлення клієнта та отримує квитанцію про надходження оплати. Модератор перевіряє наявність оплати, вільність відповідного сервера та відправляє необхідні дані клієнту, щодо налаштування програмного засобу для підключення до сервера.

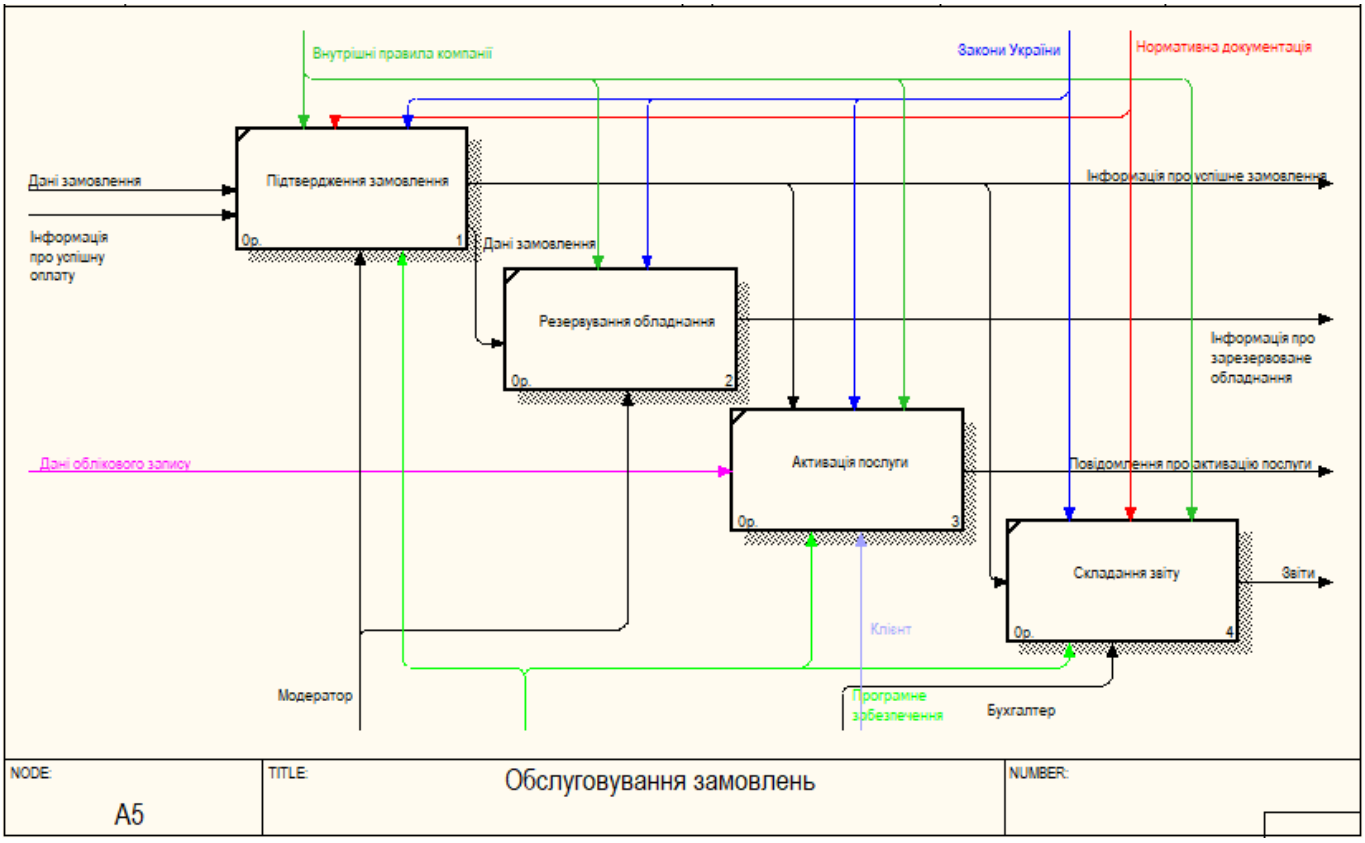

Рисунок 2.9 – Діаграма декомпозиції підсистеми «Обслуговування замовлення»

DFD-діаграми надають можливість описувати потоки інформації між частинами системи та між системою та її зовнішнім оточенням. Завдяки цьому DFD діаграми застосовуються для створення моделей інформаційного обміну системи.[16]

Для виявлення та потоків даних та формування моделі використовується DFD діаграма у якій відображені наступні елементи нотації:

– зовнішні сутності системи(External Entity);

– процеси(Process);

– сховища даних(Data store);

– потоки даних між ними(Data flow).

Дана вид нотації діаграм дозволяє виявити та зобразити зв’язки між внутрішніми підсистемами та зовнішнім середовищем.

На рис. 2.8 представлена концептуальна DFD-діаграма, що зображує погляд на підсистеми з точки зору адміністратора. На концептуальній діаграмі потоків даних додаються наступні зовнішні сутності: клієнт, модератор та банк-партнер. Від зовнішньої сутності «Клієнт» система отримує:

– реєстраційні дані;

– логін та пароль;

– пошукові дані;

– нові дані.

Вихідними даними до клієнта є:

– повідомлення про налаштування;

– список замовлень(дати, склад замовлення, термін дії активних послуг);

– повідомлення про додавання друзів;

– інформація про успішне замовлення.

Від зовнішньої сутності «Модератор» система отримує:

– інформацію про обладнання;

– інформацію про ігрові додатки;

– логін та пароль.

Вихідними даними до модератора є: звітність та результат неефективного проектування у вигляді списку очікуючих замолень.

Згідно діаграми система виконує певні взаємодії з банкам-партнерами, створюючи запит на проведення фінансової тарнзації клієнта та повертає інформацію про успішну оплату послуги.

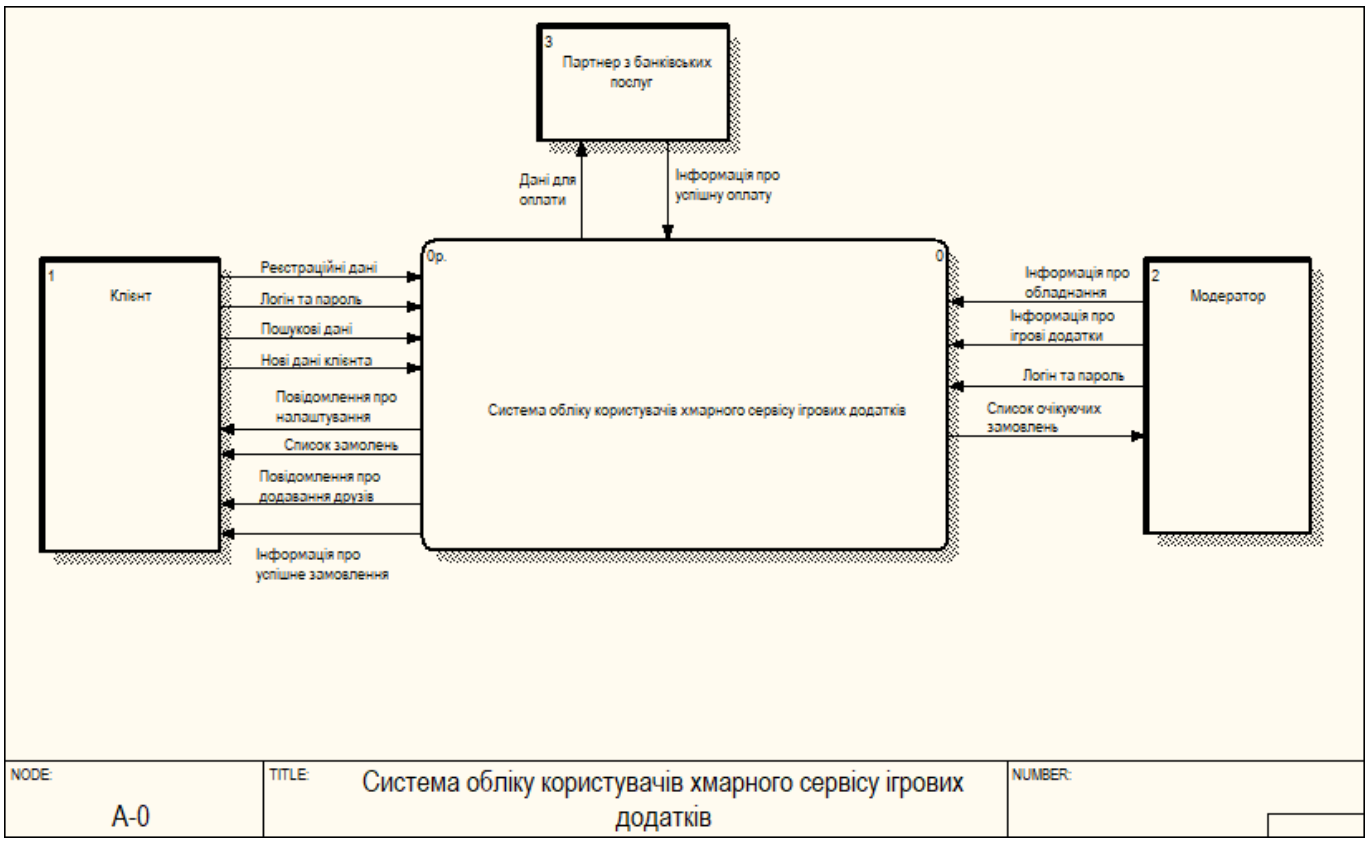

Рисунок 2.10 – Концептуальна діаграма DFD

Декомпозиція концептуальної діаграми DFD виходячи діаграми IDEF0 дозволяє провести потоки даних між підсистемами та зовнішніми сутностями (Рисунок 2.10).

Для опису основних підсистем та потоків даних необхідно визначити наступні сховища даних(Data Store):

– клієнти – найменування «user». Перелік атрибутів сховища: логін, пароль, емейл, роль, дата народження, додаткова інформація. Сховище даних зберігає реєстраційні дані користувачів та передає інформацію про користувача підсистемам керування облікового запису й замовлення послуги;

– обладнання – найменування «server». Перелік атрибутів сховища: назва серверу, набір обладнання, дата активації, статус серверу, ціна, опис. Сховище даних зберігає інформацію актуальних серверів, оновленням якої займається модератор. Бере участь у формування списків серверів для інформування користувачів;

– ігрові додатки – найменування «game». Перелік атрибутів сховища: назва гри, жанри, опис ігрового додатка, ціна, дата видання. Сховище даних зберігає

інформацію про ігрові додатки, які наявні у хмарному сервісі. Інформація оновлюється модератором.

– замовлення клієнтів – найменування «order». Перелік атрибутів сховища: тип послуги, номер серверу, ціна, статус замовлення, коментар, тривалість, дата замовлення, ідентифікаційних номер клієнта. Дана сутність відповідає за збереження замовлень. Замовлення записуються до сховища даних та змінюють статус залежно від дій модератора. На основі замовлень формується звіт.

– послуги – найменування «service». Перелік атрибутів сховища: назва послуги, опис, коефіцієнт тривалості, коефіцієнт ціни. Сховище даних зберігає дані про актуальні послуги. Налаштування для оформлення замовлення також надаються з даного сховища. Інформація оновлюється модератором.

Рисунок 2.11 – Діаграма декомпозиції DFD

Для створення наступного рівня декомпозиції були створені наступні діаграми для кожної з підсистем:

– декомпозиція підсистеми «Реєстрація та авторизація»(Рисунок 2.10). Підсистема використовує сховище «user» для перевірки логіну, пароля та емейл, запису інформації нового клієнта та встановлення ролі клієнта;

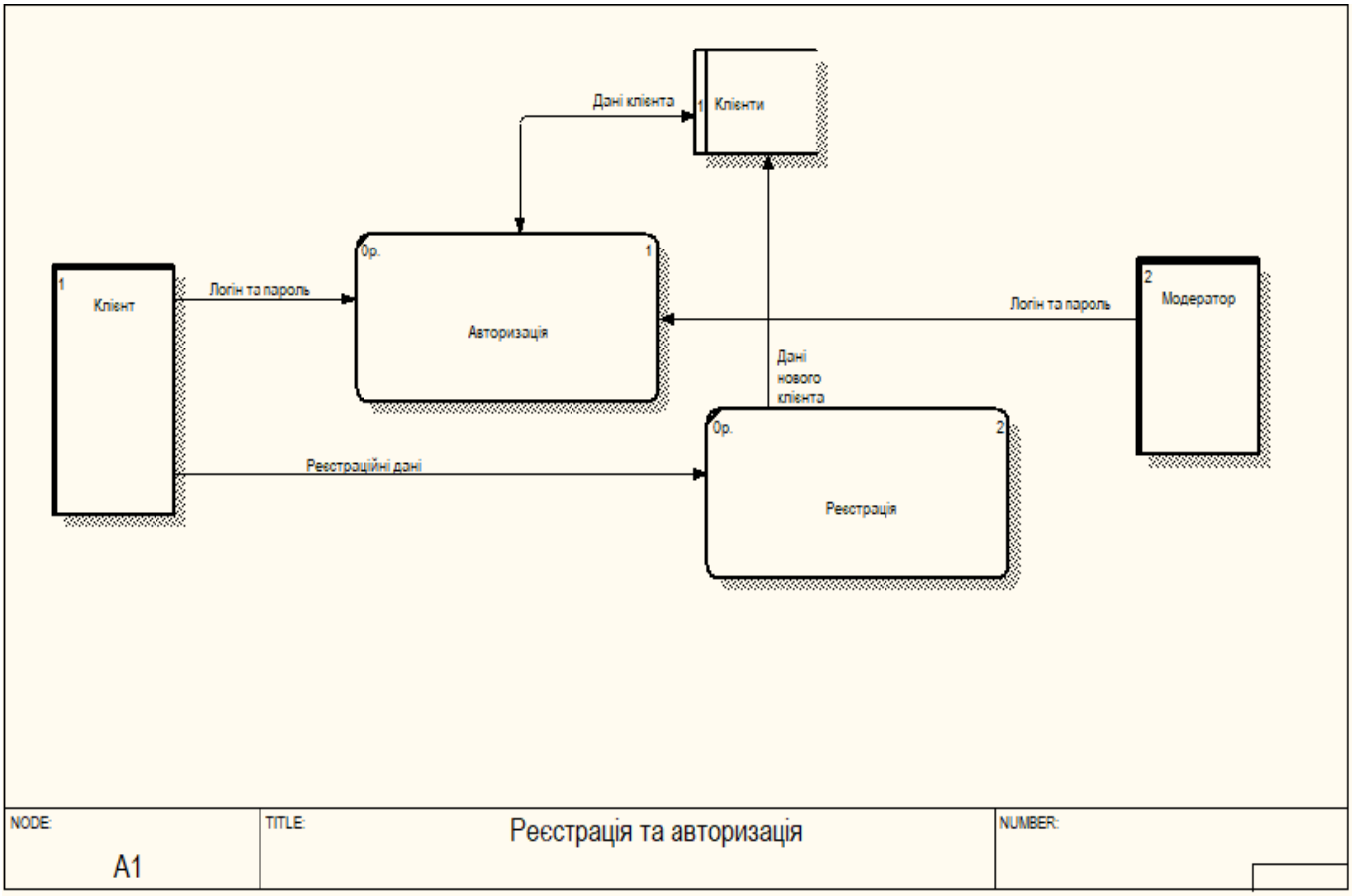

Рисунок 2.12 – Діаграма декомпозиції підсистеми «Реєстрація та авторизація»

