

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЕКТ

на тему

**РОЗРОБКА ВЕБ-ПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ ОНЛАЙН-
КІНОТЕАТРУ**

здобувача Барабаша Олександра Михайловича

Науковий керівник:

к.е.н., доцент

_____ Лозовик Ю. М.

**Проект допущений до захисту перед
екзаменаційною комісією з атестації
здобувачів вищої освіти**

в.о. завідувача кафедри:

к.е.н., доцент.

_____ Тішков Б.О.

КИЇВ 2024

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці
ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО: Керівник проектної групи(гарант) освітньо-професійної програми _____ Іванченко Г.Ф. “_____” _____ 2024 р.	ЗАТВЕРДЖУЮ: в.о. завідувача кафедри _____ Тішков Б.О. “_____” _____ 2024 р.
--	--

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти ***Барабаша Олександра Михайловича***
очної (денної) форми навчання
на підготовку кваліфікаційного бакалаврського проекту
на тему: «Розробка веб-платформи для організації роботи онлайн-кінотеатру»

Тему затверджено наказом ректора Університету від «11» березня 2024 р. №
529-ст.

Кваліфікаційний бакалаврський проект виконується на матеріалах
дослідження предметної області джерела

План кваліфікаційного бакалаврського проекту

Розділ I. Характеристика та аналіз предметної галузі

Розділ II. Розробка вимог і моделювання інформаційної системи

Розділ III. Проектування та реалізація компонентів системи

Об'єкт дослідження. Процеси управління медіаконтентом та забезпечення зручного обміну даними між користувачами онлайн-кінотеатру

Предмет дослідження. Проектування та розробка веб-платформи для онлайн-кінотеатру, яка включає збереження фільмів, серіалів, їх перегляд, а також можливість обміну цими даними між користувачами

Мета кваліфікаційного бакалаврського проекту. Розробка та реалізація веб-платформи, яка надає користувачам зручні та ефективні інструменти для організації перегляду медіаконтенту онлайн

**Конкретні завдання, які студент повинен виконати для досягнення
поставленої мети:**

У розділі I. Надати характеристику предметної області та об'єкта дослідження.
Проаналізувати практичний досвід використання ІС в предметної галузі.

У розділі II. Провести аналіз та розробити специфікацію вимог до системи.
Описати постановку задачі та алгоритми функціонування системи. Здійснити
моделювання системи.

У розділі III. Спроектувати та реалізувати компоненти системи.

Завдання підготував

науковий керівник _____

Лозовик Юрій Миколайович

“ _____ ” _____ 2024 р.

Завдання одержав

студент _____

Барабаш Олександр Михайлович

Відгук

наукового керівника про кваліфікаційний бакалаврський проект
студента Навчально-наукового інституту
“Інститут інформаційних технологій в економіці”
галузі знань 12 “Інформаційні технології”
спеціальності 122 “Комп’ютерні науки”
Барабаша Олександра Михайловича

на тему: «**Розробка веб-платформи для організації роботи онлайн-кінотеатру**»

- 1. Актуальність теми:** питання розробки веб-платформи для організації пошуку та перегляду цікавого контенту за акторами, жанрами та назвами фільмів є важливим для кіноманів. Тема роботи є актуальною для кожного, хто має бажання і час переглядати кінофільми у зручному форматі та якості. Для вирішення цього питання автор роботи, Барабаш Олександр Михайлович, проектує і розробляє новий варіант веб-застосунку для організації онлайн-кінотеатру.
- 2. Позитивні риси кваліфікаційного бакалаврського проекту:** автор роботи, Барабаш Олександр Михайлович, здійснив проектування та розробку веб-платформи з використанням сучасного стеку технологій, проаналізував аналоги та визначив напрями подальшого удосконалення продукту.
- 3. Наявність самостійних розробок автора:** автор реалізує велику кількість структурних і поведінкових діаграм, розробляє прототип інтерфейсу системи та реалізує проектні рішення у авторському продукті. Продукт – цілісний і реалізує визначений функціонал щодо пошуку та переглядів відео.
- 4. Цінність теоретичних висновків та практичних рекомендацій:** розроблений програмний продукт буде корисним для всіх, хто бажає переглядати відеоконтент.

5. Наявність недоліків:

- задовільний виклад матеріалу;
- робота більш практичного спрямування, ніж наукового;
- у списку використаних джерел відсутні наукові джерела, в переліку наведені здебільшого посилання на веб-сайти;
- відсутній огляд наукової літератури;
- інтерфейс та серверна частина програми потребують доопрацювання;
- поза увагою залишилось питання використання авторських прав щодо розміщення відео-контенту на розробленому сайті.

6. Загальна оцінка кваліфікаційного бакалаврського проекту та його допущення до захисту перед ЕК: вивисвітлене вище дозволяє констатувати, що дипломний проект відповідає встановленим вимогам, що висувуються до таких видів робіт, а його автор, Барабаш Олександр Михайлович, володіє навиками проектування та розробки веб-системи для організації онлайн-кінотеатру. Дипломна робота рекомендується до захисту в екзаменаційній комісії з атестації здобувачів першого (бакалаврського) рівня вищої освіти з позитивною оцінкою.

Науковий керівник

к.е.н., доцент

_____ Лозовик Юрій Миколайович

Рецензія
на кваліфікаційний бакалаврський проект
Барабаша Олександра Михайловича
тема

«Розробка веб-платформи для організації роботи онлайн-кінотеатру»

Актуальність теми кваліфікаційного бакалаврського проекту і доцільність його розроблення. У сучасному світі кіноіндустрії зростає популярність онлайн-кінотеатрів, адаптованих до потреб користувачів. Завдяки швидкому темпу життя та постійному доступу до Інтернету, існує великий попит на веб-платформи, які забезпечують зручний доступ до фільмів та організують їх перегляд. Розробка такої системи є необхідною для задоволення сучасних потреб у розвагах.

Якість проведеного дослідження. Автор проекту добре розуміє вимоги сучасної аудиторії та проблеми, з якими стикаються онлайн-кінотеатри. Він успішно дослідив можливі шляхи їх вирішення, базуючись на надійних академічних джерелах.

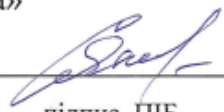
Позитивні риси кваліфікаційного бакалаврського проекту. Автор виконав значну роботу з аналізу потреб та вимог користувачів, що сприяло розробці функціональних можливостей системи, що найбільш ефективно відповідають їхнім потребам. Дизайн інтерфейсу також добре продуманий і забезпечує зручну навігацію та задоволення користувацького досвіду.

Зауваження. У проекті можна було більш детально описати процес розробки системи, зосереджуючись на виборі технологій і архітектурних рішень.

Практична значимість висновків і рекомендацій. Сформовані висновки чітко описують переваги та недоліки розробленої системи, а також можливі шляхи для її вдосконалення. Рекомендації щодо подальшого розвитку системи також є цінними для майбутніх досліджень у цій галузі.

Вважаю, що робота виконана на високому професійному рівні, має теоретичну та практичну значущість й заслуговує на оцінку «Відмінно»

ВСП «Фаховий коледж інформаційних
систем і технологій КНЕУ ім. В. Гетьмана»
завідуюча відділення «Програмування»

 Світлана ЯНЧУК
підпис, ППБ



Місце печатки організації, де працює рецензент

АНОТАЦІЯ

кваліфікаційного бакалаврського проекту студента 4 курсу
навчально-наукового інституту «Інститут інформаційних технологій в
економіці»

Барабаша Олександра Михайловича, виконаної на тему:
«Розробка веб-платформи для організації роботи онлайн-кінотеатру»
Київ: кафедра інформаційних систем в економіці, 2024 р.

Кваліфікаційний бакалаврський проект присвячений створенню веб-платформи для управління онлайн-кінотеатром з метою полегшення процесу його функціонування та забезпечення зручного доступу до кінематографічного контенту. Проект складається з трьох розділів, що логічно взаємодіють між собою.

У першому розділі надається аналіз сучасного стану ринку онлайн-кінотеатрів, обґрунтовується необхідність створення нової веб-платформи та проводиться порівняльний огляд існуючих рішень у даній галузі.

Другий розділ присвячений розробці методологій проектування та моделювання системи. В ньому проводиться аналіз вимог користувачів та опис функціональності майбутньої платформи. Також у цьому розділі розглядається структура бази даних та алгоритми її взаємодії з веб-інтерфейсом.

У третьому розділі детально описується технічне забезпечення платформи. Розглядається вибір технологічних рішень для реалізації серверної та клієнтської частин системи, а також надається інформація про інтеграцію зовнішніх сервісів та API.

У висновках проводиться оцінка виконаного проекту, вказуються переваги та недоліки розробленої веб-платформи, а також рекомендації з її подальшого вдосконалення та впровадження в реальне життя онлайн-кінотеатру для перегляду відео контенту.

РЕФЕРАТ

Кваліфікаційний бакалаврський проект містить 87 сторінок, 3 таблиці, 76 рисунків, список літератури з 32 найменувань, 4 додатки.

РОЗРОБКА ВЕБ-ПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ ОНЛАЙН-КІНОТЕАТРУ

Перелік ключових слів: *Наприклад* база даних (БД), система керування базами даних (СКБД), клієнт, інформаційна система, вихідна форма, атрибут, зв'язок запиту, класи сутностей, структурний зв'язок, репозиторії, малюнки, інформаційна система, інформаційна база, автоматизоване робоче місце.

Об'єктом розробки є веб-платформи для організації роботи онлайн-кінотеатру, що спеціалізований на перегляді фільмів онлайн.

Мета кваліфікаційного бакалаврського проекту полягає в розробці веб-платформи для організації роботи онлайн-кінотеатру, що надасть користувачам можливість переглядати куплений відео контент на сайті.

закріпленні вивченого матеріалу для його подальшої реалізації в майбутньому.

Методами розроблення кваліфікаційного бакалаврського проекту є розробка веб-платформи в середовищі розробки IntelliJ IDEA і використанні його відносно зручного функціоналу в виді додаткових функцій : навігатора БД і т. д.

Апаратура використана при проектуванні та розробці є персональним комп'ютером.

Результати досягнуті в процесі роботи (також їх новизна та ступінь впровадження) було реалізовано методи роботи з базами даних, а саме, додавання, редагування та видалення даних також було реалізовано відтворення відео роликів з прев'ю на веб-сторінках.