– декомпозиція підсистеми «Керування облікового запису»(Рисунок 2.13). Підсистема використовує сховище «user» для редагування інформації користувачем, отримання списку клієнтів для пошуку та сховище «order» для пошуку активних послуг та зміни статус послуг;

– декомпозиція підсистеми «Модерація контенту»(Рисунок 2.14). Підсистема використовує сховища «server» та «service». Надає функціонал модератору для оновлення інформації у вказаних сховищах, а саме функції додавання нових серверів та ігрових додатків, редагування та видалення неактуальної інформації. Відповідні потоки даних до відповідних функцій заміни, додавання та видалення інформації було зображено на рисунку 2.14;

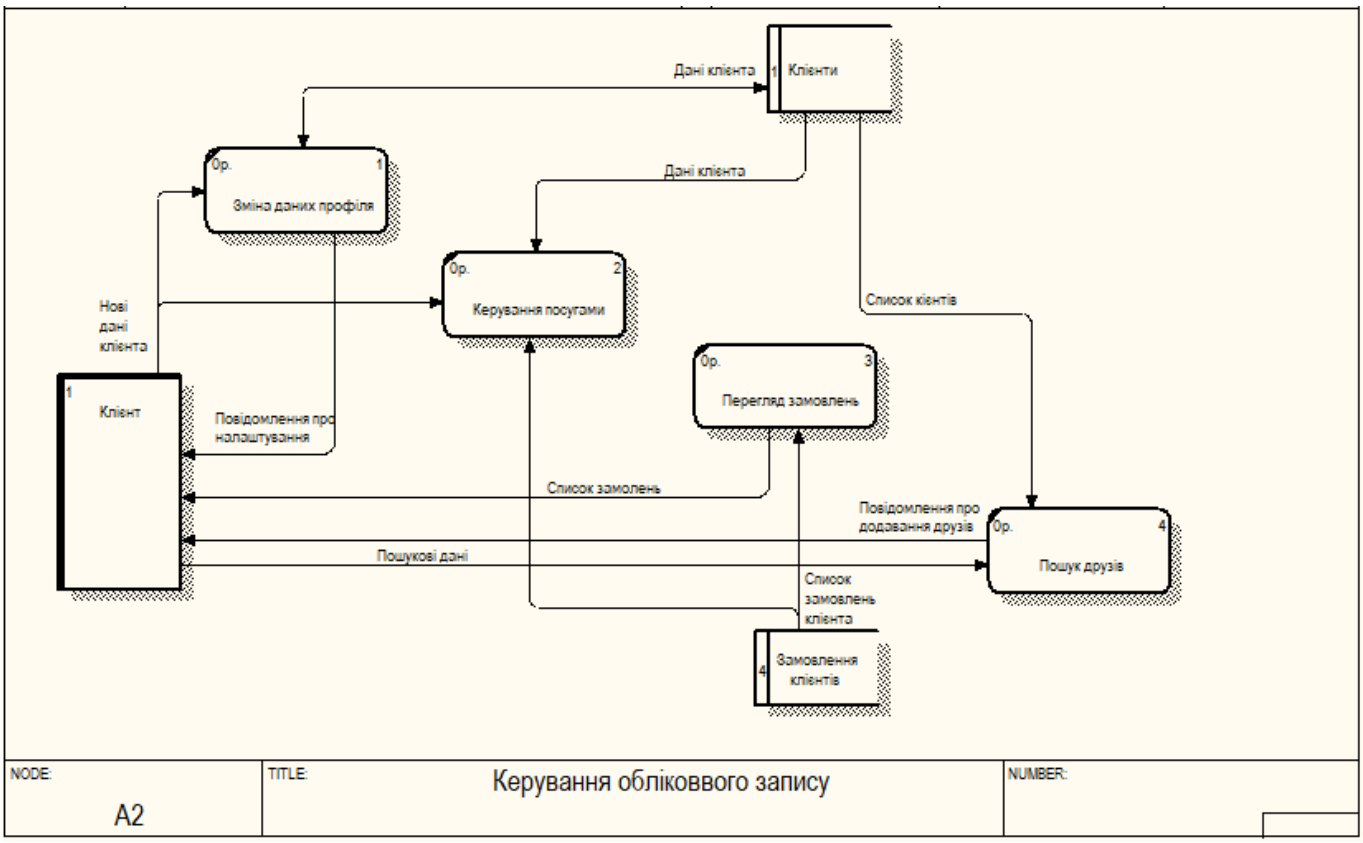

Рисунок 2.13 – Діаграма декомпозиції підсистеми «Керування облікового запису»

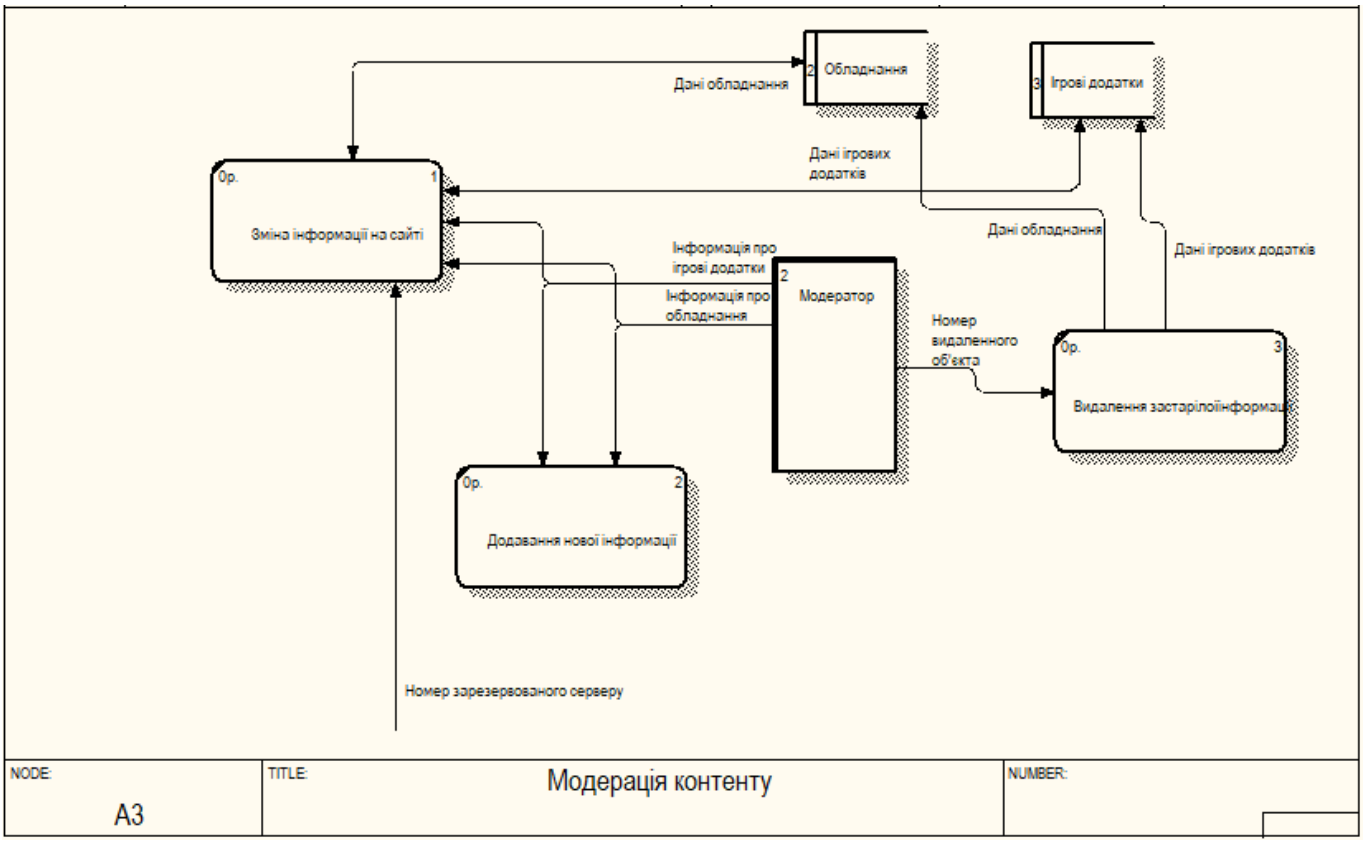

Рисунок 2.14 – Діаграма декомпозиції підсистеми «Модерація контенту»

– декомпозиція підсистеми «Замовлення послуги»(Рисунок 2.15). Підсистема використовує сховище «user», «service», «order», «game», «order». Зі сховища користувачів отримує дані авторизованого користувача. Зі сховища послуги отримує список послуг, які в подальшому будуть відображені користувачу. Зі сховища обладнання надаються дані про сервери, які використовуються функціях пошуку та презентації користувачу. Опис серверів виконується за допомогою сховища ігрових додатків, який використовуються функціях пошуку та презентації користувачу. Сформоване замовлення передається на зберігання до сховища замовлень клієнта;

– декомпозиція підсистеми «Обслуговування замовлень»(Рисунок 2.16). Даний процес використовує сховище обладнання для зміни статусу серверу та сховище замовлень клієнтів, щоб сформувати список очікуваних замовлень та звіт.

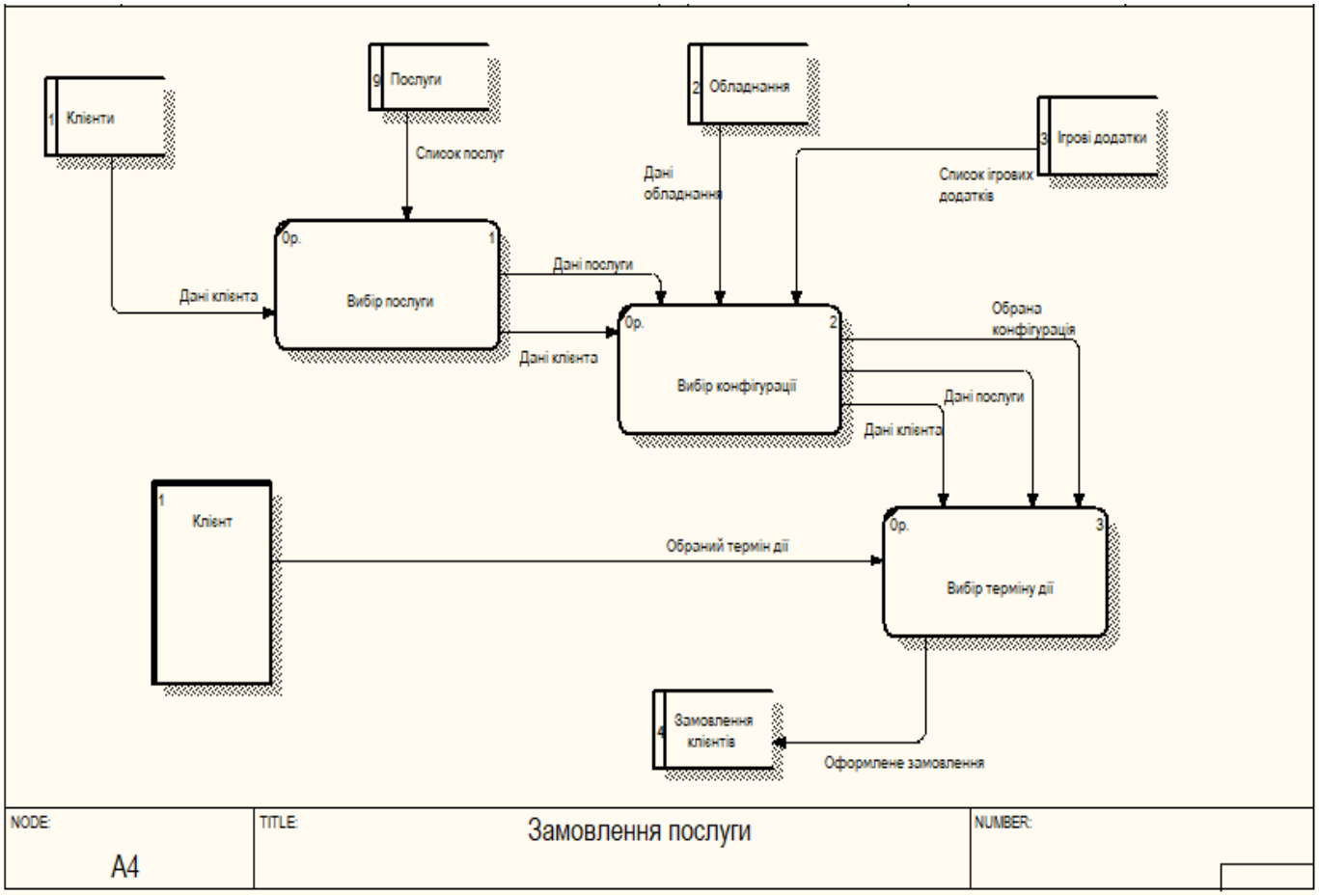

Рисунок 2.15 – Діаграма декомпозиції підсистеми «Замовлення послуги»