Одержані результати можуть бути використані (рекомендації щодо практичного застосування здобутих результатів, сфери використання) у подальшому вдосконаленні даного проекту або при створенні наступного проекту.

Рік виконання кваліфікаційного бакалаврського проекту: 2024 р.

Рік захисту кваліфікаційного бакалаврського проекту: 2024 р.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ	2
ВСТУП.....	3
РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ	4
1.1. Характеристика предметної галузі та об'єкта дослідження	4
1.2. Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі	5
1.2.1. Дослідження існуючих інформаційних систем для обраної предметної області.	6
1.2.2. Порівняння програмних систем	10
1.2.3. Позитивні характеристики існуючих систем, що їх слід відтворити у проектних рішеннях	11
1.2.4. Недоліки існуючих систем та напрями їх удосконалення	13
РОЗДІЛ 2. РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ	15
2.1. Аналіз і специфікація вимог до інформаційної підсистеми	15
2.1.1. Бізнес-вимоги до системи	15
2.1.2. Функціональні вимоги до системи	22
2.1.3. Нефункціональні вимоги до системи.....	23
2.2. Постановка задачі.....	25
2.2.1. Характеристика задачі	25
2.2.2. Вихідна інформація	27
2.2.3. Вхідна інформація.....	30
2.2.4. Математичне забезпечення та алгоритм функціонування системи.....	31
2.3. Моделювання інформаційної підсистеми	36
2.3.1. Моделювання поведінки системи.....	36
2.3.2. Моделювання структури системи	39
2.3.3. Розробка тест-кейсів.....	43
2.3.4. Трасування вимог до системи	45
2.4. Моделювання користувацького інтерфейсу	47
РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ПІДСИСТЕМИ	63
3.1. Інформаційне забезпечення	63
3.2. Технічне забезпечення.....	69
3.3. Програмне забезпечення.....	71
3.4. Результати реалізації інформаційної підсистеми.....	77
ВИСНОВКИ	83
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	85

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

JSP	Jakarta Server Pages
HTML	HyperText Markup Language
XML	EXtensible Markup Language
GNU	GNU's Not Unix
CGI	Common Gateway Interface
JVM	Java Virtual Machine
MVC	Model view controller
RAD	Rapid Application Development
API	Application Programming Interface
JNDI	Java Naming and Directory Interface
JDBC	Java Database Connectivity
EJB	Enterprise JavaBeans
HTTP	HyperText Transfer Protocol
IC	Інформаційна система
ER-model	Entity-Relationship model
СКБД	Система керування базами даних

ВСТУП

Актуальність створення веб-платформи для організації роботи онлайн-кінотеатру має декілька переваг у порівнянні з іншими типами сайтів, якщо ціль компанії полягає у прокаті фільмів через інтернет. Такий тип сайту як онлайн-кінотеатр був з самого початку задуманий як ефективний засіб для здійснення прокату фільмів безпосередньо в режимі онлайн. Актуальність теми обумовлена швидким зростанням кількості українських користувачів мережі Інтернет як потенційних споживачів онлайн контенту під час того як все стає цифровим.

Мета кваліфікаційного бакалаврського проекту – навчитися розробляти комп'ютерні системи, а саме веб-платформу для організації роботи онлайн-кінотеатру для прокату контенту з кіноіндустрії в онлайн режимі. Розробити концепції та реалізація інформаційної системи для онлайн-кінотеатру з метою забезпечення користувачам зручного та ефективного способу вибору, оплати та перегляду відео контенту, а також оптимального управління цим контентом для забезпечення їхніх потреб та задоволення.

Практична цінність отриманих результатів полягає в тому, що розроблена комп'ютерна система може бути використана для прокату контенту з кіноіндустрії, а саме фільмів та серіалів як в любительській так і професійній зйомці в мережі Інтернет.

Використані інструментальні засоби: середовище розробки IntelliJ IDEA Ultimate 2023.2.5, система для проектування та розробки баз даних MySQL Workbench 8.0 CE , Draw.IO, UMLetino 14.3 та Lucidchart, операційна система Windows 10 Pro, редактор тексту Microsoft Word 2019 для оформлення і підготовки пояснювальної записки до кваліфікаційного бакалаврського проекту.

Структура роботи визначена її метою і завданнями та складається зі вступу, трьох розділів, висновків, додатків та списку джерел.

РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Характеристика предметної галузі та об'єкта дослідження

Активний розвиток українського ринку інформаційних технологій стимулював нові напрями підприємницької діяльності та створення суб'єктів господарювання. У зв'язку з цим в Інтернеті з'являються компанії, магазини, аукціони та інші платформи з різноманітним контентом, які розширюють можливості та пропонують нові способи отримання товарів і послуг, що, в свою чергу, стимулює розвиток мережевої економіки.

Онлайн-кінотеатр — це інтерактивний веб-сайт, який на основі новітніх технологій рекламує свій контент або товари замовника, здійснює їх продаж, маркетинг, посередницьку діяльність, надає додаткові послуги, пропонує користувачам різні варіанти оплати, виставляє рахунки з метою максимального задоволення потреб споживачів та отримання стабільного прибутку власником [1].

В іншому значенні, Онлайн кінотеатр – це спеціалізований веб-сайт, який належить фірмі-кінокомпанії, телеканалу тощо та призначений для просування контенту на ринку, збільшення обсягів прокату чи повноцінного продажу, залучення нових глядачів.

Детальніше розглянемо основні переваги та недоліки Онлайн кінотеатрів. Відтак, перевагами онлайн кінотеатрів є те, що вони можуть пропонувати значно більшу кількість фільмів та серіалів, ніж реальні кінотеатри і забезпечувати глядачів значно більшим обсягом інформації, що необхідна для вибору фільму чи серіалу. Додатково, використання віртуальних технологій дозволяє налаштувати індивідуальний підхід до клієнтів, враховуючи їх попередні візити до онлайн кінотеатру і переглянутих в ньому фільмів. Використання онлайн кінотеатру як ефективного способу маркетингових досліджень (анкетування, конференції глядачів і т. п.). Крім того, онлайн кінотеатри дають можливість цілодобово переглядати контент, не здійснюючи тиску на глядача.

До недоліків онлайн кінотеатрів можна віднести: низьку якість запису і звуку, якщо ви хочете дивитися фільми онлайн безкоштовно, передачі, мультики або кліпи. Тим не менш, ви можете стати повноправним користувачем платних

кінотеатрів в режимі онлайн, і вам будуть доступні новинки індустрії кіно у самому високому дозволі. Зрозуміло, для того, щоб отримати задоволення від перегляду, у вас повинна бути відповідна техніка, а також підключення до високошвидкісного і безлімітного Інтернету.

Для роботи онлайн-кінотеатру необхідно використовувати веб-вітрину. Веб-вітрина – це поєднання товарного каталогу, системи навігації та можливостей для купівлі або прокату контенту. Паралельно проводиться рекламна діяльність, вивчення попиту та аналітична робота.

Функціонування веб-вітрини схоже на посилкову торгівлю, але замовлення, ознайомлення з товаром та його отримання здійснюються через Інтернет. В цьому випадку веб-вітрина виступає інструментом залучення покупця, інтерфейсом для взаємодії з ним та проведення маркетингових заходів.

Основні вимоги, які ставлять користувачі до онлайн-кінотеатру, включають:

- зрозумілий інтерфейс та зручну систему навігації;
- зручну систему посилань, що дозволяє оптимально отримати необхідну інформацію;
- мінімальну кількість дій для здійснення покупки.

Виходячи з вищесказаного, метою створення веб-системи є залучення цільової аудиторії для перегляду контенту з кіноіндустрії, а саме перегляд серіалів та фільмів як за гроші так і безкоштовно, що надає цілодобовий доступ до цього контенту.

Майбутніми клієнтами онлайн кінотеатру будуть люди, що бажають переглядати контент з кіноіндустрії онлайн.

Найвищою популярністю користуються такі онлайн кінотеатри як: NETFLIX, MEGOGO, Apple TV+, Sweet TV та інші.

1.2. Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

В області прокату або повноцінного надання перегляду контенту з кіноіндустрії лідируючі позиції займають NETFLIX, MEGOGO, Apple TV+ та SWEET.TV. Розглянемо кожен з них окремо.

1.2.1. Дослідження існуючих інформаційних систем для обраної предметної області.

NETFLIX — одна з найпопулярніших стрімінгових платформ у світі, що пропонує тисячі серіалів і фільмів, які стали культовими в останні роки.

Головна перевага NETFLIX — унікальний контент, який важко знайти в інших місцях, особливо на телебаченні. Наприклад, серіал "Дуже дивні справи" має величезну популярність[2].

Основний недолік цієї платформи для українських користувачів — відсутність дубляжу українською та російською мовами, що характерно і для інших іноземних платформ. Проте є значна кількість контенту з перекладеними субтитрами.

Передплата Basic, яка коштує 5,99 євро, дозволяє дивитися NETFLIX лише з одного пристрою без HD-якості. За передплату Standard, яка коштує 9,99 євро, можна дивитися фільми з двох пристроїв одночасно у HD-якості. Максимальна передплата Premium за 11,99 євро дозволяє переглядати контент на чотирьох пристроях одночасно у якості Ultra HD. У всіх варіантах передбачено безлімітний доступ до будь-яких серіалів, програм і фільмів. На рис. 1.1 представлено головну сторінку веб-сайту[3].

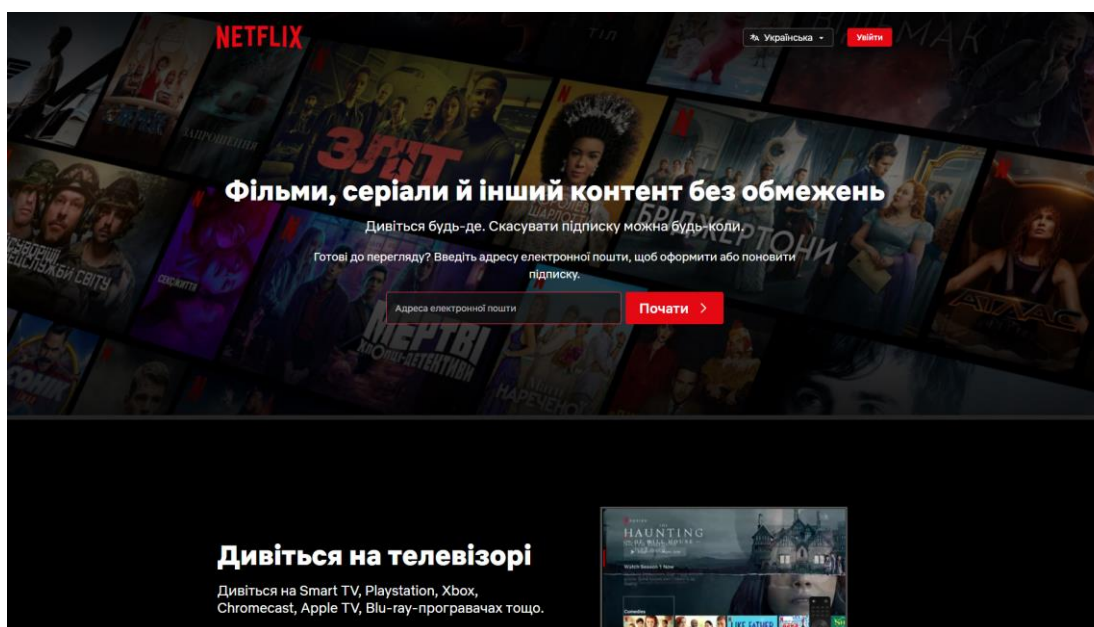


Рисунок 1.1 – Головна сторінка онлайн кінотеатру NETFLIX

MEGOGO — мабуть, найпопулярніший легальний онлайн-кінотеатр в Україні, який значно розширився за останні кілька років.

MEGOGO навіть має власні телеканали, де транслюють різні шоу, спортивні події тощо, причому багато з них доступні безкоштовно.

Бібліотека фільмів і серіалів на платформі значно розширилася, хоча найвибагливіші користувачі можуть скаржитися на нестачу якісного контенту та надлишок російських серіалів і фільмів.

MEGOGO пропонує три варіанти передплати: за 129, 199 і 299 грн/місяць. Вони відрізняються кількістю телеканалів і доступних для перегляду фільмів[4].

Однак, якщо ви хочете легально подивитися конкретний фільм у високій якості, його можна купити окремо. За більшість фільмів у бібліотеці просять від 20 до 100 грн. На рис. 1.2 представлено головну сторінку веб-сайту[5].

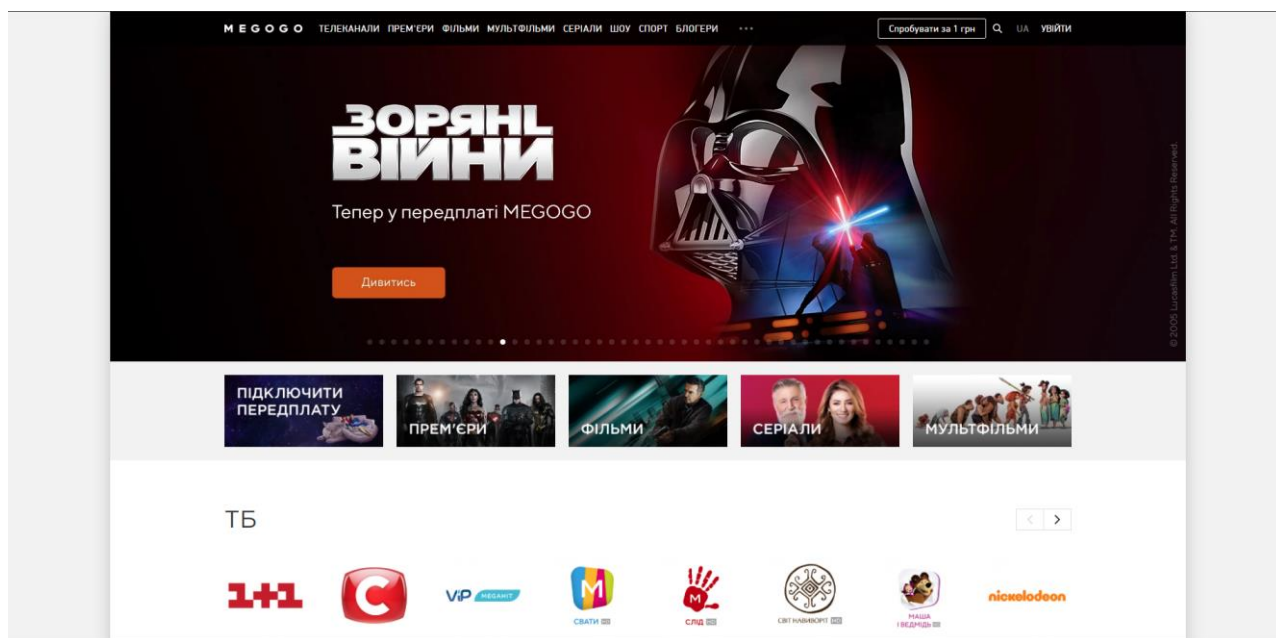


Рисунок 1.2 – Головна сторінка онлайн кінотеатру MEGOGO

Контент Apple TV+ можна переглядати на веб-сайті Apple або через додаток Apple TV, який з 2019 року став доступним на різних побутових пристроях і платформах (включаючи конкурентні). З часом планується збільшення кількості пристроїв, що підтримують цей додаток.

Представництво в Україні: можна звернутися до служби підтримки через сайт, платформа підтримує українські субтитри та український дубляж[6].

Наявність 4K UHD-відео: є, також підтримується звук Dolby Atmos.

Вартість місячної передплати: \$4,99, можливий тиждень безкоштовного пробного періоду, а за передплатою Apple One — місяць безкоштовного доступу.

Плюси платформи: велика кількість серіалів з високою якістю відео. Окремою перевагою є безкоштовні для підписників продукти, створені студією Apple Originals. Раніше вважалося, що без наявності TV-приставки від Apple не можна переглядати контент від Apple Originals, проте тепер будь-який смарт-телевізор підтримує цей застосунок.

Мінуси платформи: за окремі бажані фільми доведеться заплатити додатково. Фільми у високій якості мають зависоку ціну для українського ринку. Незважаючи на нижчу вартість підписки порівняно з Netflix, Apple TV+ пропонує менше контенту — трохи більше ніж 50 серіалів і 30 фільмів. На рис. 1.3 представлено головну сторінку веб-сайту[7].

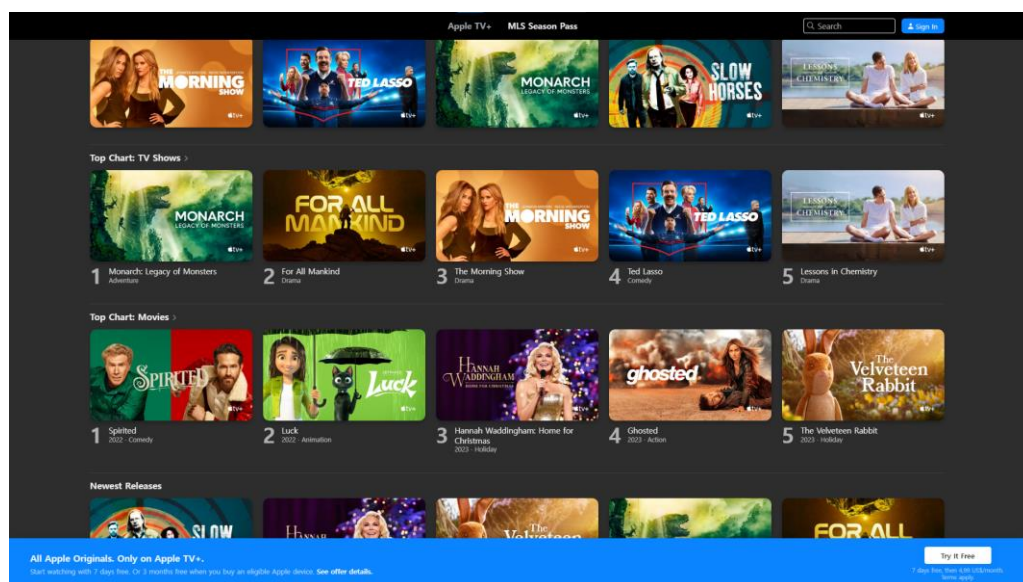


Рисунок 1.3 – Головна сторінка онлайн кінотеатру Apple TV+

У вересні 2020 року sweet.tv став першою платформою відео на вимогу в Україні, яка почала додавати легальний український дубляж до фільмів та серіалів. Вони уклали прямі та непрямі угоди з такими студіями, як Walt Disney

Pictures/20th Century Studios, Paramount Pictures, Universal Pictures (через російський підрозділ PictureBox Russia), Sony Pictures та Warner Bros. Entertainment. Голлівудські студії надавали sweet.tv власний легальний україномовний дубляж для фільмів, вироблених після 2006 року, а для старіших фільмів (вироблених до 2006 року, коли в українському кінопрокаті ще не було україномовного дубляжу) компанія створювала власний дубляж у межах проекту «Hollywood українською»[8].

Головний недолік цієї платформи полягає в тому, що деякий контент є тимчасовим, і умови його перегляду можуть змінюватися з припиненням прав сервісу на безкоштовну трансляцію фільмів та серіалів. За додаткові фільми та серіали, які відсутні в передплаті, доведеться платити окремо або брати їх напрокат.

Передплата S, яка коштує 99 грн/міс, не включає фільмові, преміальні, дитячі та спортивні телеканали. Максимальна передплата L за 197 грн/міс включає перегляд фільмових, преміальних, дитячих та спортивних телеканалів. З максимальним планом доступний безлімітний доступ до будь-яких серіалів, програм і фільмів при наявності інтернет-з'єднання у будь-якій країні світу. На рис. 1.4 представлено головну сторінку веб-сайту[9].