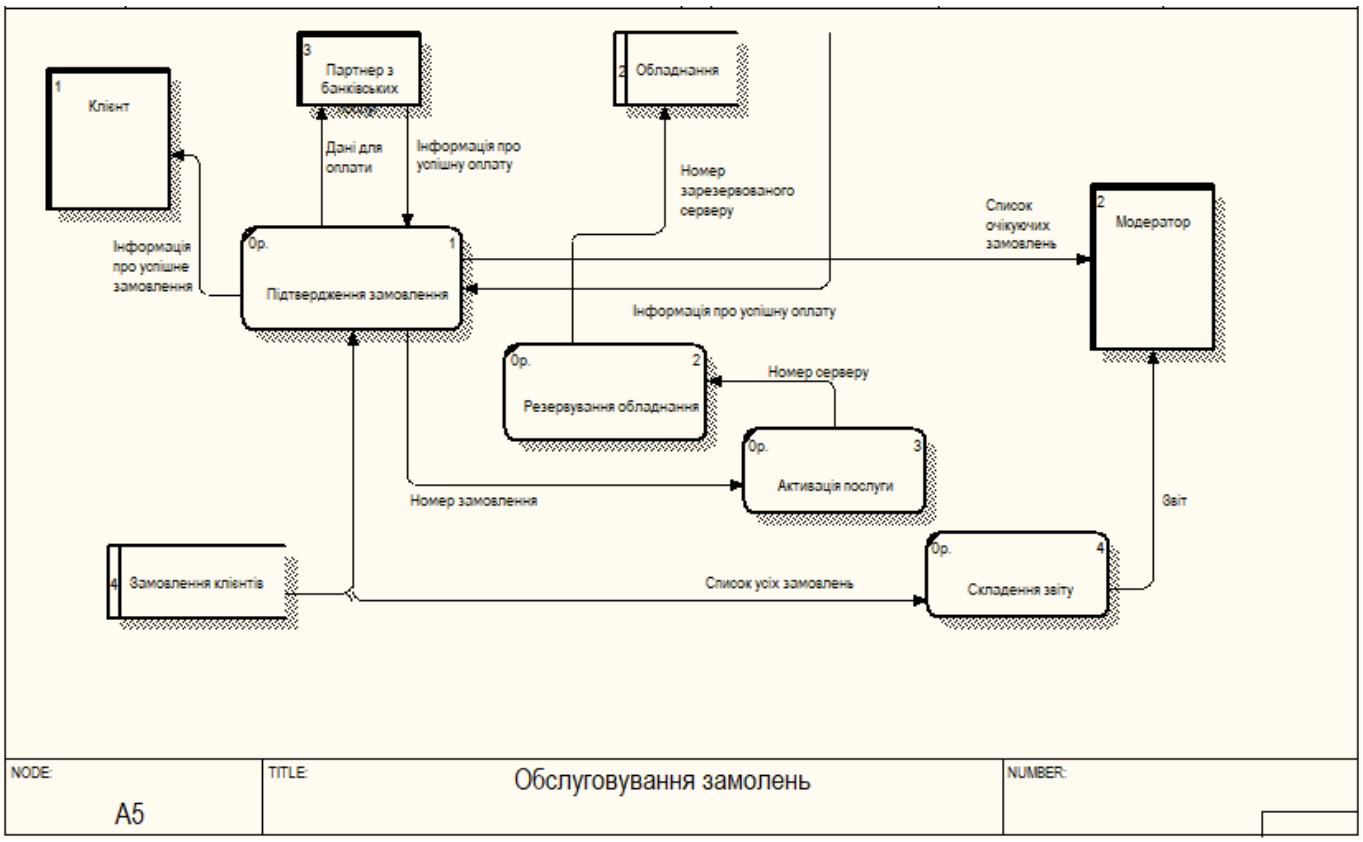

Рисунок 2.16 – Діаграма декомпозиції підсистеми «Обслуговування замовлень»

Етапом який завершує опис системи обліку хмарного сервісу це опис варіантів використання. Більш зручним інструментом для опису є UML.

Діаграма UML - це часткове графічне зображення моделі системи (або в стадії розробки, або вже використовується). Він містить графічні елементи, такі як вузли, пов’язані з ребрами, але може містити додаткову документацію, записану як текст.

UML забезпечує засіб візуалізації архітектурних креслень системи на схемі, включаючи такі елементи[17], як:

– будь-які види діяльності;

– окремі компоненти системи та те, як вони можуть взаємодіяти з іншими програмними компонентами;

– як буде працювати система;

– як взаємодіють сутності з іншими компонентами та інтерфейсами;

– зовнішній інтерфейс користувача.

UML виявився настільки успішним, що врешті-решт він закінчився використанням в інших областях, таких як бізнес-моделювання чи моделювання непрограмних систем.

Мова визначає дві категорії діаграм, хоча формального заборони на поєднання різних типів діаграм не існує.

Структурні діаграми показують статичну структуру системи та її частин, а також те, як ці частини співвідносяться між собою на різних рівнях абстракції та реалізації. Елементи на структурній діаграмі представляють значущі поняття системи і можуть включати абстрактні, реальні та реалізаційні концепції. Діаграми поведінки, навпаки, показують динамічну поведінку об’єктів у системі, що можна описати як низку змін системи в часі. [18]

Опис варіантів використання системи обліку хмарного сервісу зображена на рисунку 2.17.

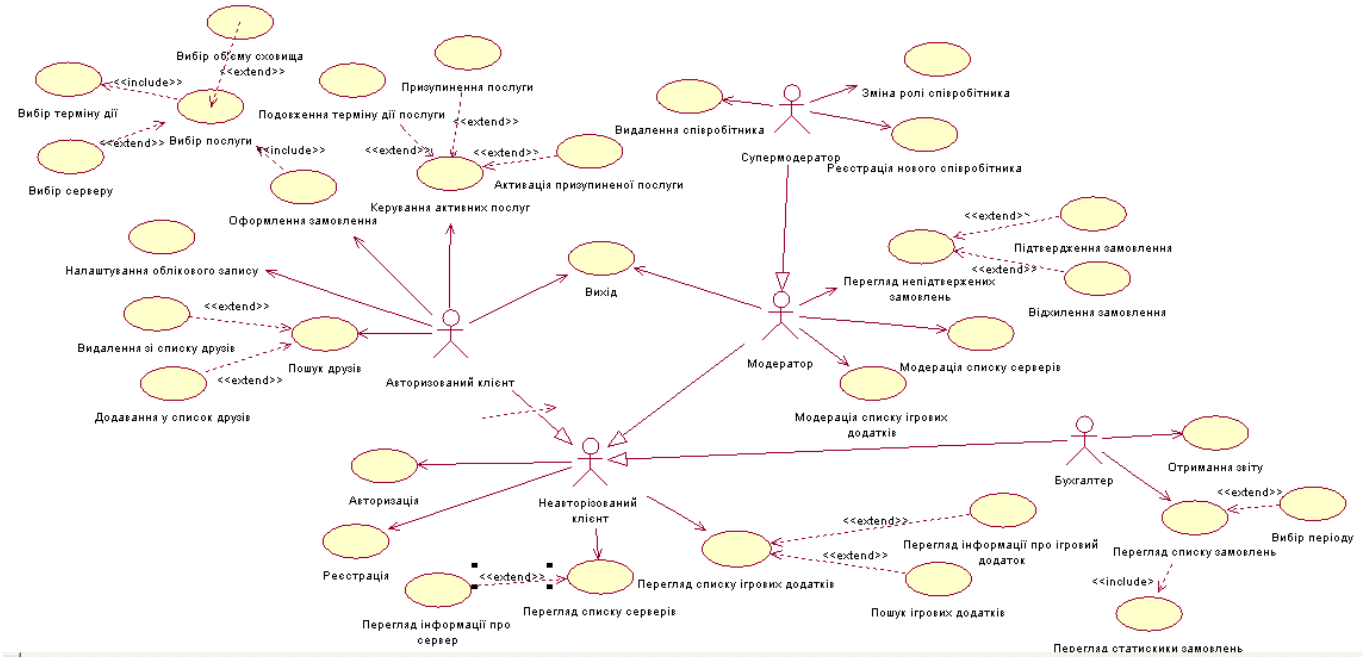

Рисунок 2.17 – Діаграма варіантів використання

Актор - це позначення сутності у UML, яка взаємодії з системою.

Актор може бути зовнішній об’єкт, внутрішній суб’єкти системи або інша система. На даній діаграмі зображені такі актори:

– неавторизований користувач – це користувач, який моє можливості лише отримувати загальну інформацію про послуги хмарного сервісу та змінити статус актора на інший. Детально про можливості: авторизація, реєстрація, перегляд списків та інформації про сервери, ігрові додатки, послуги;

– авторизований клієнт – це користувач, який може користуватись послугами та має наступні можливості: пошуку друзів, оформлення замовлення, керування

активними послугами та редагування свого профілю. Разом з цим актор має же перелік функціоналу що і неавторизований користувач.

– модератор – це користувач, що є керуючою особою технічного рівня. Має наступний функціонал: додавання, редагування та видалення інформації про сервери, ігрові додатки та послуги, отримання списку замовлень та обробляти замолень.

– супермодератор – це користувач, що є керуючою особою організаційного рівня, має функціонал модератора та має додатковий функціонал для регулювання персоналу у системі.

– бухгалтер – це користувач. Функціонал: формування даних про замовлення та складання звітів.

Варіант використання «Авторизація» забезпечує відображення форми вводу даних авторизації клієнта, виконує перевірку введених даних на існування.

Варіант використання «Реєстрація» відповідає за функцію вводу даних для реєстрації нового клієнта, для подальшого отримання доступу к послугам хмарного сервісу.

Варіант використання «Перегляд списку серверів» відповідає за функцію зображення списку актуальних серверів компанії для подальшого замовлення.

Варіант використання «Перегляд інформації про сервер» є розширенням варіанту використання «Перегляд списку серверів» та підтримується функцією надання списку актуальних конфігурацій серверів та зі списком встановлених ігрових додатків.

Варіант використання «Перегляд списку ігрових додатків» зображує функцію отримання списку ігрових додатків компанії для пошуку серверів.

Варіант використання «Пошук ігрових додатків» є розширенням варіанту використання «Перегляд списку ігрових додатків» та підтримується функцією пошуку ігрових додатків, які наявні у хмарному сервісі.

Варіант використання «Перегляд інформації про ігровий додаток» є розширенням варіанту використання «Перегляд списку ігрових додатків» та підтримується функцію відображення інформації про ігровий додаток, а також список серверів.

Варіант використання «Налаштування облікового запису» підтримується функцією редагування інформації профілю клієнта та встановлення певних налаштувань.

Варіант використання «Пошук друзів» підтримується функцією пошуку у базі клієнтів для активації послуги віртуальної мережі та сумісної гри.

Варіант використання «Додавання у список друзів» є розширенням варіанту використання «Пошук друзів» та підтримується функцією додавання знайденого користувача до віртуального серверу та списку друзів.

Варіант використання «Видалення зі списку друзів» є розширенням варіанту використання «Пошук друзів» та підтримується функцією виключення зі списку друзів обранихкориствачів.

Варіант використання «Керування активних послуг» підтримується функцією зображення списку активних та призупинених послуг.

Варіант використання «Призупинення послуги» є розширенням варіанту використання «Керування активних послуг» та підтримується функцією зміни статусу активної послуги на «призупинена».

Варіант використання «Активація призупиненої послуги» є розширенням варіанту використання «Керування активних послуг» та підтримується функцією зміни статусу призупиненої послуги на «активна».

Варіант використання «Подовження терміну дії послуги» є розширенням варіанту використання «Керування активних послуг» та підтримується функцією формування замовлення на подовження активної послуги.

Варіант використання «Оформлення замовлення» є включенням до варіанту використання «Вибір послуги» та підтримується функцією формування рекомендаційного списку актуальних послуг.

Варіант використання «Вибір послуги» є включенням до варіанту використання «Вибір терміну дії» та підтримується функцією зображення форми замовлення для клієнта.

Варіант використання «Вибір терміну дії» підтримується функцями зображення форми з термін дії, завершення оформлення замовлення та зберігання у базі даних.

Варіант використання «Вибір серверу» є розширенням варіанту використання «Вибір послуги» та підтримується функцією повернення рекомендаційного списку актуальних серверів компанії з можливість вибору серверу для подальшого оформлення замовлення.

Варіант використання «Вибір об’єму сховища» є розширенням варіанту використання «Вибір послуги» та підтримується функцією вибору необхідного об’єм хмарного сховища.

Варіант використання «Вихід» підтримується функцією виходу авторизованого користувача та змини його статус на «неавторизований».

Варіант використання «Модерація списку ігрових додатків» підтримується функцією оновлення списку ігрових додатків для актора «Модератор».

Варіант використання «Модерація списку серверів» підтримується функціями оновлення списку серверів та зміни списки встановлені ігрові додатки на серверах для актора «Модератор».

Варіант використання «Перегляд непідтверджених замовлень» підтримується функцією формування списку непідтверджених замовлень.

Варіант використання «Підтвердження замовлення є розширенням варіанту використання «Перегляд непідтверджених замовлень» та підтримується функцією зміни статусу замовлень на «пдтверджено».

Варіант використання «Відхилення замовлення» є розширенням варіанту використання «Перегляд непідтверджених замовлень» та підтримується функцією зміни статусу замовлень на «відхилено».

Варіант використання «Реєстрація нового співробітника» підтримується функцією реєстрації нового співробітника.

Варіант використання «Зміна ролі співробітника» підтримується функцією керування обліковими записами користувачів для актора «Супермодератор».

Варіант використання «Видалення співробітника підтримується функцією керування обліковими записами користувачів для актора «Супермодератор».

Варіант використання «Перегляд списку замовлень» є включенням до варіанту використання «Перегляд статистики замовлень» та підтримується функцією формування звітів для користувачів з правами бухгалтера.

Варіант використання «Перегляд статистики замовлень» підтримується функцією формування статистики для користувачів з правами бухгалтера.

Варіант використання «Вибір періоду» розширяє можливості прецеденту «Перегляд списку замовлень» та підтримується функцією керування списком замовлень за певний період.

Варіант використання «Отримання звіту підтримується функцією формування звітів для користувачів з правами бухгалтера. Це дозволяє отримати від системи звіт доходів.

Застосування системного підходу дозволило сформувати модель системи основного бізнес-процесу, що дозволяє знайти неоптимальні елементи та в подальшої їх успішної автоматизації.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОСЛІДЖЕНОЇ СИСТЕМИ

3.1 Проектування системи

Виходячи з опису системи за допомогою IDEF0 діаграми складено функціональні вимоги до автоматизованої системи.

Система повинна реалізовувати наступні функції для користувачів:

– реєструвати нового облікового запису;

– авторизуватися у системі;

– оформлювати замовлення;

– налаштовувати свій профіль;

– перегляду своїх замовлень;

– пошуку за назвою гри та їх розробниками;

– перегляду інформації про гру та їх розробників;

– пошуку інших облікових записів користувачів та додавання їх у «друзі»;

– керування активними послугами.

Система повинна реалізовувати наступні функції для модераторів:

– авторизуватися у системі;

– перегляд «очікуючих» замовлень;

– підтвердження замовлення;

– зміна статусу серверу;

– внесення та редагування інформації про сервер, гру та обладнання;

– перегляд інформації про гру та їх розробників.

Система повинна реалізовувати наступні функції для бухгалтеру:

– авторизуватися у системі;

– отримувати інформацію про всі замовлення.

Розроблювані програмні компоненти системи обліку мають виконати завдання автоматизації роботи компанії, що надає послуги хмарного сервісу ігрових додатків. Виходячи з цілей створення системи, розглянемо системні вимоги до неї.

Системні вимоги до системи автоматизації:

а) клієнтська частина системи обліку має бути створена за допомогою HTML та з використанням мови програмування Python разом з компонентами бібліотеки Django. Результатом є веб-сторінки зі стилями CSS;

б) серверна частина системи обліку має бути розроблена з використанням мови програмування Python, фреймворка Django та СУБД MySQL. Серверну частину встановити на будь-який комп’ютер з операційною системою для якої існують версії Python, Django та MySQL, а це практично будь яка операційна система, що використовується на сервері;

в) інтерфейс системи має в будь-якому сучасному браузері , що підтримує HTML5 та CSS3. При чому як на настільних комп’ютерах, так і на мобільних пристроях.