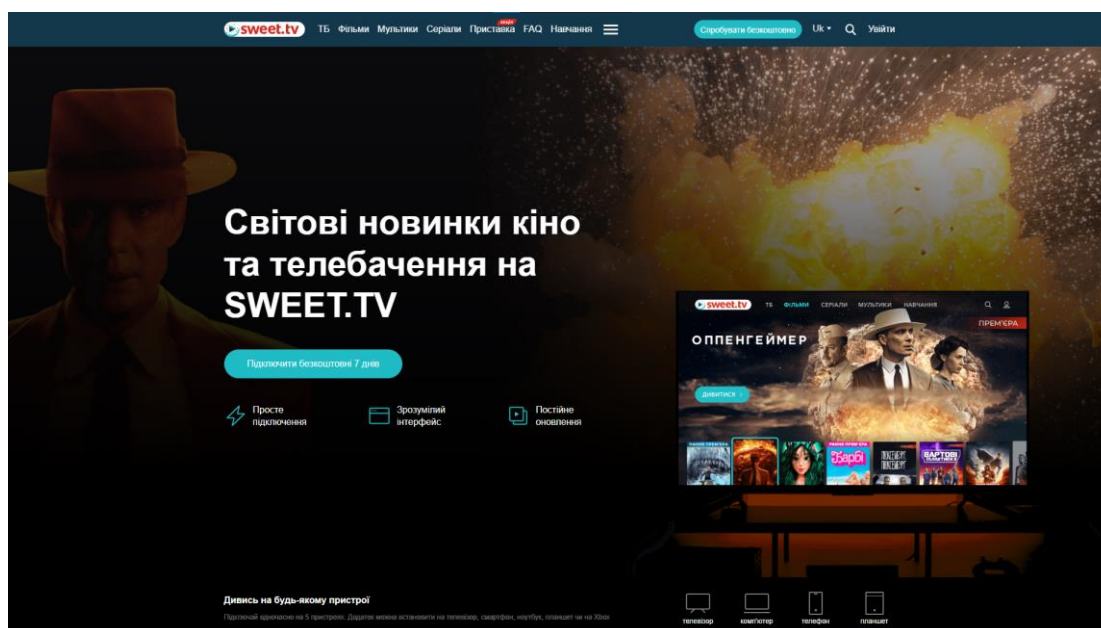


Рисунок 1.4 – Головна сторінка онлайн кінотеатру SWEET.TV

1.2.2. Порівняння програмних систем

Результати порівняння програмних систем NETFLIX, MEGOGO, DIVAN.TV та , наведені в табл. 1.1:

Таблиця 1.1 – Порівняння програмних систем

№	Можливості	Програмні системи			
		NETFLIX	MEGOGO	Apple TV+	SWEET.TV
1	Оригінальний контент	Так	Ні	Так	Ні
2	Піратський дубляж	Ні	Ні	Ні	Так
3	Наявність 4K UHD-Video	Так	Так	Так	Так
4	Окрема плата за додатковий контент	Ні	Так	Так	Так
5	Можливість спробувати сервіс безкоштовно	Так	Так	Так	Так
6	Наявність реклами	Так	Так	Так	Так
7	Зрозумілий інтерфейс	Ні	Так	Ні	Так
8	Обмеження контенту за країною	Так	Ні	Ні	Ні
9	Обмеження контенту через відсутність оригінальної приставки	Ні	Ні	Так	Ні
10	Заборона на обмін паролями	Так	Ні	Ні	Ні

Отже у процесі аналізу програмних систем NETFLIX, MEGOGO, Apple TV+ та SWEET.TV виявлено, що всі ці програмні системи мають особливі характеристики.

NETFLIX і Apple TV+ є провідними у світовому стрімінговому просторі, з першою здобутою популярністю завдяки великій кількості оригінального контенту та, в разі NETFLIX, унікальному характеру деяких продуктів. Однак, у випадку NETFLIX, проблемою для українських користувачів може стати відсутність дубляжу українською та російською мовами.

MEGOGO визначається розширенням своєї бібліотеки фільмів і серіалів, але деякі користувачі можуть висловлювати скарги на якість контенту та надлишок російської продукції. Зазначено також можливість окремої оплати за додатковий контент.

Apple TV+ пропонує безкоштовний період проби та визначається якісним оригінальним контентом. Однак, порівняно з іншими платформами, кількість контенту є меншою, а деякі фільми можуть бути платними.

SWEET.TV виступає як інноваційна стрімінгова платформа в Україні з доступом до HD-телебачення та українських фільмів. Зазначено, що деякий контент може бути тимчасовим, змінюючи умови доступу.

У підсумку, вибір між цими платформами залежить від особистих вподобань та вимог користувача, оскільки кожна з них має свої переваги та обмеження.

1.2.3. Позитивні характеристики існуючих систем, що їх слід відтворити у проектних рішеннях

Існуючі стрімінгові платформи, такі як NETFLIX, MEGOGO, Apple TV+ та SWEET.TV, володіють рядом позитивних характеристик, які можна врахувати при проектуванні нових систем:

1. Оригінальний контент:

- Високоякісний та унікальний оригінальний контент привертає аудиторію та відзначається власним стилем.

2. Відсутність реклами для платників:

- Можливість перегляду контенту без рекламних вставок для платників привертає користувачів та створює комфортний перегляд.

3. Можливість пробного періоду:

- Наявність безкоштовного пробного періоду дозволяє користувачам випробувати сервіс перед прийняттям рішення про підписку.

4. Висока якість контенту (4K UHD):

- Підтримка високої якості відео (4K UHD) сприяє залученню та задоволенню вимог вибагливих глядачів.

5. Мобільні додатки та кросплатформеність:

- Наявність додатків для різних платформ та пристроїв розширює досягнення аудиторії.

6. Зручний інтерфейс користувача:

- Простий та зрозумілий інтерфейс сприяє легкому навігації та використанню платформи.

7. Локалізація контенту:

- Наявність дублювання або субтитрів на різних мовах забезпечує широкий географічний охоплення аудиторії.

8. Опції підписок та пакети послуг:

- Різноманітні пакети передплат і послуг дозволяють користувачам вибирати оптимальний варіант для своїх потреб.

9. Системи підтримки та зручність звернення за допомогою:

- Існування системи підтримки користувачів та можливість швидкого звернення за допомогою сприяє розв'язанню проблем та задоволенню користувачів.

10. Використання технологій нового покоління:

- Інноваційні технології, такі як 4K UHD-Video, Dolby Atmos, та інші, підвищують якість перегляду та роблять платформу привабливою.

Враховуючи ці позитивні характеристики, при розробці нових стрімінгових проєктів слід приділяти увагу створенню унікального та

високоякісного контенту, оптимізації користувацького інтерфейсу, а також вдосконаленню послуг та підтримки для задоволення потреб сучасної аудиторії.

1.2.4. Недоліки існуючих систем та напрями їх удосконалення

Існуючі стрімінгові платформи мають свої недоліки, які можна врахувати для подальшого удосконалення:

1 Одноманітність контенту:

- Деякі платформи можуть стикатися з проблемою одноманітності або переваги деяких жанрів над іншими. Удосконалення полягає в розширенні різноманітності контенту та привертанні широкої аудиторії.

2 Обмеження регіонального контенту:

- Деякі сервіси обмежують доступ до певного контенту залежно від регіону. Удосконалення включає в себе розширення географії доступу до контенту та уникнення обмежень.

3 Відсутність синхронізації контенту:

- Відсутність можливості синхронізації перегляду між різними пристроями може бути недоліком. Удосконалення включає в себе розробку ефективної системи синхронізації та продуктивніше управління переглядом на різних пристроях.

4 Проблеми з якістю стрімінгу:

- Затримки або низька якість стрімінгу можуть впливати на задоволення користувачів. Удосконалення включає в себе вдосконалення інфраструктури для стабільного та високоякісного стрімінгу.

5 Брак деяких функцій на різних пристроях:

- На деяких пристроях може відсутній функціонал, який присутній на інших. Удосконалення полягає в розробці універсального та консистентного інтерфейсу на різних платформах.

6 Вартість послуг та вартість додаткового контенту:

- Вартість деяких послуг та додаткового контенту може бути високою. Удосконалення включає в себе вивчення можливостей зниження цін та приведення їх у більший відповідність забезпеченому сервісу.

7 Неоднакові умови користування для різних регіонів:

- Удосконалення включає в себе встановлення більш однакових умов користування для користувачів у різних країнах, з урахуванням їхніх потреб та вимог.

8 Проблеми з безпекою та обміном паролями:

- Удосконалення має включати в себе підвищення рівня безпеки та впровадження ефективних засобів захисту від обміну паролями між користувачами.

9 Недостатня прозорість та інформаційна підтримка:

- Удосконалення включає в себе покращення прозорості щодо умов користування, а також забезпечення зрозумілої та доступної інформаційної підтримки для користувачів.

10 Співпраця з ліцензійними власниками:

- Вдосконалення включає в себе активну співпрацю з ліцензійними власниками для підтримання прав на контент та уникнення правових питань.

Враховуючи ці недоліки, розробники можуть покращити стрімінгові платформи та забезпечити їхню більшу ефективність та задоволення користувачів.

РОЗДІЛ 2. РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1. Аналіз і специфікація вимог до інформаційної підсистеми

2.1.1. Бізнес-вимоги до системи

Розглянемо бізнес вимоги і бізнес-цілі що висуваються стейкхолдерами до веб-платформи для організації роботи онлайн-кінотеатру, що представлені на рис. 2.1 – 2.7.