До інтерфейсу клієнтської частини входять наступні групи елементів:

– блок реєстрації, якай складає поля вводу електронної пошти, логіну, паролю та підтвердження паролю. Також є кнопка «Зареєструватись», яка запускає процедуру реєстрації;

– блок відображення сторінкового списку ігор;

– панель пошуку, де є поле текстового вводу, меню вибору пошуку(за назвою, за розробником) та кнопки, яка викликає процедуру пошуку;

– сторінка профілю клієнта, за можливістю редагувати інформацію та керувати активними послугами;

– блок авторизації, який складає з полів вводу логіна і пароля та кнопки, яка викладає процедуру авторизації;

– сторінка замовлень клієнта;

– сторінка з формами оновлення інформації про сервер, гру та обладнання.

Діаграми послідовності UML - це діаграми взаємодії, які детально описують спосіб виконання операцій. Вони фіксують взаємодію між об’єктами в контексті співпраці. Діаграми послідовності - це фокус на часі, і вони наочно показують порядок взаємодії, використовуючи вертикальну вісь діаграми, щоб представити час, які повідомлення надсилаються та коли. [19]

Діаграми послідовності фіксують:

– взаємодія, яка відбувається у співпраці, яка реалізує варіант використання або операцію (діаграми екземплярів або загальні діаграми)

– взаємодії на високому рівні між користувачем системи та системою, між системою та іншими системами або між підсистемами (іноді їх називають діаграмами системних послідовностей)

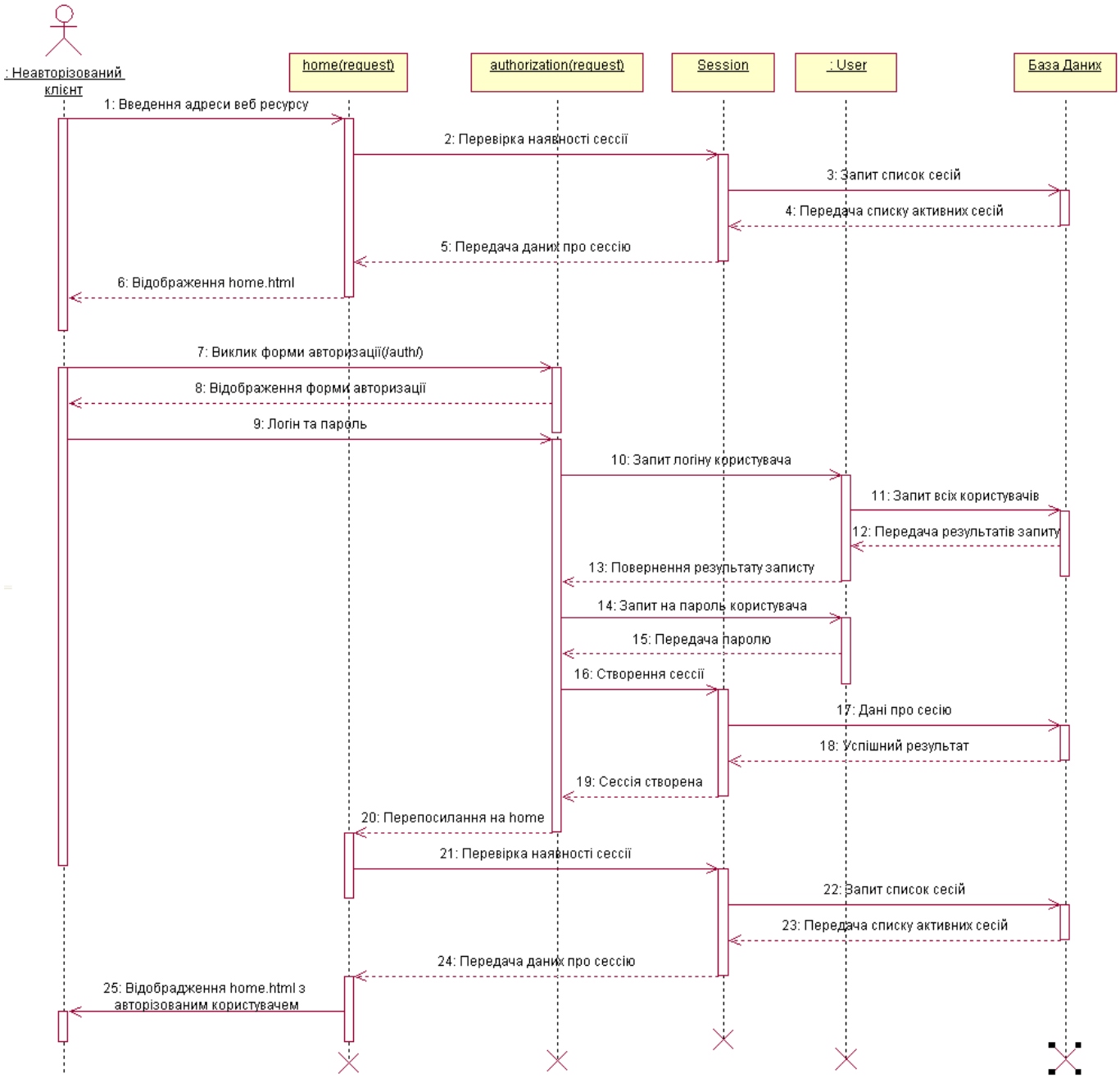

Рисунок 3.1 – Діаграма послідовності дій «Авторизації»

На рис. 3.1 зображена побудована діаграма послідовності дій для майбутньої функції «Авторизації» підсистеми «Авторизації та реєстрація», яку ініціює неавторизований користувач. Користувач створює запит вводом адреси веб ресурсу для зображення головної сторінки. Наступним кроком система перевіряє чи активна сесія користувача, якщо ні то система надсилає головну сторінку з функціоналом неавторизованого користувача, тобто присутні кнопки «Реєстрація» та «Авторизація». При альтернативній ситуації зображує сторінку з функціоналом авторизованого клієнта.

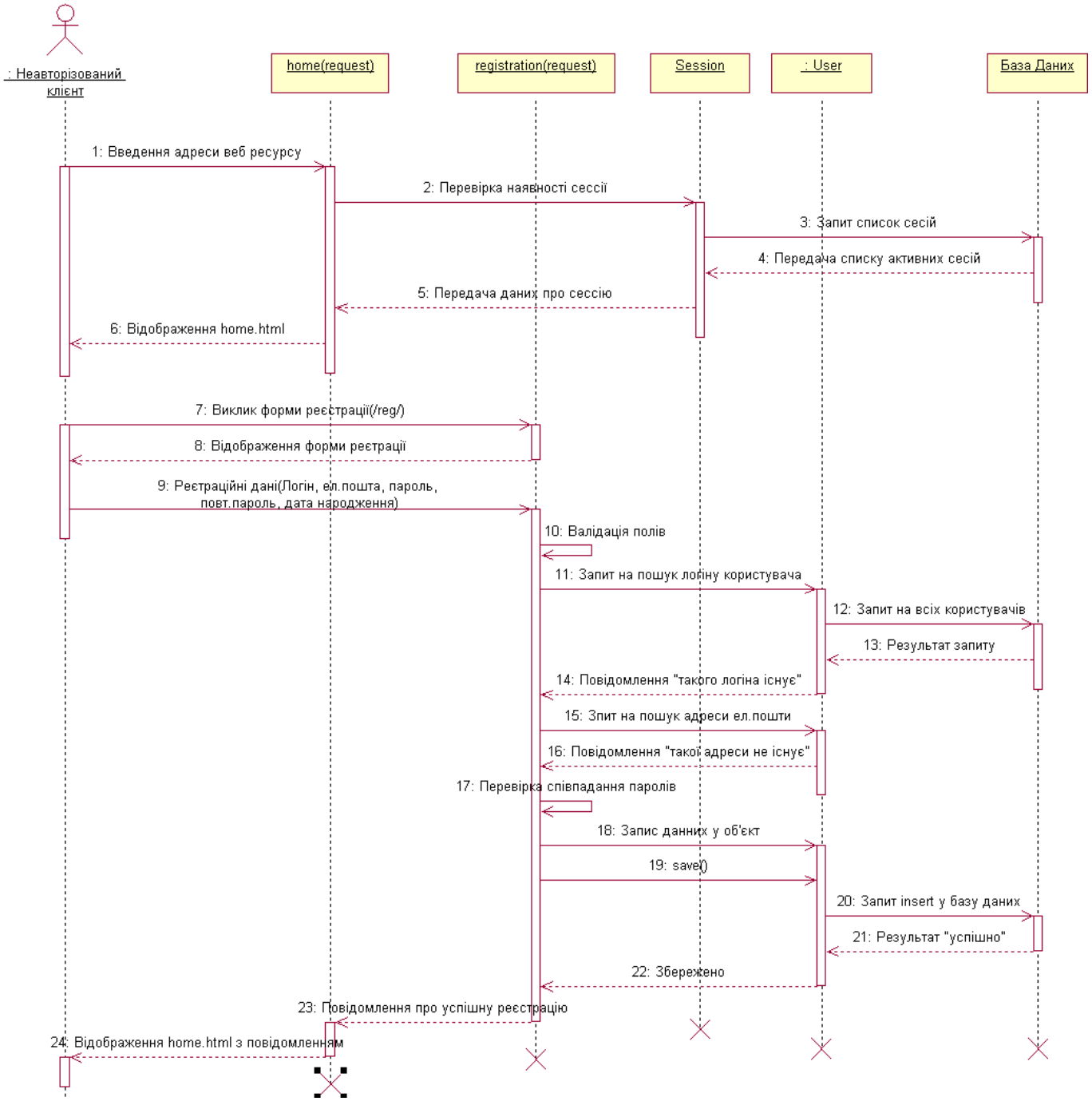

Рисунок 3.2 – Діаграма послідовності дій «Реєстрації»

Наступним кроком є натискання на кнопку «Авторизація» після чого викликається метод authorization(request), який надсилає форму авторизації. Користувач вводить ідентифікаційні дані та натискає на кнопку «авторизуватись». Викликається метод авторизації, який обробляє дані. Обробка даних йде наступним чином: задається запит знаходження користувача у моделі даних, модель даних відправляє запит на надання списку користувачів, після отримання списку проводитися пошук заданого логіна, якщо такий існує надсилає підтвердження про

існування користувача та інформація якого зберігається у моделі, задається запит на отримання пароля знайденого користувача, перевірка збігу паролів. У випадку успішної обробки, створюється активна сесія у базі даних та у браузері користувача. За цим йде посилання на головну сторінку, яка перевіряє наявність активної сесії користувача для відображення авторизованої сторінки.

На рис. 3.2 зображена діаграма послідовності дій для сервісу «Реєстрації» системи обліку, що ініціюється неавторизованим користувачем. Початковим кроком є ввод адреси веб ресурсу, де до системи надходить запит на відображення головної сторінки. Здійснюється перевірка активної сесія користувача для відображення певних елементів сторінки. У випадку відсутності сесії відображується кнопка «Реєстрація». При натисненні на цю кнопка викликається метод registration(request), який створює форму реєстрації.

Вхідні дані що вводить користувач: логін, адреса електронної пошти, пароль, підтвердження пароля, дату народження. Активація кнопки «Зареєструватись» метод реєстрації отримує введені дані та починає їх обробка.

Метод реєстрації проводить перевірку полів у відповідність до формату, далі створює запит на пошук вказаного логіна у моделі даних. Модель даних здійснює запит на надання списку існуючих користувачів, після повернення списку проводитися пошук вказаного логіна, інформація про користувачів зберігається у моделі, задається запит на пошук вказаного емейлу. У випадку відсутності вказаних даних проводиться перевірка вказаних паролів.

Наступним кроком є зберігання вхідних даних у модель «user» та викликає функцію save(), що зберігає дані моделі у базі даних. Зворотнім зв’язком з бази даних отримується повідомлення про успішне зберігання. Далі метод реєстрації посилає на головну сторінку та передає повідомлення, що вказаний користувач був зареєстрований.

Результатом дії даної послідовності є надання можливості авторизуватися у системі.

На рис. 3.3 зображена діаграма послідовності дій для сервісу «Оформлення замовлення» інформаційної системи для замовлення послуги оренди сервера, що виконується авторизованим користувачем. Оформлення замовлення починається з переходу на розділ «Маркет», де до системи надходить запит на відображення сторінки послуг, який обробляє метод market(request). Для цього система перевіряє чи активна сесія користувача для відображення певних елементів сторінки. Авторизуватися у системі потрібно для безпосереднього оформлення замовлення.

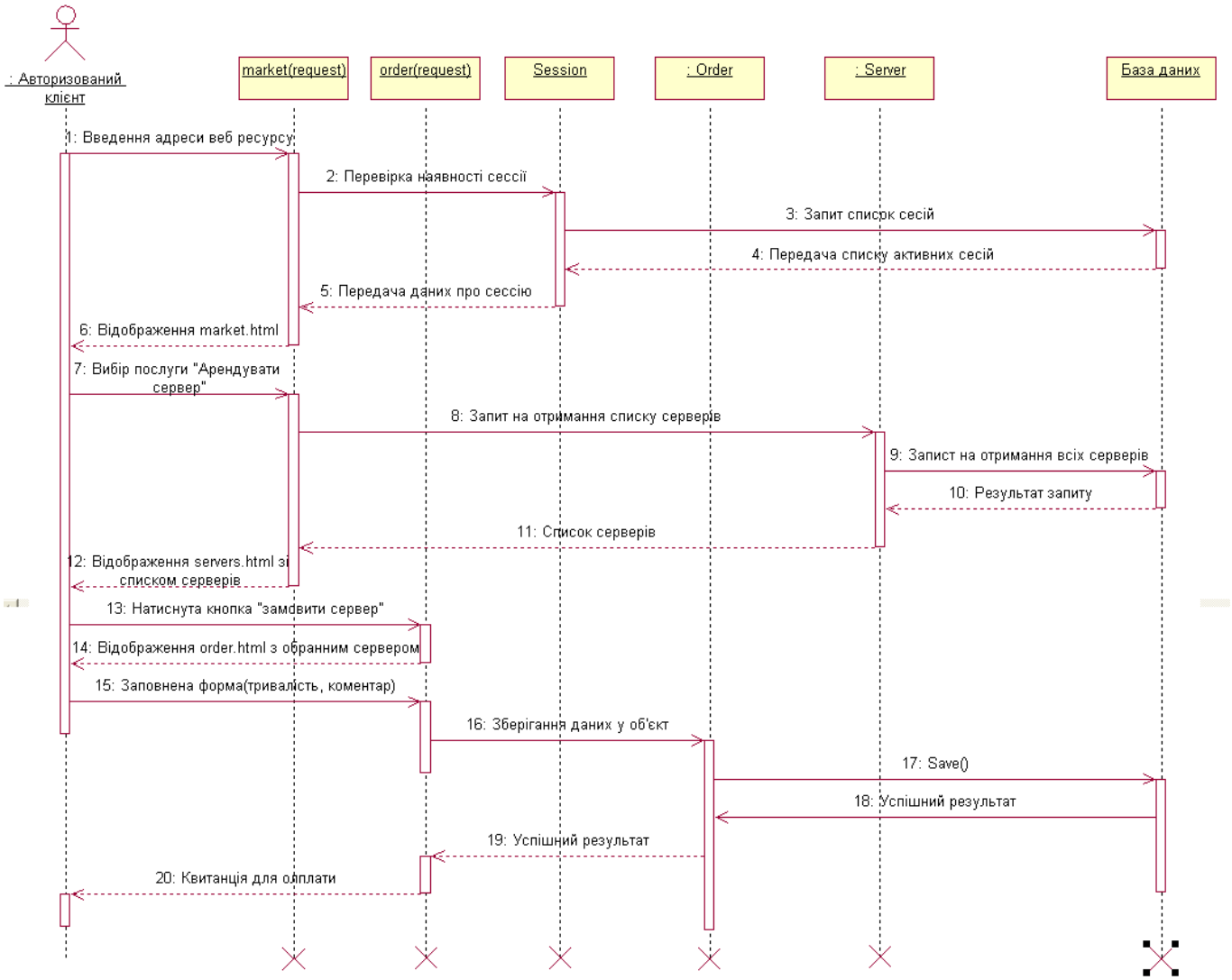

Рисунок 3.3 – Діаграма послідовності дій «Оформлення замовлення» послуги оренди сервера

На сторінці послуг користувач обирає послуги оренди серверу, що викликає запит до моделі «server» на отримання списку серверів. Модель формує запит до бази даних та отримує список серверів. Метод на основі списку серверів формує html сторінку відображення списку серверів.

Користувач обирає сервер та натискає кнопку замовити. Цей запит обробляє метод order(request), який формує html сторінку даних замовлення.

Користувач вказує тривалість, пише коментар та натискає кнопку «Замовити». Метод замовлень отримує вказані дані, записує їх у модель та викликає метод save(), який записує дані у базу даних с позначкою «новий». Отримавши позитивний результат, він формує квитанцію для оплати та відсилає її користувачу з повідомленням «Замовлення зроблене».

Діаграма кооперації використовуються, щоб показати, як об’єкти взаємодіють, виконуючи поведінку конкретного випадку використання або частини випадку використання. Поряд із діаграмами послідовностей, співпраця використовується дизайнерами для визначення та уточнення ролі об'єктів, які виконують певний потік подій у випадку використання. Вони є основним джерелом інформації, що використовується для визначення класових обов'язків та інтерфейсів.[20]

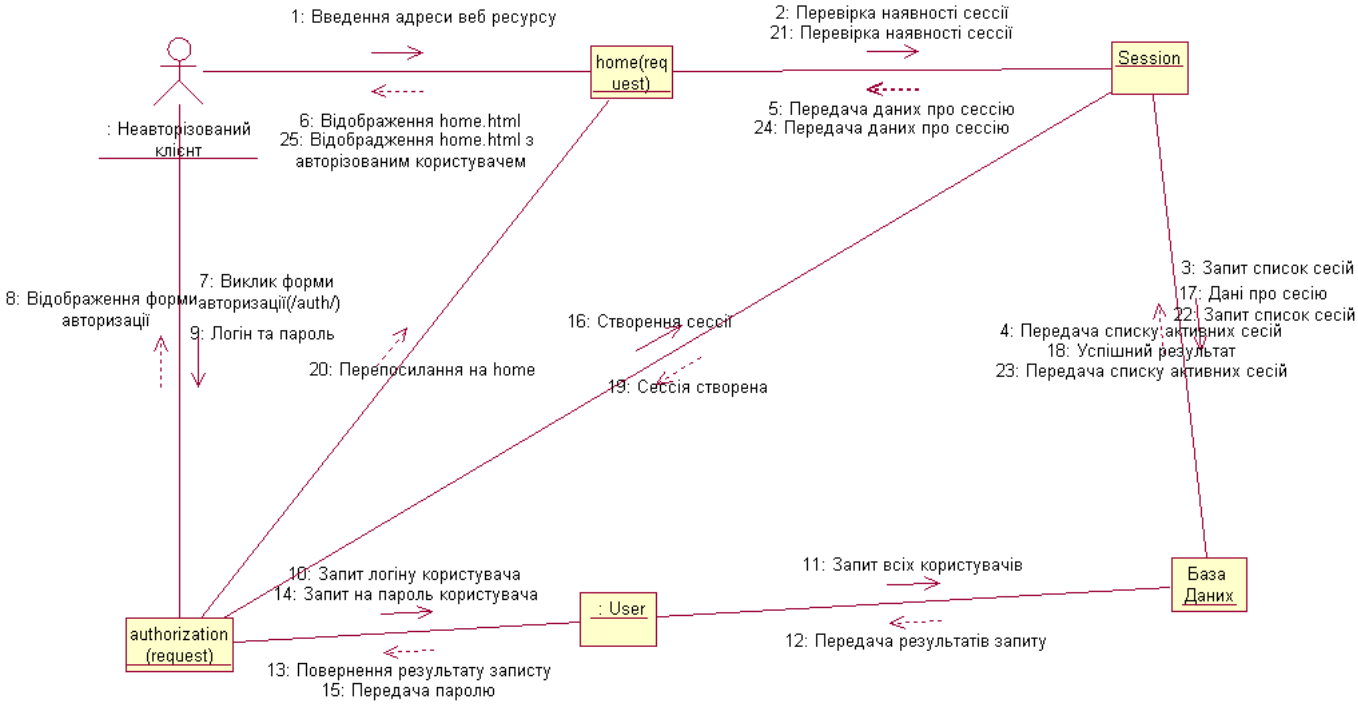

Рисунок 3.4 Діаграма кооперації «Авторизація»

Для визначання обміну повідомленнями між певними об’єктами та користувачем систем побудовані діаграми кооперації для наступних варіантів використання:

– авторизація. На рисунку 3.4 створена діаграму кооперації. З цієї діаграми можна визначити з якими елементами взаємодіє користувач та які повідомлення отримує від системи. Користувач взаємодіє за двома методами системи, які обробляють його запити.

– реєстрація. На рисунку 3.5 створена діаграма кооперації для реєстрації. На даній діаграмі можна побачити достатню кількість повідомлень між методом обробки реєстрації та моделлю даних. Це означає що присутність такої модель знижує навантаження на базу даних, так як потребує від бази даних лише один раз отримати дані користувачів та обробляти ці дані без необхідності створювати додаткові запити.

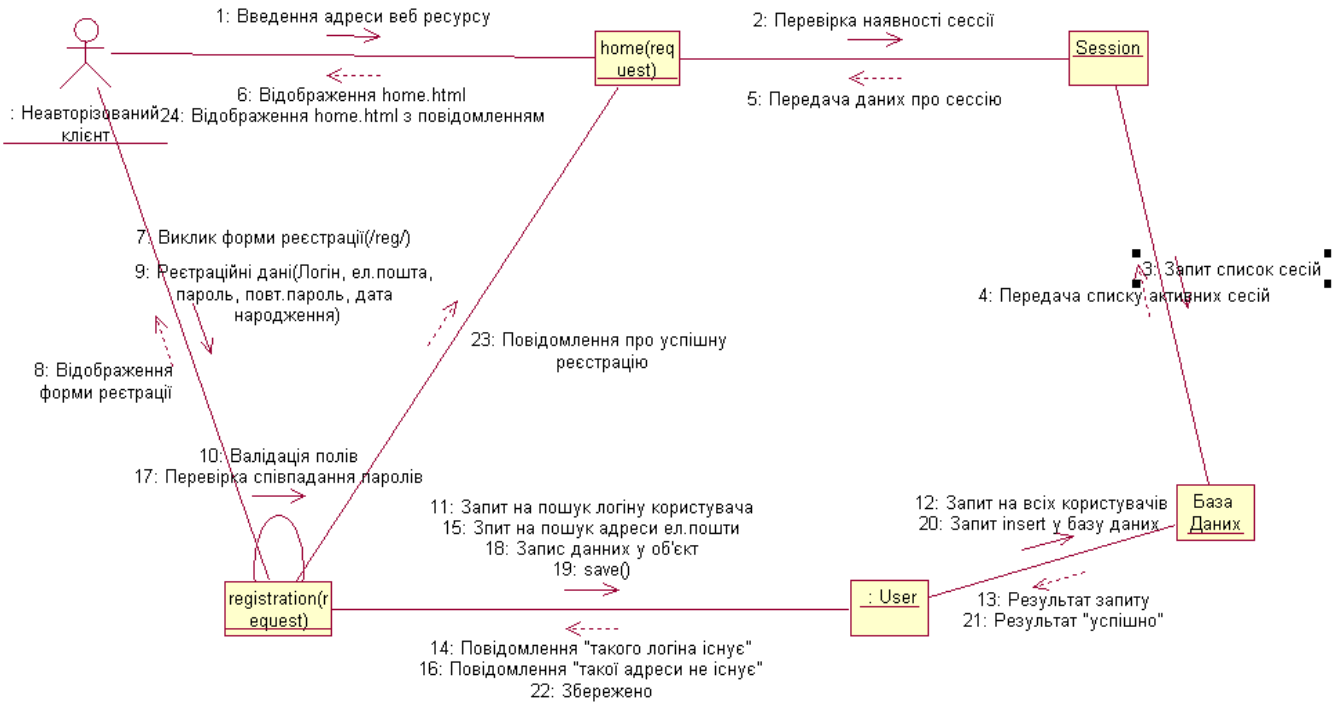

Рисунок 3.5 – Діаграма кооперації «Реєстрація»

– оформлення замовлення. На рисунку 3.6 створена діаграма кооперації для оформлення замовлень.

Операція перевірки стану сесії присутня у вказаному переліку функцій, так як це забезпечує механізм визначення певних можливостей кожному з акторів системи.

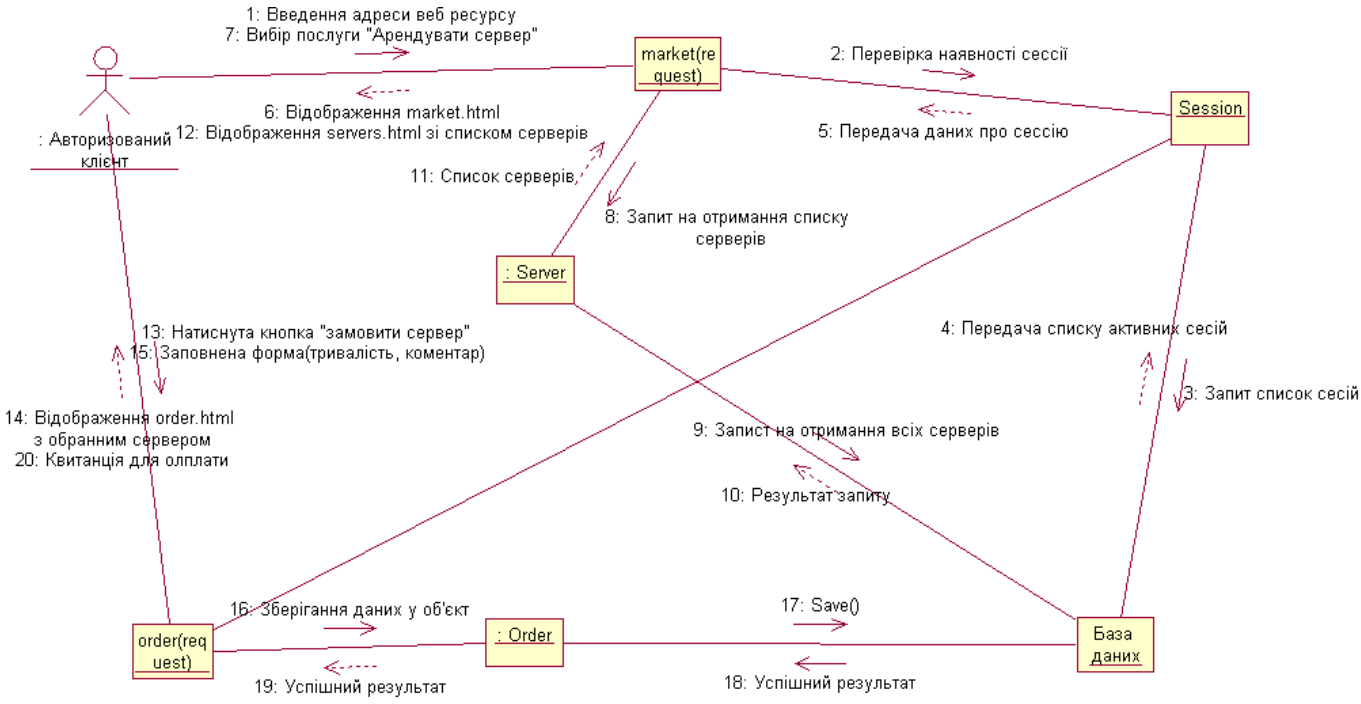

Рисунок 3.6 – Діаграма кооперації «Оформлення замовлення»

Діаграма станів (Statechart diagram) - це поведінкова діаграма, яка представляє поведінку системи з використанням кінцевих переходів стану. Діаграми стану також називаються машинами стану і діаграмами стану. Ці терміни часто використовуються як взаємозамінні. Діаграма стану використовується для моделювання динамічної поведінки класу у відповідь на час та зміну зовнішніх стимулів.

Використання діаграми стану:

– для викладу подій, відповідальних за зміну стану;

– для моделювання динамічної поведінки системи;

– для визначення реакцію предметів/класів на внутрішні або зовнішні дії.[21]

На рис. 3.7 зображена діаграма станів системи з точки зору клієнта. На цій діаграмі є наступні елементи:

– початковий стан - використовуємо заповнене чорним кольором коло, яке представляє початковий стан системи або класу.

– перехід - використовуємо суцільну стрілку для відображення переходу або зміни управління з одного стану в інший. Стрілка позначена подією, яка спричиняє зміну стану.

– стан - використовуємо округлий прямокутник для представлення стану. Стан представляє умови або обставини об'єкта класу в момент часу.

– приєднання - використовуємо округлу суцільну прямокутну смужку, щоб зобразити нотацію Приєднання з вхідними стрілками зі станів приєднання та вихідною стрілкою до загального стану цілі. Ми використовуємо нотацію приєднання, коли два або більше станів одночасно збігаються в один при виникненні події або подій.

– самоперехід - використовуємо тверду стрілку, що вказує назад до самого стану, щоб зобразити самоперехід. Можуть бути сценарії, коли стан об’єкта не змінюється після настання події.

– складений стан - використовуємо округлий прямокутник, щоб також представити складений стан. Представляє стан із внутрішніми діями, використовуючи складений стан.

– кінцевий стан - використовуємо заповнене коло в позначенні кола, щоб представити кінцевий стан на діаграмі автомата стану.