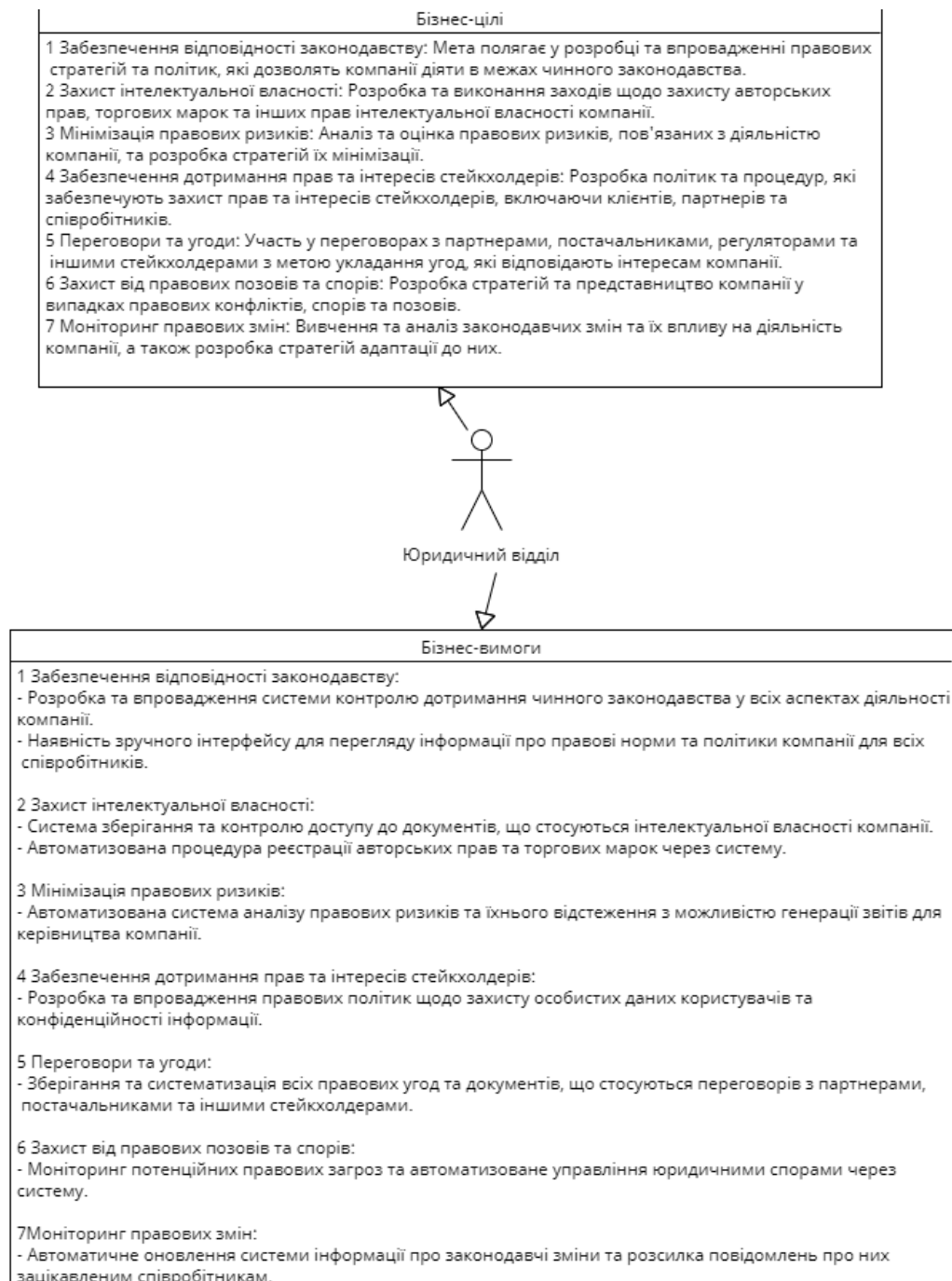


Рисунок 2.1 – Бізнес вимоги та бізнес-цілі юридичного відділу

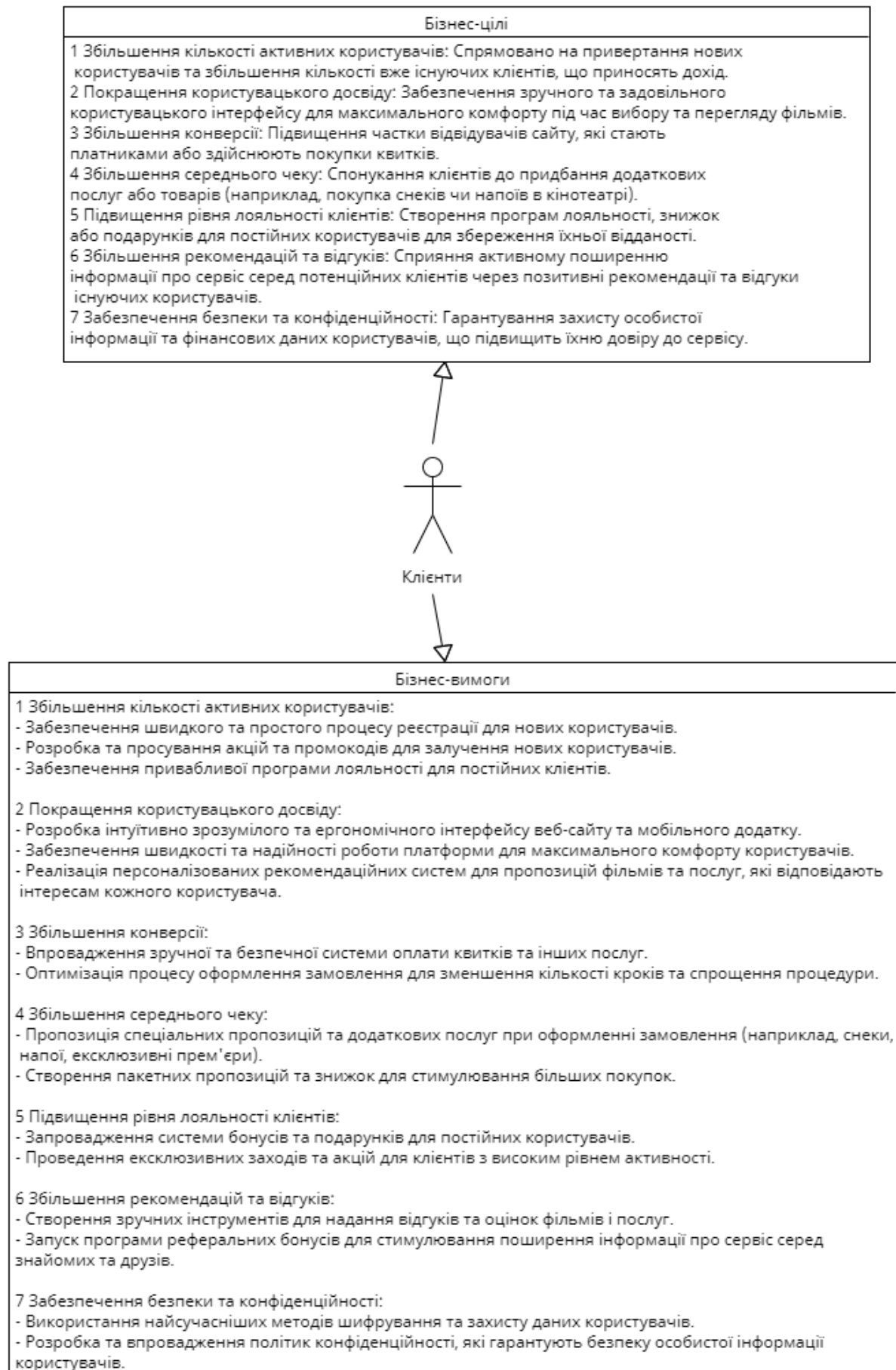


Рисунок 2.2 – Бізнес вимоги та бізнес-цілі клієнтів

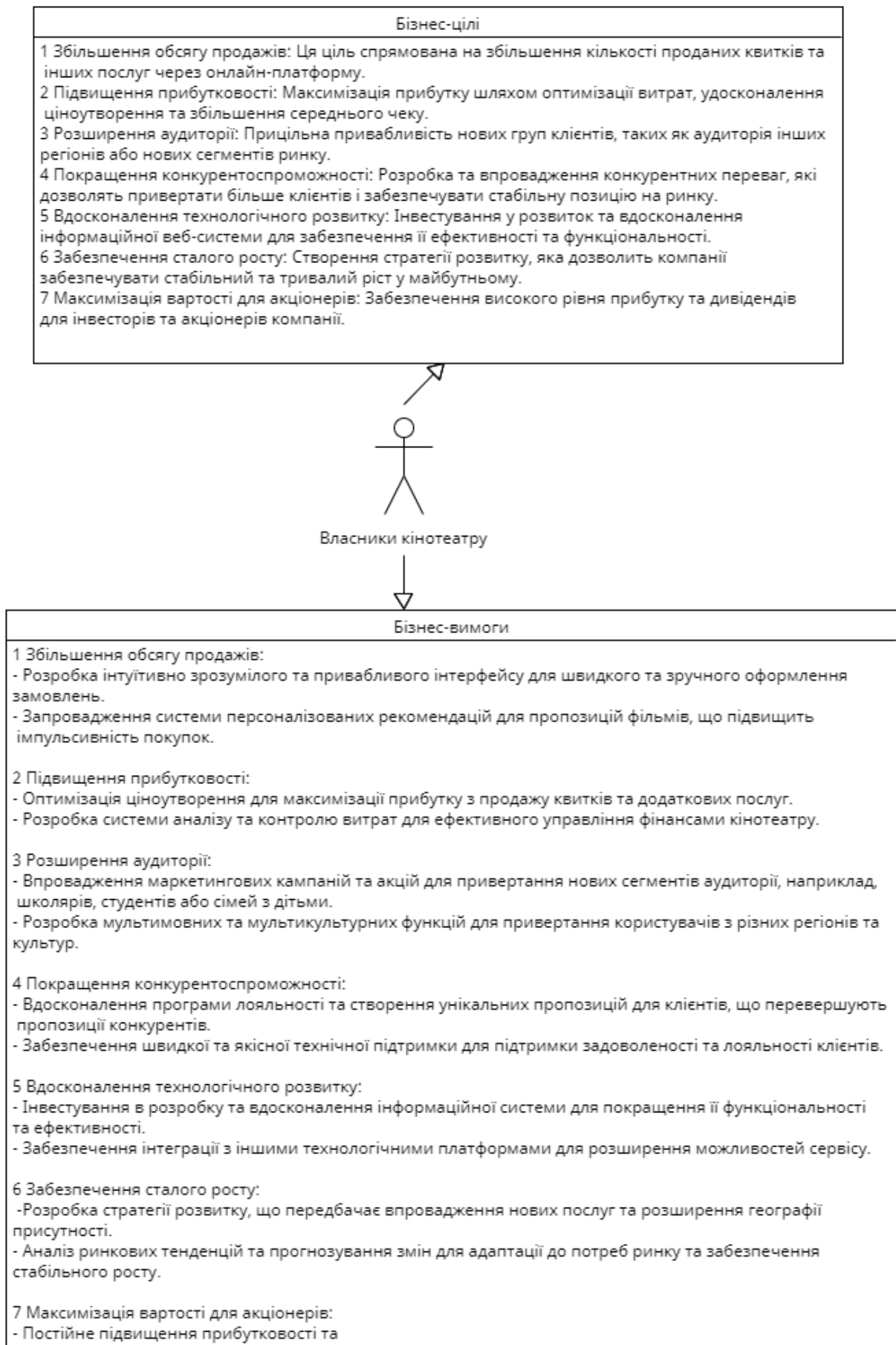


Рисунок 2.3 – Бізнес вимоги та бізнес-цілі власників кінотеатру

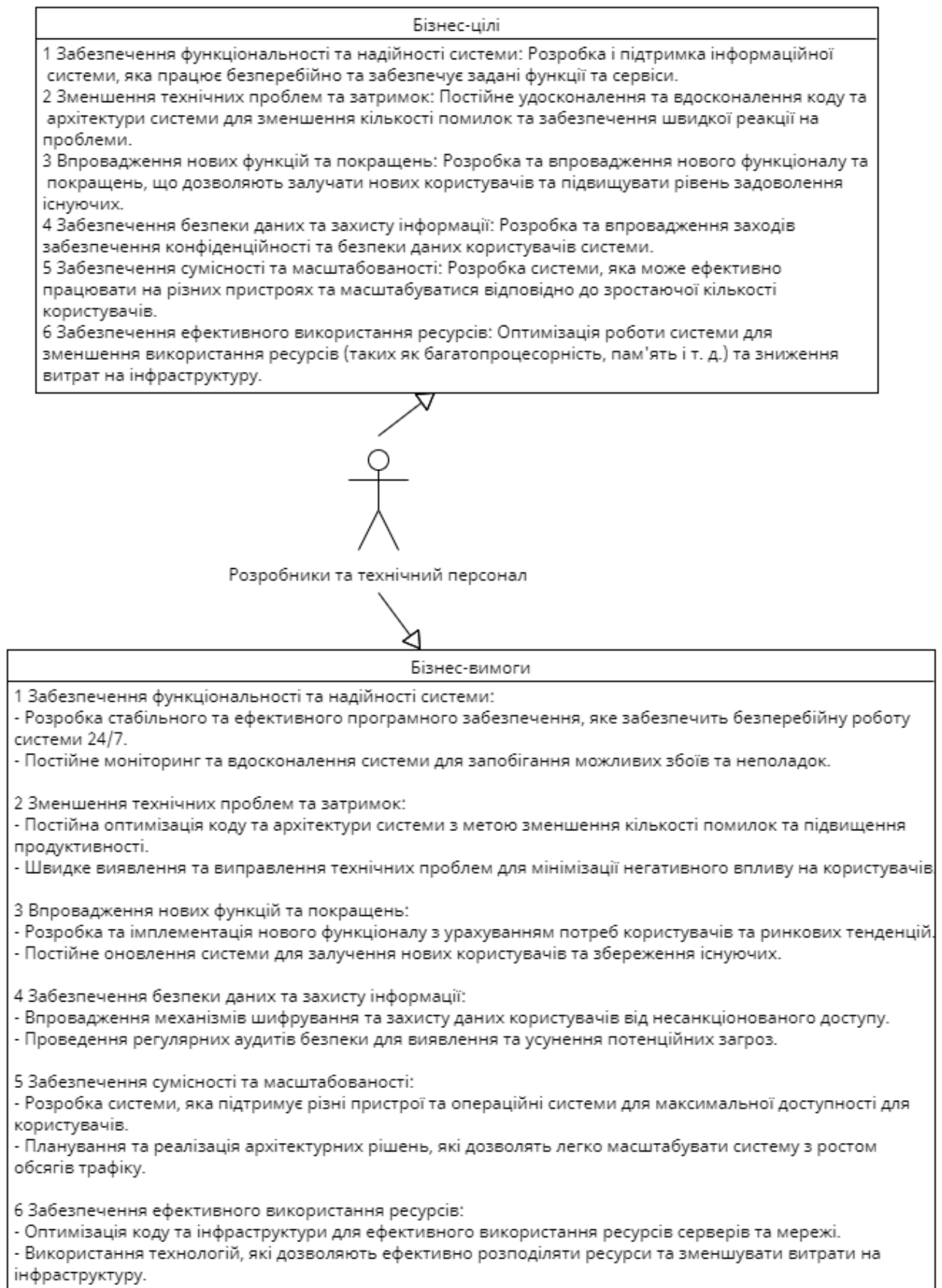


Рисунок 2.4 – Бізнес вимоги та бізнес-цілі розробників та технічного персоналу

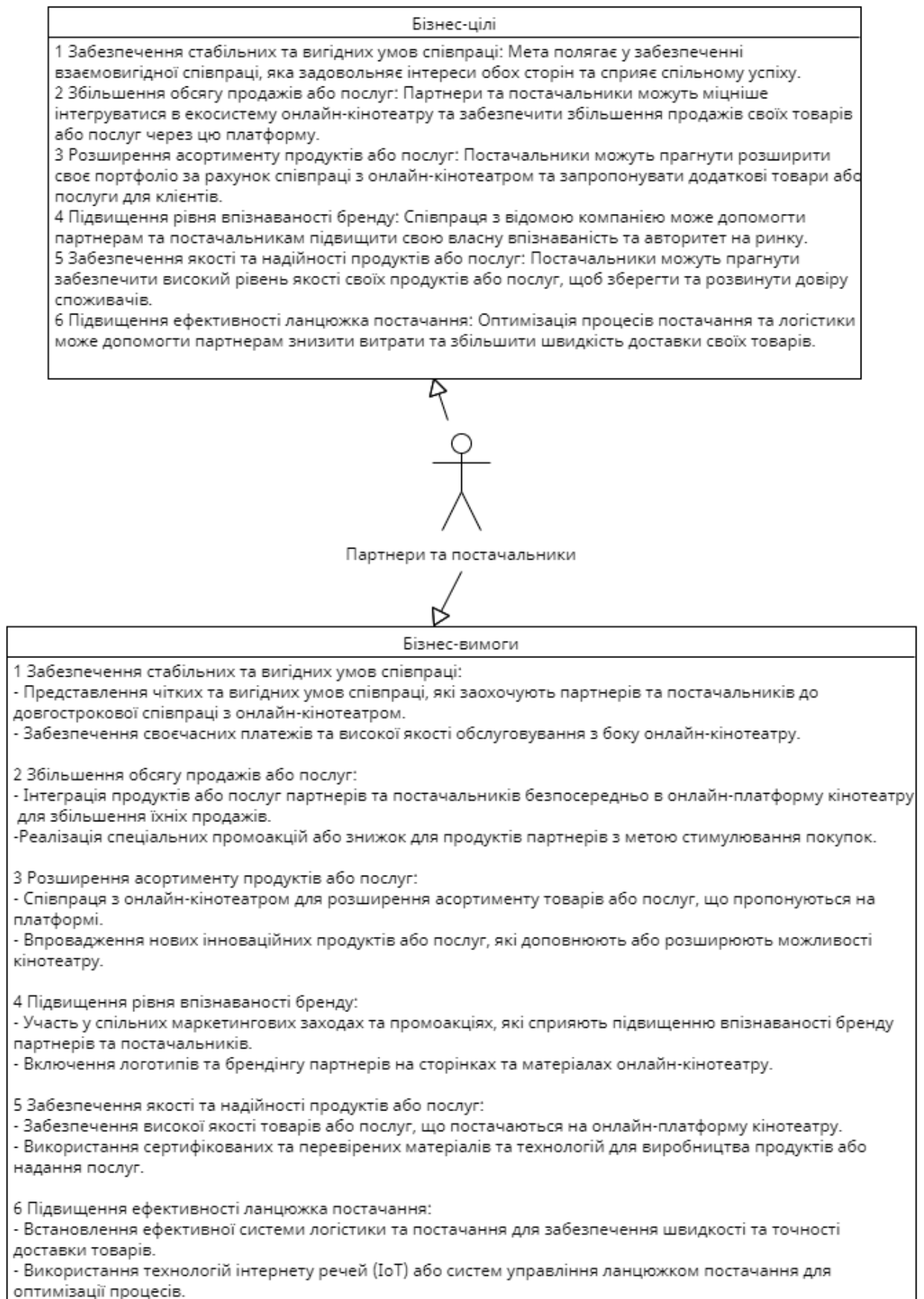


Рисунок 2.5 – Бізнес вимоги та бізнес-цілі партнерів та постачальників

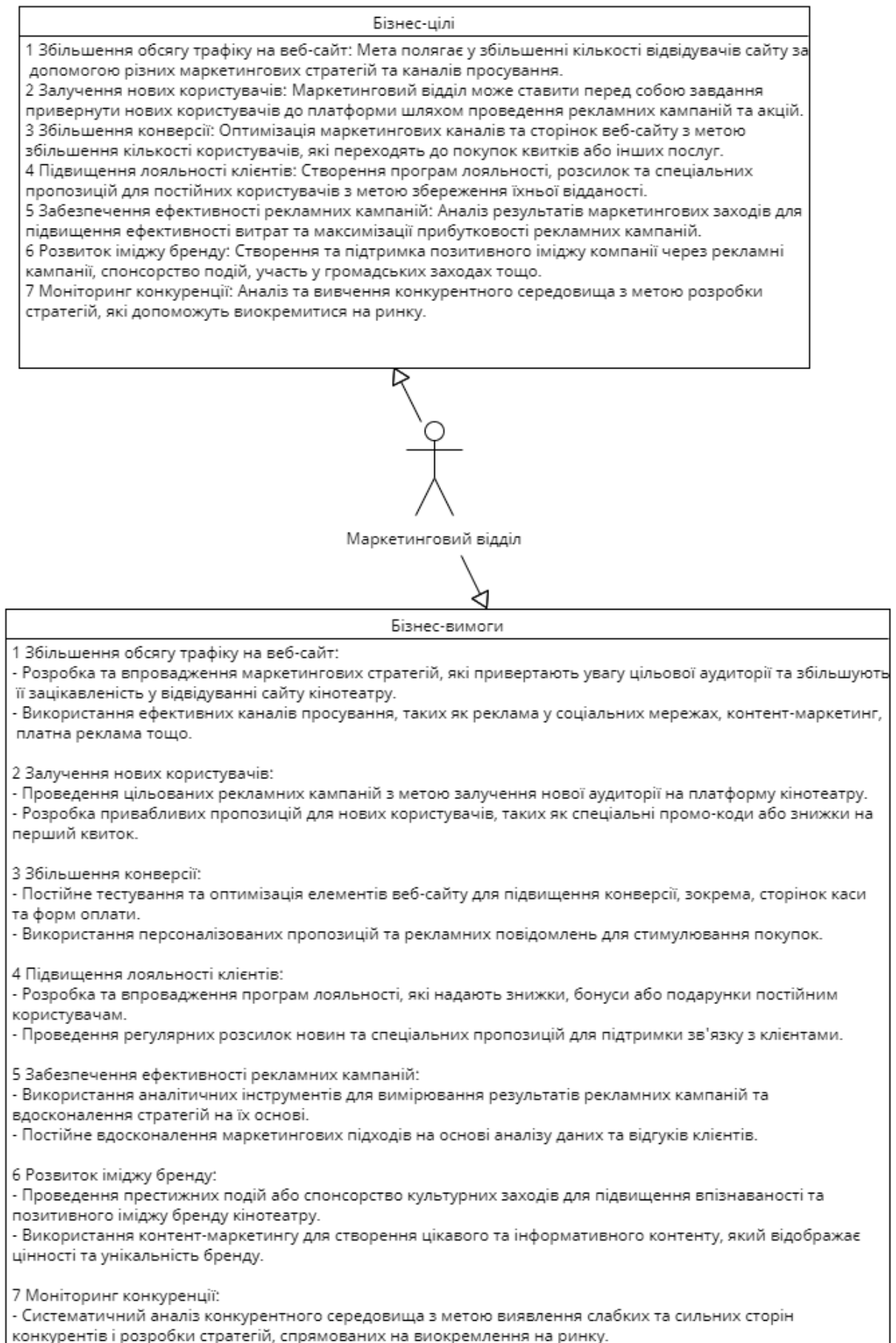


Рисунок 2.6 – Бізнес вимоги та бізнес-цілі маркетингового відділу

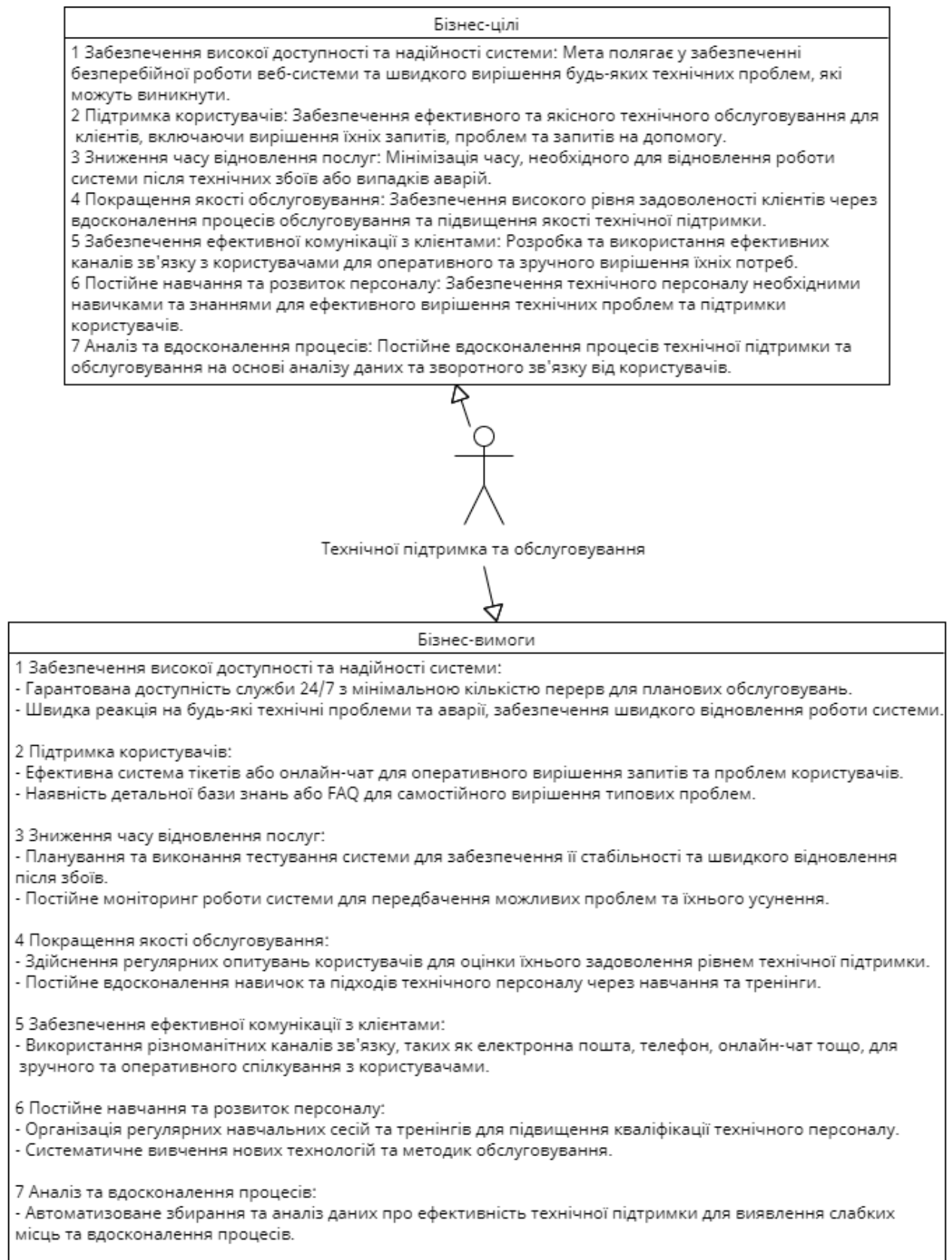


Рисунок 2.7 – Бізнес вимоги та бізнес-цілі технічної підтримки та обслуговування

2.1.2. Функціональні вимоги до системи

Функціональні вимоги до веб-платформи для організації роботи онлайн-кінотеатру включають такі основні аспекти:

Аутентифікація:

Користувачам має бути надана можливість реєстрації на платформі шляхом подання запиту на реєстрацію. Після реєстрації користувачі повинні мати можливість виконувати вхід на платформу, надаючи вірні облікові дані. Після входу система має надавати права доступу відповідно до рівня аутентифікації. Крім того, система повинна надсилати сповіщення користувачам про некоректно вказані дані під час процесу реєстрації або входу.

Пошук відеоматеріалу:

Платформа повинна забезпечити можливість користувачам подавати запити на виведення найпопулярніших відеоматеріалів або списку доступних відеоматеріалів. Крім того, користувачі повинні мати змогу обирати необхідний відеоматеріал з виведеного списку.

Оплата:

Користувачам має бути надана можливість подавати запити на оплату обраних відеоматеріалів. Після цього користувачі повинні мати можливість здійснювати оплату обраним методом (наприклад, кредитною карткою або електронним переказом). Після успішної оплати система має надавати користувачам чек про здійснену оплату, а також надсилати сповіщення про невдачу оплати.

Перегляд та обговорення відеоматеріалу:

Користувачам має бути надана можливість перевіряти наявність відеоматеріалу в їхній бібліотеці на платформі. Крім того, користувачі повинні мати можливість переглядати обраний відеоматеріал. Додатково, користувачі повинні мати можливість подавати запити на додавання відгуку та оцінки до відеоматеріалу.

Отже дані елементи зображені на діаграмі функціональних вимог представлених на рис. 2.8.

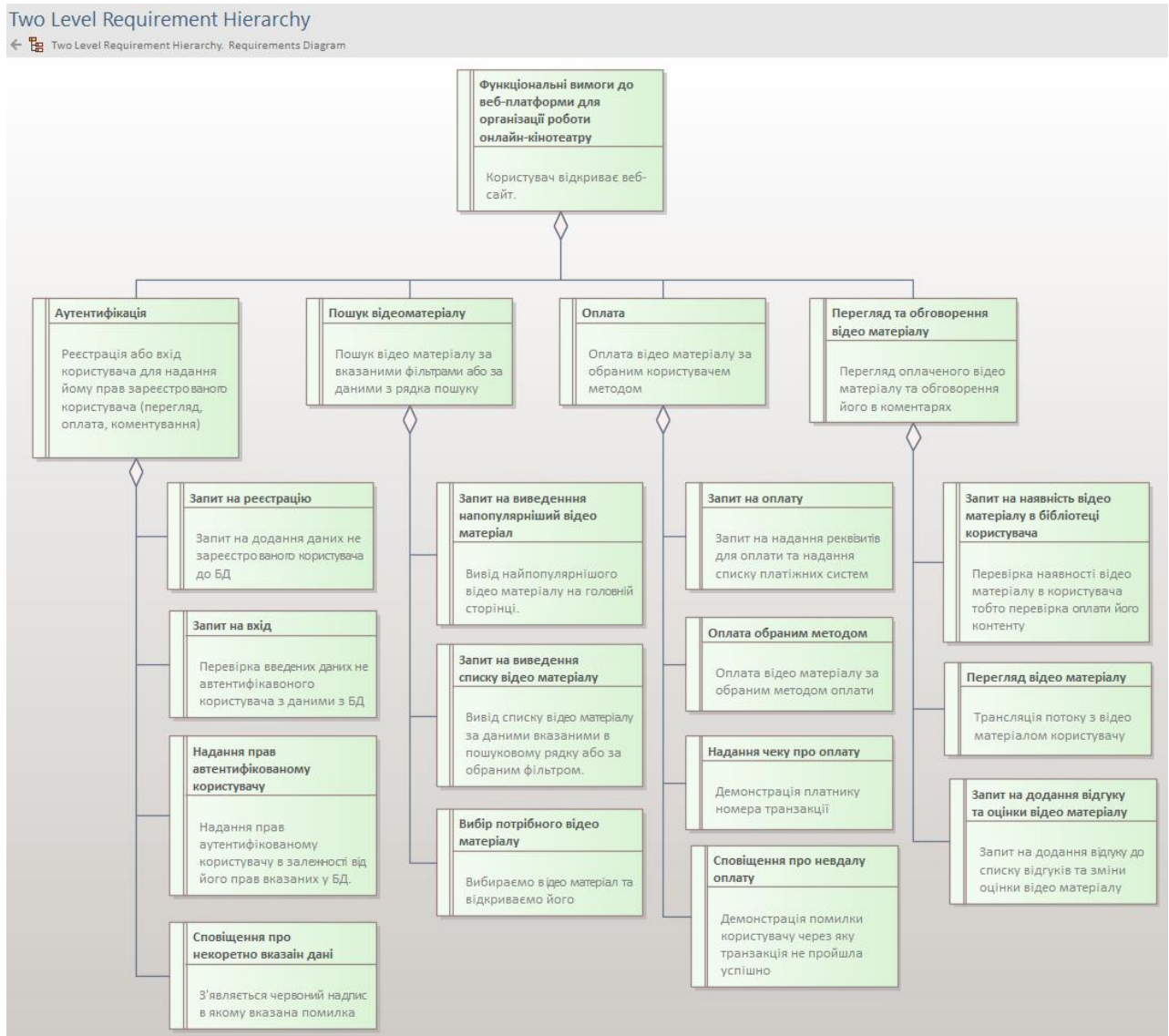


Рисунок 2.8 – Функціональні вимоги до веб-системи веб-платформи для організації роботи онлайн-кінотеатру

2.1.3. Нефункціональні вимоги до системи

Нефункціональні вимоги до веб-платформи для організації роботи онлайн-кінотеатру мають вирішувати різні аспекти її функціонування та взаємодії з користувачами і середовищем. Продуктивність системи передбачає забезпечення швидкості роботи, мінімальний час завантаження сторінок та ефективне використання ресурсів сервера. Масштабованість означає здатність системи адаптуватися до зростання обсягів трафіку і користувачів без втрати продуктивності. Безпека системи полягає в захисті даних користувачів, запобіганні несанкціонованому доступу і забезпеченні конфіденційності.