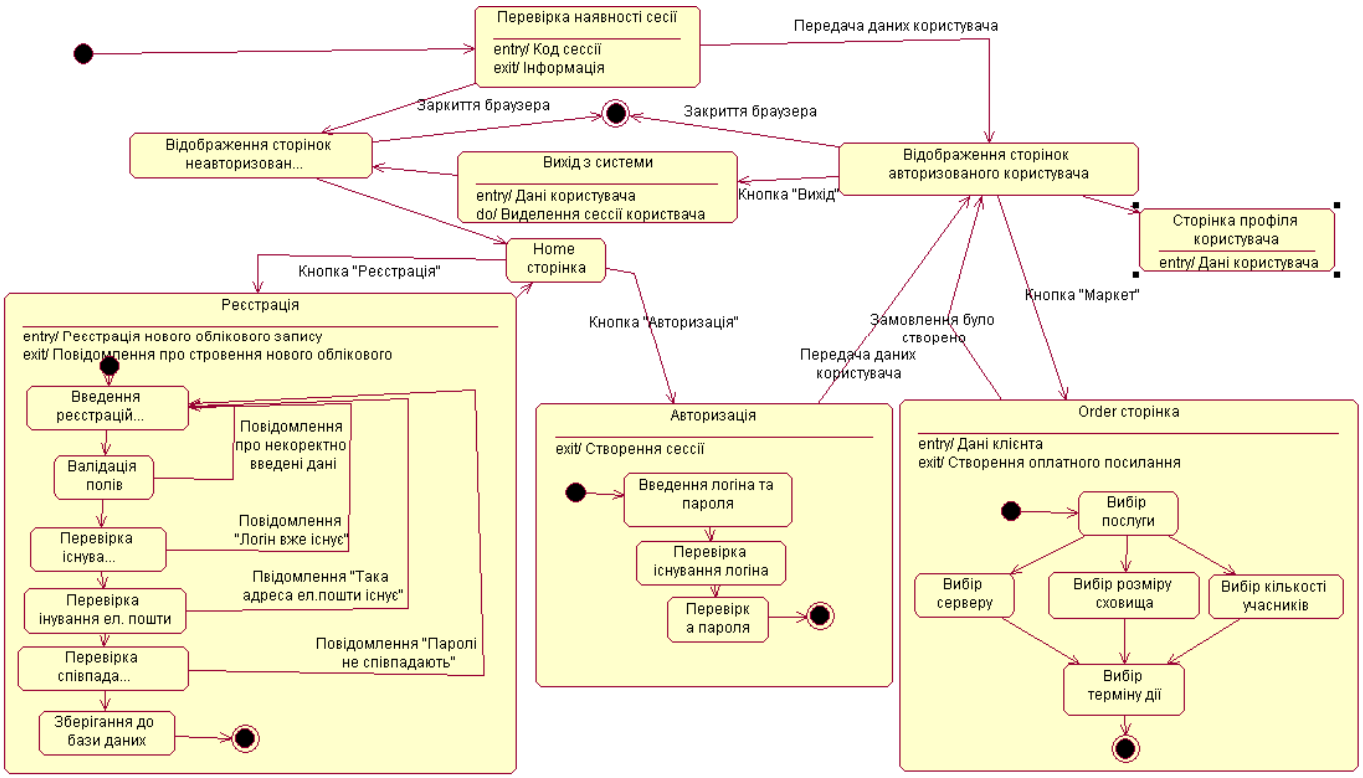

Рисунок 3.7 – Діаграма станів системи

Діаграма діяльності є блок-схемою, що представляє потік від однієї діяльності до іншої. Діяльність можна описати як роботу системи. Основне призначення діаграм діяльності полягає у фіксації динамічної поведінки системи. Операції переходу в наступний стан виконується, якщо поточної операції в попередньому стані була завершена.[22]

Для того, щоб зобразити діяльність системи та показати послідовність переходів між станами була побудована діаграма діяльності системи.(Рисунок 3.8)

Початок роботи системи - перевірка сесії у користувача та надається доступ до певних станів системи.

Операції «Авторизація» та «Вихід з системи» зображують різницю між акторами. В залежності до актора змінюється функціонал та права доступу .

Перехід від операцій відображень виконується лише коли користувач виконує ту або іншу дію.

Вихід з даної системи виконується закриттям браузера.

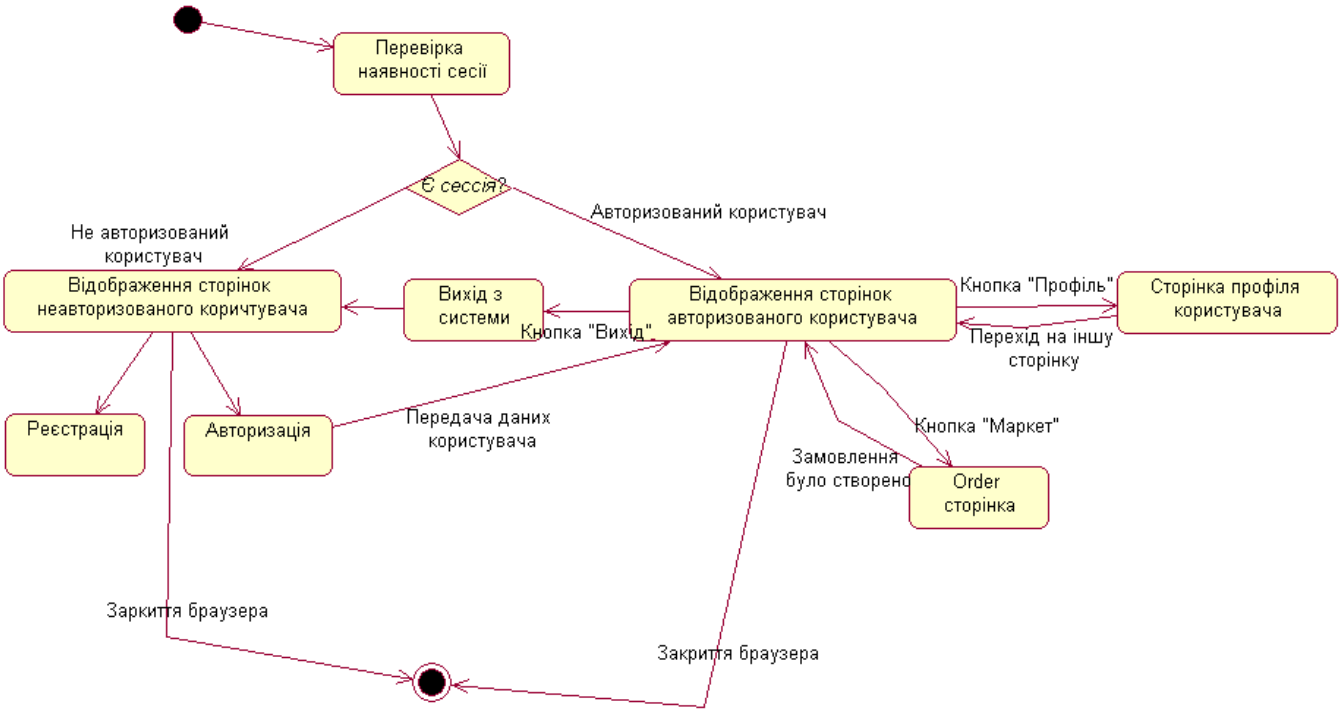

Рисунок 3.8– Діаграма станів системи

3.2 Вибір мови програмування та обґрунтування вибору СУБД

Для розробки системи обліку хмарного сервісу зоглядаються наступні програмні інструменти: мова програмування Python та фреймворк Django.

Python - це мова кодування загального призначення - це означає, що, на відміну від HTML, CSS та JavaScript, він може використовуватися для інших типів програмування та розробки програмного забезпечення.

Це, серед іншого, включає back-end розробку, розробку програмного забезпечення, науку даних та написання системних сценаріїв.

Серед інших переваг, Python відрізняється простотою, легкість для читання і простим синтаксисом; він легко інтегрується з зовнішніми компонентами, написаними на інших мовах програмування; має багатопарадигмену архітектуру і підтримує об'єктне, функціональне і модульне програмування; володіє великою колекцією вже запрограмованих інтерфейсів і утиліт. Набір вбудованих інструментальних засобів робить його надзвичайно гнучким і динамічним мовою програмування, що ідеально підходять не тільки для швидкого вирішення тактичних завдань, а й для розробки перспективних стратегічних рішень. Незважаючи на своє загальне призначення, Python часто називають мовою сценаріїв, так як він дозволяє легко і просто використовувати інші програмні компоненти та керувати ними.[23]

Django - є високорівневою веб-структурою Python, яка дозволяє швидко розвивати безпечні та ремонтопридатні веб-сайти. Побудований досвідченими розробниками, Django піклується про більшу частину клопоту веб-розробки, тому ви можете зосередитись на написанні свого додатку, не вимагаючи винаходити колесо. Це безкоштовне та відкрите джерело, має процвітаючу та активну спільноту, чудову документацію та безліч варіантів безкоштовної та платної підтримки. Перевагами цього фреймворку є:

– безпека. Захист від помилок, пов'язаних з безпекою і ставлять під загрозу проект. Мається на увазі такі поширені помилки, як ін'єкції SQL, крос-сайт підробки, clickjacking і крос-сайтовий скриптинг.

– масштабованість. Фреймворк Django найкращим чином підходить для роботи з найвищими трафіками. Велика кількість завантажених сайтів використовують Django для задоволення вимог, пов'язаних з трафіком.

– різнобічність. Менеджмент контенту, наукові обчислювальні платформи, навіть великі організації - з усім цим ефективно справляється Django.[24]

У якості бази даних зоглядається MySQL .

MySQL - вільна система управління базами даних.

Переваги MySQL: працює на различних платформах, має підтримку декількох одночасно запитів, містить записи фіксованої і змінної довжини, має гнучку систему привілеїв і паролів, володіє легкістю управління таблицею, включаючи додавання та видалення ключів і полів [25].

Недоліки та обмеження MySQL: проблеми з надійністю через деяких способів обробки даних MySQL (зв'язку, транзакції, аудити) іноді поступається іншим СУБД по надійності.

Сучасна СУБД Oracle - це потужний програмний комплекс, що дозволяє створювати додатки будь-якого ступеня складності. Центром цього комплексу є база даних, що зберігає інформацію, кількість якої практично безмежна за рахунок вбудованих засобів масштабування [26].

Використовувати MyISAM краще в тих таблицях, в яких переважає один тип доступу. Використання MyISAM виправдане у випадку коли переважає лише зчитування або запис. MyISAM має сенс використовувати, коли переважають операції insert або select, але вкрай мало delete або update. Також одночасні запити до різних частин таблиці виконуються повільніше.

InnoDB являється транзакційним типом, оскільки підтримує відкат до попереднього стану, якщо хоча б один із запитів не виконується. Даний тип таблиць

підтримує зовнішні ключі, до яких можна застосувати дії при певних обставинах (ON UPDATE, ON DELETE, …).

Але має свої недоліки такі як:

а) повільніше виконуються insert операції;

б) старі версії не підтримують повнотекстовий пошук;

в) поблеми з продуктивністю COUNT (*).

Співставивши всі аргументи, можна зробити висновок, що для даної предметної області кращим вибором буде використання InnoDB.

3.3 Розробка програмного забезпечення

Так як система передбачає наявність бази даних, необхідно спроектувати модель даних. Для створення даної моделі використовується програмний засіб All Fusion ErWin Data Modeler.

All Fusion ErWin Data Modeler є інструментом моделювання даних для візуалізації метаданих та схеми баз даних для розуміння складних джерел даних та дизайну.

Модель даних – це сукупність взаємопов'язаних структур даних та проведення операцій над цими структурами. В залежності від вигляду моделі і типу структур – знаходяться методи організації і обробки даних.[27]

Моделі даних erwin зменшують складність, полегшуючи розробку, розгортання та розуміння джерел даних для задоволення потреб бізнесу. erwin DM також автоматизує та стандартизує завдання проектування моделей, включаючи складні запити, для поліпшення вирівнювання бізнесу, забезпечення цілісності даних та спрощення інтеграції. Erwin має наступні переваги:

– можливість переглядати структуровані або неструктуровані корпоративні дані незалежно від їх розташування - у реляційній базі даних або базі даних NoSQL, сховищі даних, будь то локально або в хмарі, в межах одного інтерфейсу.

– дозволяє автоматично генерувати моделі даних та конструкції баз даних для підвищення ефективності та зменшення помилок.

– дає інтегрований погляд на концептуальні, логічні та фізичні моделі даних допомагає діловим та технічним зацікавленим сторонам зрозуміти структури та значення даних.

Для системи обліку хмарного сервісу спроектована логічна модель бази даних, яка представлена на рисунку 3.9.

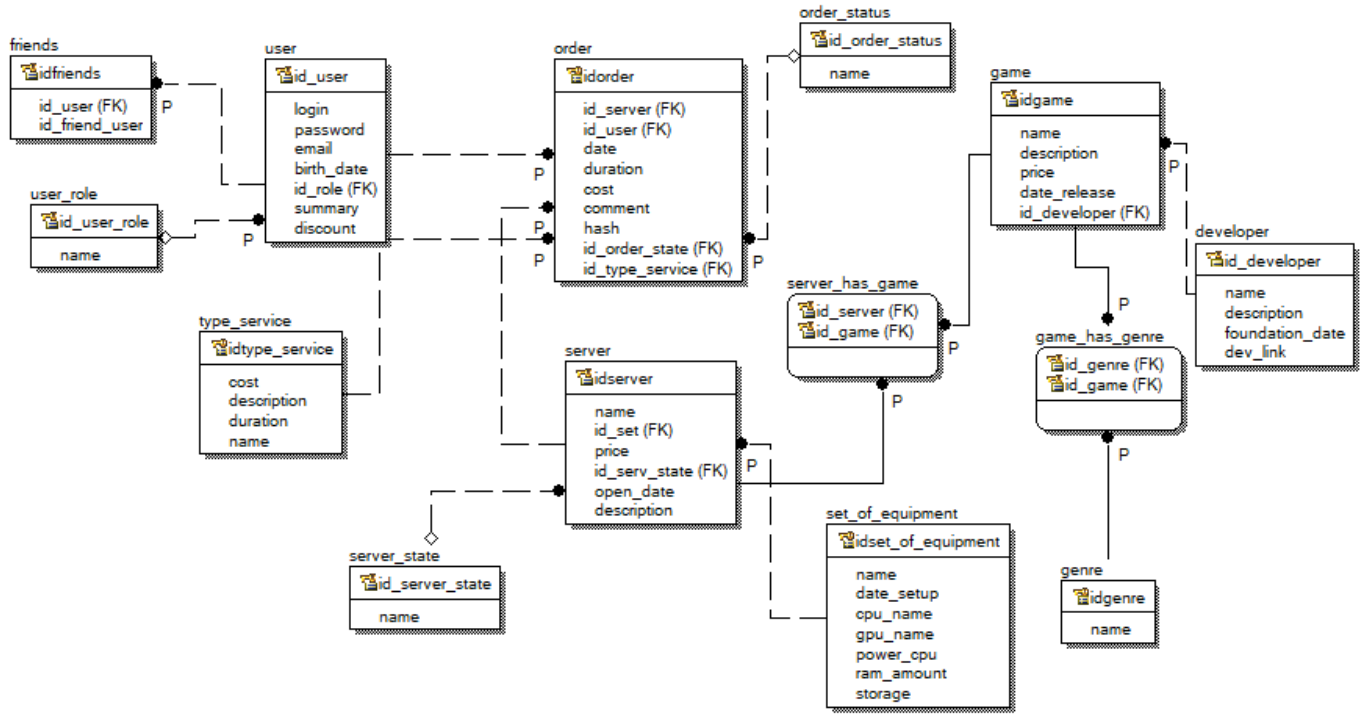

Рисунок 3.9 Логічна модель бази даних

Логічна модель бази даних системи обліку хмарного сервісу та має такі сутності:

– User – користувачах;
– Order – замовлення;
– Server – сервер;
– Set_of_equipment – обладнання;
– Game – ігровий додаток;
– Server_has_game – сутність, яка пов’язує сутності Server та Game;
– User_role – роль користувача;
– Order_status – стан замовлення;
– Server_state – стан серверу;
– Developer – розробник ігрового додатку;
– Genre – жанр ігрового додатку;
– Game_has_genre – сутність, яка пов’язує сутності Game та Genre;
– Friend – сутність, яка відповідає за список друзів;
– Type_service – інформація про послуги.

Для створення фізичної моделі бази даних необхідно визначити тип даних кожного атрибуту. Перелік сутностей, атрибутів та їх типів наведений у таблиці 3.1.

Були застосовані такі типи даних:

– числовий;

– текст;

– дата;

– число з рухомою комою.

Таблиця 3.1 – Перелік атрибутів та типів даних кожної сутності.

№	Найменування сутності	Найменування атрибута	Призначення	Тип даних (домен)
1	Friend	idfriend	Первинний ключ	Лічильник, цілочисловий
		id_user	Зовнішній ключ	Цілочисловий
		id_friend_user	Зовнішній ключ	Цілочисловий
2	Developer	id_developer	Первинний ключ	Лічильник, цілочисловий
		name	Назва компанії розробника ігор	Текст, довжиною в 45 символів
		description	Опис компанії розробника ігор	Текст, довжиною в 2000 символів
		foundation_date	Дата заснування	Дата
		dev_link	Посилання на сайти розробника	Текст, довжиною в 100 символів
3	Game	idgame	Первинний ключ	Лічильник, цілочисловий
		name	Назва ігрового додатку	Текст, довжиною в 45 символів
		description	Опис ігрового додатка	Текст, довжиною в 2000 символів
		price	Ціновий коефіцієнт	Число з рухомою комою
		date_release	Дата випуску гри	Дата
		id_developer	Зовнішній ключ	Цілочисловий
4	GameHasGenre	id_genre	Складовий ключ	Цілочисловий
		id_game	Складовий ключ	Цілочисловий
5	Genre	idgenre	Первинний ключ	Лічильник, цілочисловий
		name	Назва жанру	Текст, довжиною в 45 символів

Продовження таблиці 3.1

№	Найменування сутності	Найменування атрибута	Призначення	Тип даних (домен)
6	Order	idorder	Первинний ключ	Лічильник, цілочисловий
		id_server	Зовнішній ключ	Цілочисловий
		id_user	Зовнішній ключ	Цілочисловий
		date	Дата замовлення	Дата
		duration	Термін дії	Число з рухомою комою
		cost	Вартість замовлення	Число з рухомою комою
		comment	Коментар клієнта	Текст, довжиною в 1000 символів
		hash	Ключ транзакції	Текст, довжиною в 45 символів
		id_order_state	Зовнішній ключ	Цілочисловий
		id_type_service	Зовнішній ключ	Цілочисловий
7	OrderStatus	id_order_status	Первинний ключ	Лічильник, цілочисловий
		name	Назва статусу замовлення	Текст, довжиною в 45 символів
8	Server	idserver	Первинний ключ	Лічильник, цілочисловий
		name	Назва серверу	Текст, довжиною в 45 символів
		id_set	Зовнішній ключ	Цілочисловий
		price	Ціновий коєфіцієнт	Число з рухомою комою
		id_serv_state	Зовнішній ключ	Цілочисловий
		open_date	Дата встановлення	Дата
		description	Опис серверу	Текст, довжиною в 1000 символів
9	ServerHasGame	id_server	Складовий ключ	Цілочисловий
		id_game	Складовий ключ	Цілочисловий
10	ServerState	id_server_state	Первинний ключ	Лічильник, цілочисловий
		name	Назва статусу сервера	Текст, довжиною в 45 символів
11	SetOfEquipment	idset_of_equipment	Первинний ключ	Лічильник, цілочисловий
		name	Назва обладнання	Текст, довжиною в 45 символів
		date_setup	Дата збору	Дата
		cpu_name	Назва процесора	Текст, довжиною в 45 символів

Продовження таблиці 3.1

№	Найменування сутності	Найменування атрибута	Призначення	Тип даних (домен)
11	SetOfEquipment	gpu_name	Назва графічного процесора	Текст, довжиною в 45 символів
		power_cpu	Потужність процесора	Число з рухомою комою
		ram_amount	Кількість оперативної пам’яті	Цілочисловий
		storage	Кількість та тип постійної пам’яті	Текст, довжиною в 10 символів
12	TypeService	idtype_service	Первинний ключ	Лічильник, цілочисловий
		description	Опис послуги	Текст, довжиною в 500 символів
		duration	Тривалість послуги	Число з рухомою комою
		name	Назва послуги	Текст, довжиною в 45 символів
13	User	id_user	Первинний ключ	Лічильник, цілочисловий
		login	Логін користувача	Текст, довжиною в 45 символів
		password	Пароль користувача	Текст, довжиною в 45 символів
		email	Адреса електронної пошти	Текст, довжиною в 45 символів
		birth_date	Дата народження	Дата
		id_role	Зовнішній ключ	Цілочисловий
		summary	Опис користувача	Текст, довжиною в 2000 символів
		discount	Показник знижки	Число з рухомою комою
14	UserRole	id_user_role	Первинний ключ	Лічильник, цілочисловий
		name	Назва ролі	Текст, довжиною в 45 символів

Так як була складена логічної моделі даних та обрана платформа СУБД MySQL, то можна скласти фізичну модель бази даних хмарного сервісу ігрових додатків. Така модель представлена на рисунку 3.10.

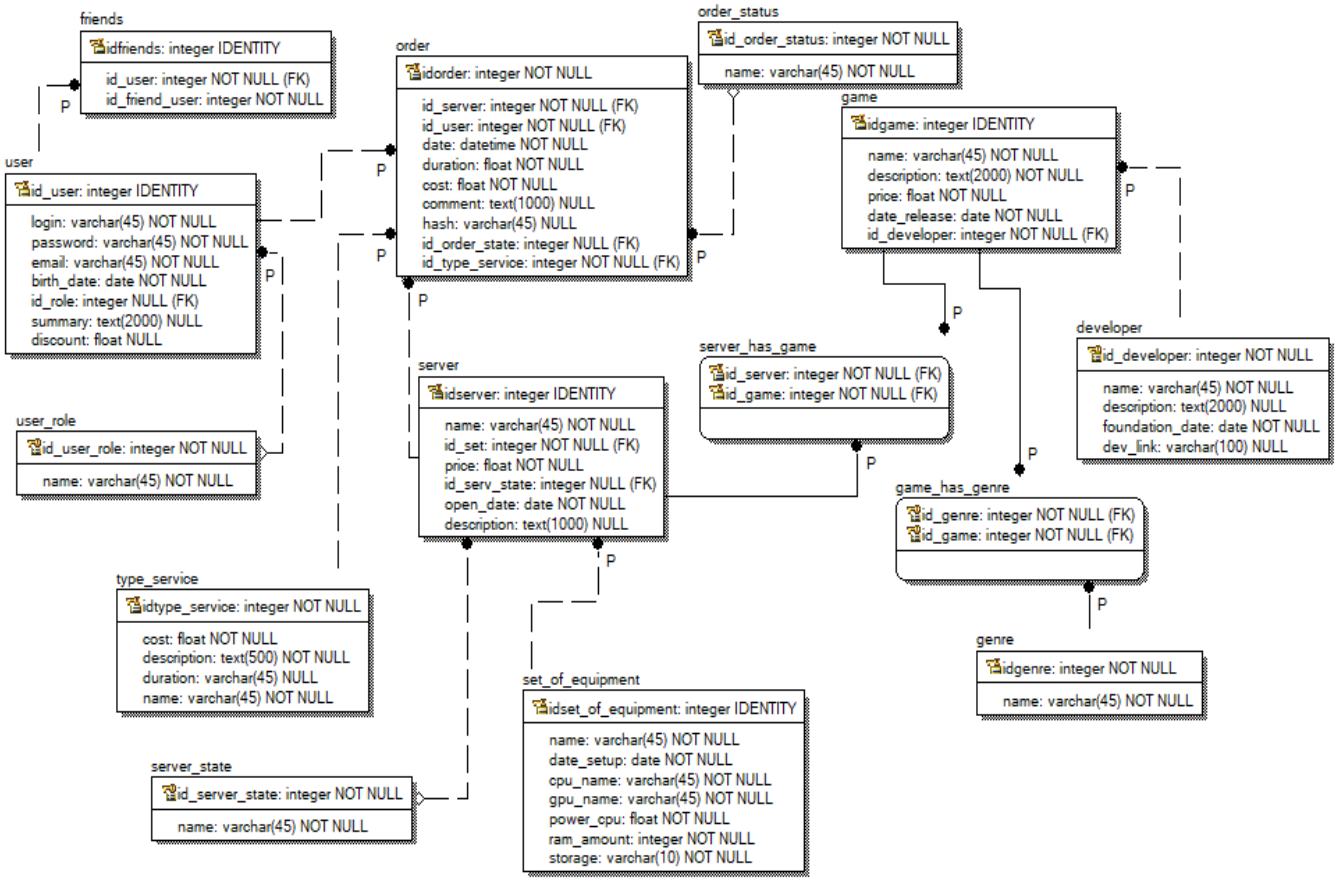

Рисунок 3.10 Фізична модель бази даних

Для розробки серверної частини повинна бути створена база даних. Для створення бази даних використовується засіб візуального проектування та створення бази даних СУБД MySQL під назвою MySQL Workbench.

Створена база даних для системи обліку хмарного сервісу ігрових додатків зображена на рисунку 3.3 за допомогою засобу MySQL Workbench.

При розробці бази даних був обраний механізм роботи таблиць бази даних InnoDB. Це означає що усі таблиці зберігаються у одному файлі бази даних. Також цей механізм забезпечує підтримку зовнішніх ключів та транзакцій.[28]

Так як база даних має певну кількість зв’язків між таблицями, то повинна бути реалізована посилкова цілісність даних. Таблиці InnoDB надають можливість реалізувати такі налаштування зв’язків.

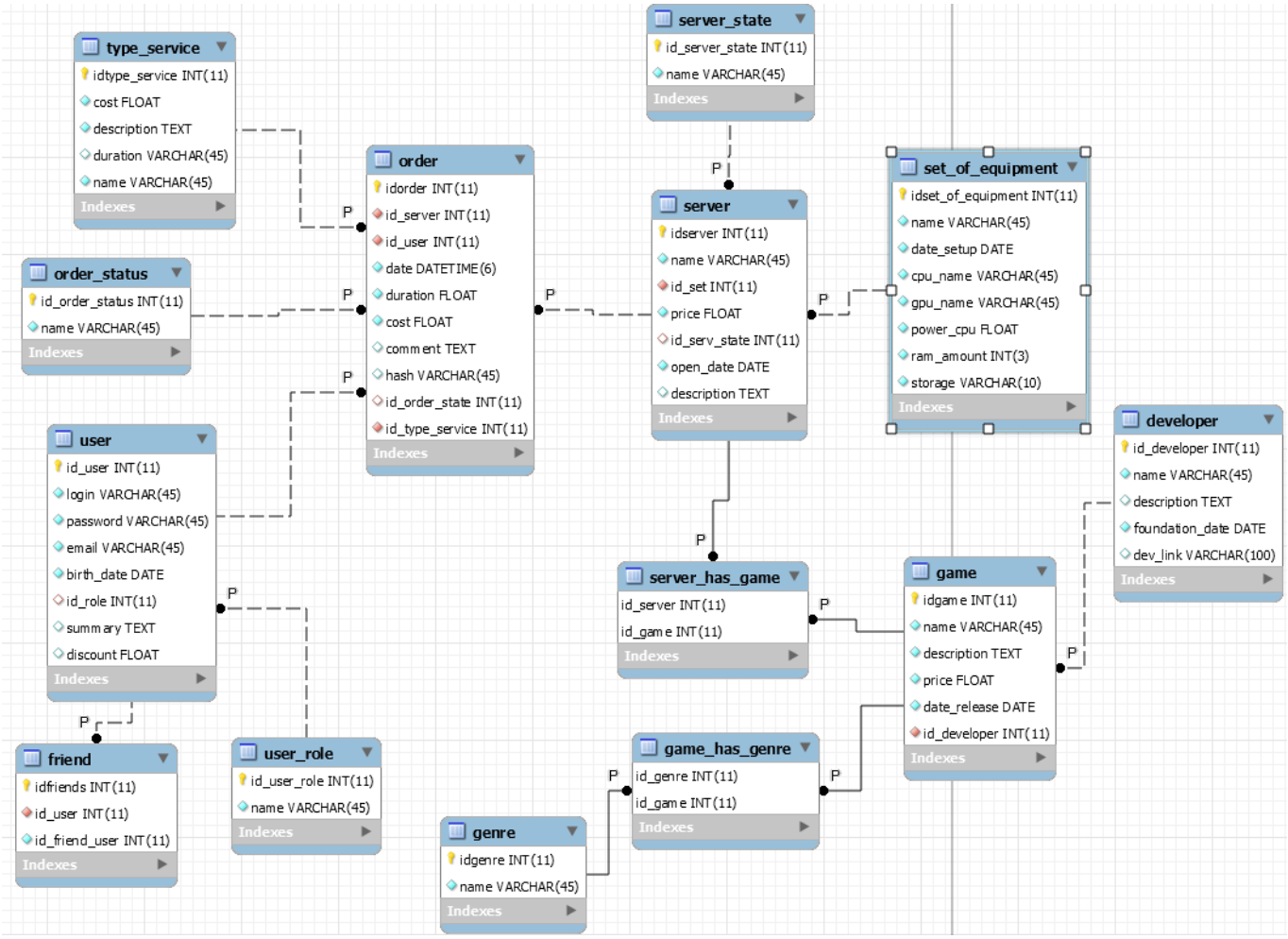

Рисунок 3.11 Схема бази даних MySQL

Умови цілісності даних визначають, які дані можуть бути записані в БД у результаті додавання або оновлення даних. При маніпулюванні даними в таблицях виконується контроль дій, який встановлює одне з правил для таких операція, як Оновлення(UPDATE) та Видалення(DELETE).

У стандарті SQL визначені такі дії підтримки цілісності даних:

– cascade. Видалення або оновлення запису призводить до видалення та оновлення записів з відповідним значенням зовнішнього ключа.

– restrict. Заборонити видалення або оновлення запису, якщо є залежні записи з відповідним значенням зовнішнього ключа.

– set null. Видалення або оновлення запису призводить до встановлення у зовнішні ключі невизначеного значення.

– set default. Вимагає, щоб при видаленні запису у зовнішній ключ встановлювалося типове значення.[29]

Для створеної бази даних встановлено такі правила для зв’язків, які наведені у таблиці 3.11

Таблиця 3.2 – Таблиця посилкової цілісності

№	Таблиця 1, зовнішній ключ	Таблиця 2, первинний ключ	Операція	Тип посилкової цілісності
1	user, id_role	user_role, id_user_role	UPDATE	CASCADE
			DELETE	SET NULL
2	server_has_game, id_server	server, id server	UPDATE	CASCADE
			DELETE	CASCADE
3	server, id_serv_state	server_state, id_server_state	UPDATE	CASCADE
			DELETE	SET NULL
4	order, id_order_state	order_state, id_order_status	UPDATE	CASCADE
			DELETE	SET NULL
5	game, id_developer	developer, id_developer	UPDATE	CASCADE
			DELETE	RESTRICT
6	server, id_set	set_of_equipment, idset_of_equipment	UPDATE	CASCADE
			DELETE	RESTRICT
7	order, id_server	server, idserver	UPDATE	CASCADE
			DELETE	RESTRICT
8	order, id_user	user, id_user	UPDATE	CASCADE
			DELETE	CASCADE
9	server_has_game, id_game	game, idgame	UPDATE	CASCADE
			DELETE	CASCADE
10	game_has_genre, id_genre	genre, idgenre	UPDATE	CASCADE
			DELETE	CASCADE
11	game_has_genre, id_game	game, idgame	UPDATE	CASCADE
			DELETE	CASCADE
12	order, id_type_service	type_service, idtype_service	UPDATE	CASCADE
			DELETE	RESTRICT
13	friend, id_user	user, iduser	UPDATE	CASCADE
			DELETE	CASCADE

3.4 Розробка алгоритму роботи системи

Алгоритм роботи системи обліку хмарного сервісу побудований на взаємодії трьох складових: моделі даних, функцій контроллера та представлення інтерфейсу користувача.

Алгоритм роботи системи зображено на рисунку 3.12.

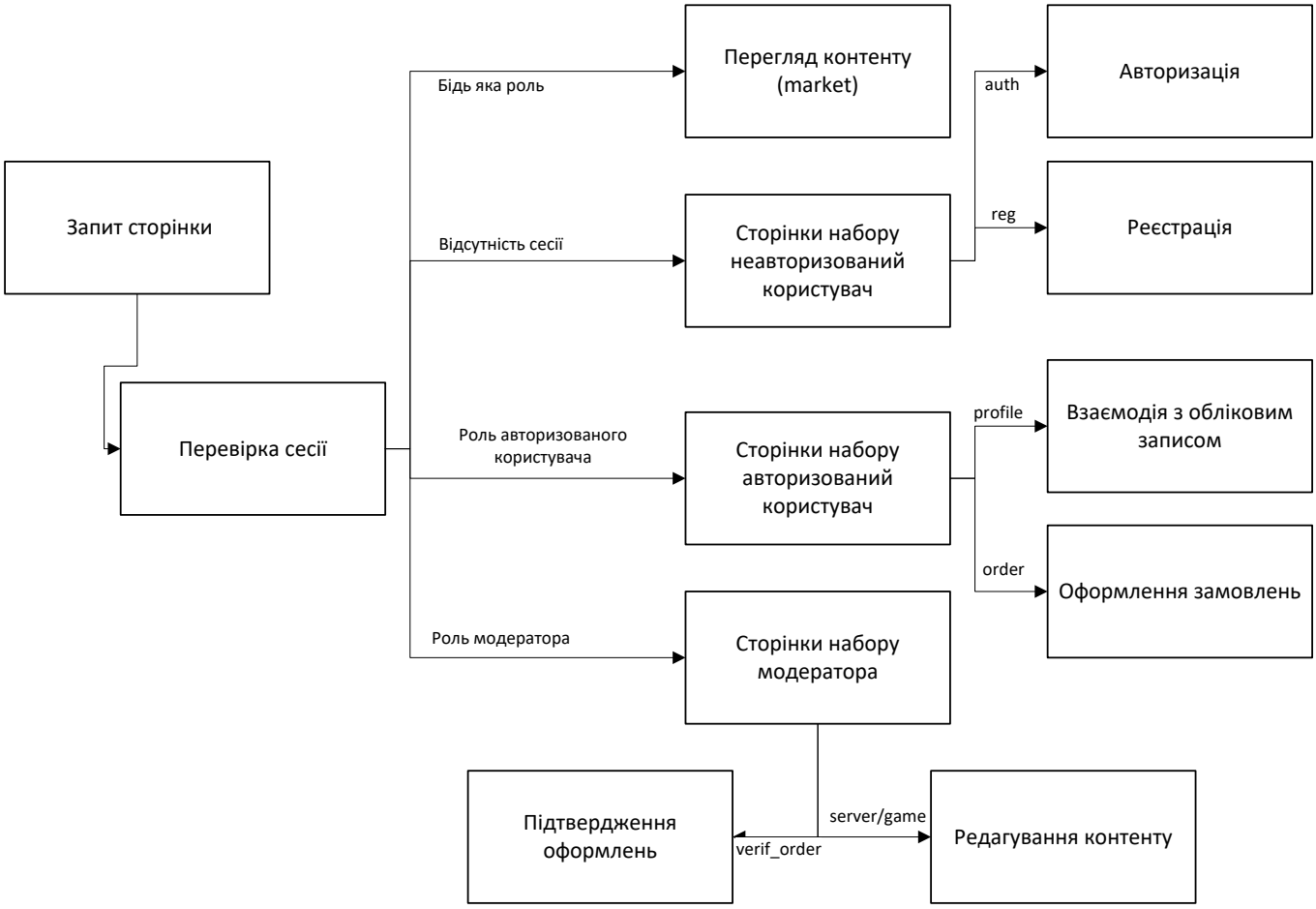

Рисунок 3.12 – Алгоритм роботи системи та надання доступу до сторінок

VC (Model-View-Controller) - це шаблон архітектурного дизайну, який заохочує вдосконалення організації додатків шляхом розділення проблем. Він ділить інтерактивний додаток на три компоненти: Model / View та Controller. Він забезпечує ізоляцію бізнес-даних (Моделі) від користувальницьких інтерфейсів (Проекції), а третій компонент (Контролери) традиційно управляє логікою, вводом користувача та координує як моделі, так і подання. Мета MVC - допомогти структурувати окремі проблеми програми на три частини:

– Model відповідає за управління даними програми. Він отримує вхід користувача від контролера.

– view означає презентацію моделі в певному форматі.

– controller реагує на введення користувачем і виконує взаємодію з об'єктами моделі даних. Контролер отримує вхідні дані, додатково перевіряє їх, а потім передає вхідні дані моделі.

MVC допомагає відокремити різні аспекти програми (вхідна логіка, бізнес-логіка та графічний інтерфейс), одночасно забезпечуючи вільне зв’язок між цими елементами. Таким чином, інформаційна логіка належить до моделі, графічний інтерфейс належить до подання. Вхідна логіка належить контролеру. Це розділення допомагає управляти складністю під час створення додатка, оскільки воно дозволяє зосередитись на одному аспекті реалізації за раз. MVC Framework має переваги:

– Одночасна розробка - оскільки MVC роз’єднує різні компоненти програми, розробники можуть паралельно працювати над різними компонентами, не впливаючи та не блокуючи один одного.

– багаторазове використання - Один і той же (або подібний) вигляд для однієї програми може бути реконструйований для іншої програми з різними даними, оскільки подання просто обробляє те, як дані відображаються користувачеві.

– покращена масштабованість - якщо у вашої програми виникають проблеми з продуктивністю, оскільки доступ до бази даних повільний, ви можете оновити апаратне забезпечення, на якому запущена база даних, без впливу інших компонентів

– низьке зчеплення - сама природа фреймворку MVC така, що між моделями, видами або контролерами існує низький зв’язок.

– краща розширюваність - оскільки компоненти мають низьку залежність один від одного, внесення змін до одного (для виправлення помилок або зміни функціональності) не впливає на інші/

На основі цих механізмів реалізована система обліку хмарного сервісу, а саме для кожна функція відповідає реалізації певної кількості функціональних компонентів.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи виконано аналіз існуючих хмарних сервісів. Визначено загальну проблему в організаційній структурі хмарного сервісів і запропоновано спроектовані моделі та надані рекомендації для автоматизації роботи систем обліку на основі практики існуючих досліджень і проведених дослідів.

Проаналізовано етапи основного бізнес-процесу організації з надання послуг хмарного сервісу.

Визначено сутність системного підходу, проаналізовано його переваги та недоліки.

Застосовано системний підхід до визначення системи обліку хмарного сервісу Використовуючи запропонований метод було спроектовано моделі, згідно до яких, можна оцінити та виявити основні неавтоматизовані елементи системи.

Для розробки автоматизованих компонентів заснована модель MVC, яка відповідає усім потребам даної предметної області.

Для побудови відповідних моделей системи хмарного сервісу та визначення функціональних вимог була використана методологія IDEF0.

Під час проектування автоматизованої системи обліку хмарного сервісу були розроблені моделі поведінки системи та послідовності взаємодій.

Для моделювання інформаційних потоків між підсистемами та зовнішніми сутностями системи обліку хмарного сервісу використовувалась DFD-діаграма та UML-діаграма варіантів використання.

Під час практичної реалізації був проведений обґрунтований аналіз вибору мов програмування та СУБД. Компоненти системи були розроблені на мові програмування Python з використанням фреймворку Django.

ПЕРЕЛІК ПОСИЛАНЬ

1. Хмарний сервіс це - Хмарні сервіси: різновиди і переваги [Електронний ресурс] – Режим доступу до ресурсу: https://iapple-59.ru/raznoe/oblachnyj-servis-eto-oblachnye-servisy-raznovidnosti-i-preimushhestva-2.html

2. Fernando Doglio, "Content as a Service: Your Guide to the What,Why, and How" ButterCMS, May 7, 2019[Електронний ресурс] – Режим доступу до ресурсу: https://buttercms.com/blog/content-as-a-service-your-guide-to-the-what-why-and-how.

3. Olson, John A. "Data as a Service: Are We in the Clouds?". Journal of Map &Geography Libraries. 6 - 2015 - P 76–78.

4. Desktop virtualization [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Desktop_virtualization #Desktop_as_a_service.

5. Antony Ananich, "What is IaaS?", Mar2, 2016 [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org /web/20160302153830/http://ananich.pro/2016/02/what-is-iaas/.

6. Donovan Jones, "Blackstone Acquires Cloudreach For Access To iPaaS Market", February 21, 2017 [Електронний ресурс] – Режим доступу до ресурсу: https://seekingalpha.com/article/4048008-blackstone-acquires-cloudreach-for-access-to-ipaas-market.

7. Zachary Flower, "Weigh the benefits of PaaS providers against lock-in risks", May 29, 2018. [Електронний ресурс] – Режим доступу до ресурсу: https://searchcloudcomputing.techtarget.com/feature/Weigh-the-benefits-of-PaaS-providers-against-lock-in-risks.

8. Softwareon-demand, Platform as a service, Infrastructure as a service, Google Apps Education Edition — Режим доступу: http://www.google.com/a/help/intl/en/edu/index.html

9. Mishcheriakov, I. A rough set based algebraic approach to modelling complex systems / Sitnikov, D., Ryabov, O., Mishcheriakov, I., Kovalenko, A. // International Journal of Design and Nature and codynamics. – 2018. – Vol. 13/ – P. 324–329

10. Лімонова Л.О. Системний підхід як методологічна основа досліджень аналізу та моделювання соціально економічних систем. 2010 // URL:https://dl.nure.ua/pluginfile.php/1168/mod_resource/content/1/01.pdf

11. Кустовська О. В. Методологія системного підходу та наукових досліджень: Курслекцій. – Тернопіль: Економічна думка, 2005. – 124 с.

12. M. Vyshniak, I. Klymova The basic issues of knowledge management bionics. // Bionics of intelligence, 2018, № 1 (90), pp. 24-30

13. Муненченко П.Є. Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Конференція «Інформаційні інтелектуальні системи» : Міжнар. молодіжний форум Харків, 2021: Том 6 С. 202-203.

14. Похилько А. Ф. CASE-технология моделирования процессов с использованием средств BPWin и ERWin учебное пособие / А. Ф. Похилько, И. В. Горбачев. – Ульяновск: УлГТУ, 2008. – 120 с.

15. Калянов Г. Н. CASE структурный системный анализ : (Автоматизация и применение) / Г. Н. Калянов. - М. : ЛОРИ, Б. г. (1996). - 242 с.

16. Боггс У., Боггс М. UML и Rational Rose Пер. с англ. И. Афанасьева / У Боггс, М. Боггс — М.: Лори, 2001. — 580 с.

17. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд.: Пер. с англ. / Дж. Рамбо, М. Блаха – СПб.: Питер, 2007. – с. 544.

18. Кватрани. Т. . Rational Rose 2000 и UML. Визуальное моделирование / Т. Кватрани. – Москва: ДМК Пресс., 2000. – 176 с.

19. Кендалл С., UML. Основные концепции / Скотт Кендалл - М.: Вильямс, 2002. - с. 140.

20. Диаграмма состояний [Електронний ресурс] – Режим доступу до ресурсу: https://books.ifmo.ru/file/pdf/424.pdf.

21. Диаграмма деятельности (действий) [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.net/1_37174_diagramma-deyatelnosti-deystviy.html.

22. Лутц М. Программирование на Python, том I, 4-е издание: Пер. с англ. / М. Лутц. – СПб.: Символ-Плюс, 2011. – 992 с.

23. Документация Django 1.4 [Електронний ресурс] – Режим доступу до ресурсу: https://djbook.ru/.

24. 4. Шварц Б.А. MySQL. Оптимизация производительности/ Б.А. Шварц, П.Н. Зайцев, В.Т. Ткаченко. – М.: Наука, 2010. – 412 с.

25. Федоров А.Н. Введение в базы данных. – Ч. 2: Настольные СУБД/ А.Н. Федоров, Н.С. Елманова// М.: Комп'ютер-пресс. – 2000. – с. 127.

26. Design Patterns MVC Pattern [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm

27. Дубейковский В. И. Эффективное моделирование с CA ERwin Process Modeler (BPwin; AllFusion Process Modeler) / В. И. Дубейковский. – 2-е изд., испр. и доп. – М. : ДИАЛОГ-МИФИ, 2009. – 384 с.

28. Шварц Б.А. MySQL. Оптимизация производительности/ Б.А. Шварц, П.Н. Зайцев, В.Т. Ткаченко. – М.: Наука, 2010. – 412 с.

29. Ссылочная целостность баз данных. [Електронний ресурс] – Режим доступу до ресурсу: https://studfiles.net/preview/3619027/page:16.