Сумісність передбачає правильне функціонування платформи на різних пристроях і веб-браузерах. Доступність системи означає забезпечення доступу до неї для всіх користувачів, в тому числі для осіб з обмеженими можливостями. Надійність полягає в стабільності і неперервності роботи системи, запобіганні відмовам і втратам даних. Локалізація і локалізованість забезпечують адаптацію системи до різних регіонів і мов. Адаптивний дизайн означає, що платформа повинна відображатися коректно на різних пристроях і розмірах екранів. Відповідність законодавству передбачає дотримання всіх вимог щодо авторських прав, захисту даних і інших правових норм. Документація повинна бути докладною і зрозумілою для розробників і адміністраторів системи, щоб забезпечити її ефективне управління і підтримку. Отже дані елементи зображені на діаграмі не функціональних вимог представлених на рис. 2.9.

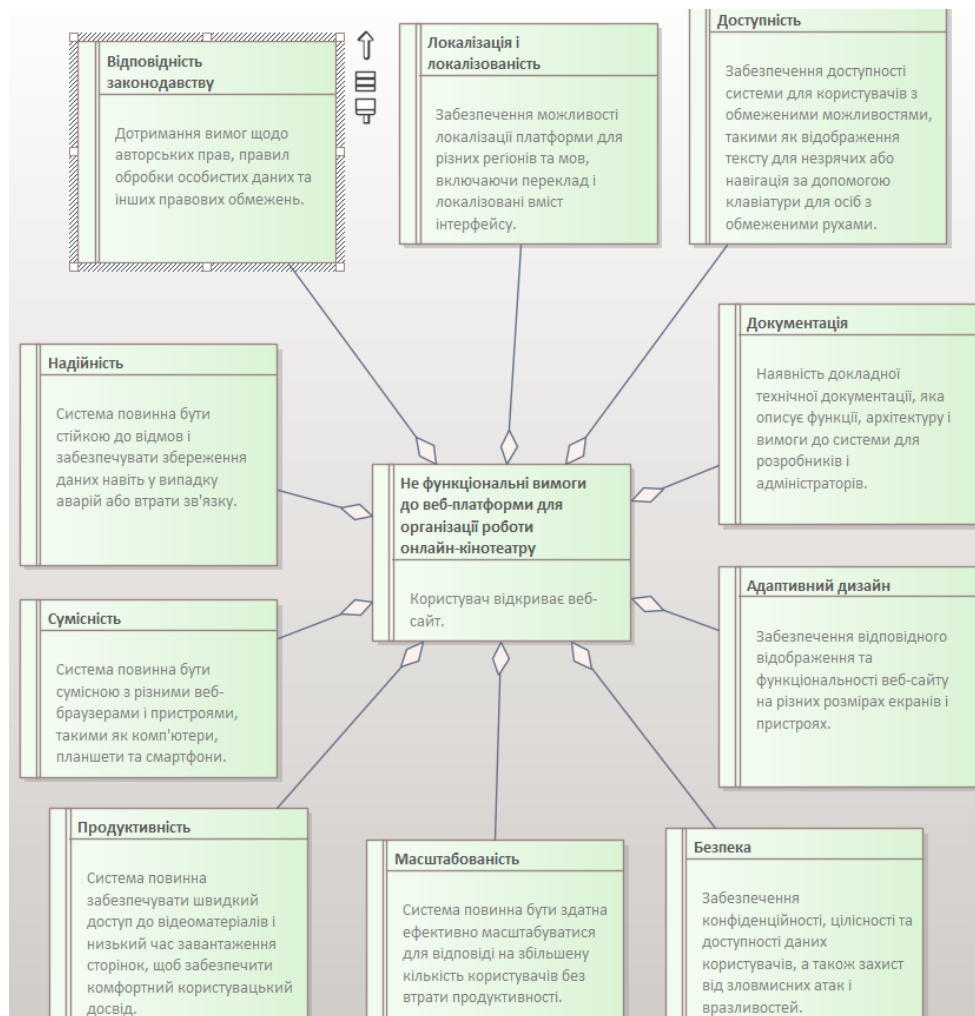


Рисунок 2.9 – Не функціональні вимоги до веб-системи веб-платформи для організації роботи онлайн-кінотеатру

2.2. Постановка задачі

2.2.1. Характеристика задачі

Важливим аспектом і складовою утворення загальної концепції веб-платформи для організації роботи онлайн-кінотеатру ми розглядаємо таку складову, як її інформаційна модель. При її розробці нам важливо розуміти майбутні взаємозв'язки даних, що входять в систему і їх кінцевий результат оброблений самою системою. Мета яку ми намагаємось досягти при розробці цієї системи полягає у наданні споживачу більш кількісних та якісних можливостей функціонування даної системи при її використанні. Інформаційну модель задачі представлено на рис. 2.10.

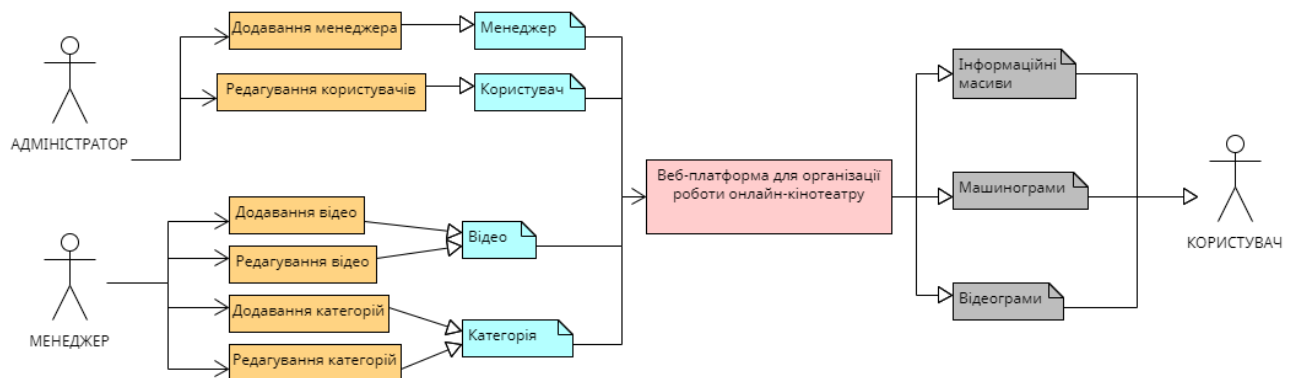


Рисунок 2.10 – Інформаційна модель задачі

Вхідна інформація:

- Реєстрація користувача: Процес, під час якого новий користувач подає необхідну інформацію для створення облікового запису веб-системи "Онлайн кінотеатр". Ці дані передаються в базу даних для подальшого створення облікового запису.
- Оновлення особистої інформації користувача - це дія, під час якої він або вона вносить зміни у свої особисті дані, такі як ім'я, адреса, контактні дані та інше. Ці зміни автоматично відображаються в базі даних, щоб забезпечити актуальність інформації.

- Додавання нових категорій: Створення нових категорій для класифікації відеоматеріалів, яке полегшує їх подальшу класифікацію та пошук для користувачів.
- Додавання нової продукції: Дія, коли адміністратор або відповідальна особа системи додає новий відеоматеріал, такий як фільм, серіал або відео курс, в базу даних.
- Придбання продукції: Дія, коли користувач здійснює покупку відеоматеріалу, і ця інформація оновлюється в базі даних, щоб надати користувачеві доступ до контенту.
- Додавання нових менеджерів: Створення облікових записів для нових менеджерів, які будуть відповідати за керування різними аспектами системи.

Вихідна інформація:

- Повідомлення про успішну реєстрацію: Електронні листи або SMS, які надсилаються користувачеві щодо успішної реєстрації.
- Повідомлення про успішну покупку: Електронні листи або SMS, які надсилаються користувачеві після успішної покупки відеоматеріалу.
- Повідомлення про вихід нових відеоматеріалів: Електронні листи або SMS, які регулярно надсилаються користувачам щодо нового контенту.
- Повідомлення про зміну ролі: Електронні листи або SMS, які надсилаються користувачам при зміні їх ролі в системі.
- Трансляція відео користувачу: Відео плеєр, який надає користувачеві можливість переглядати відеоматеріали.

Обсяг прибутку, кількість проданих відеоматеріалів, популярність відеоматеріалів: Таблиці, які містять обчислені показники і можуть використовуватися для аналізу та звітності.

2.2.2. Вихідна інформація

Веб-платформа для організації роботи онлайн-кінотеатру використовує реляційну базу даних для збереження і керування інформацією про відео-контент і користувачів. Вона створює різноманітні вихідні повідомлення, які використовуються для інформування про стан бази даних або передачі результатів операцій.

Вихідні повідомлення - це повідомлення, які формуються системою для інформаційного зв'язку з іншими задачами або для відображення на екрані за запитом користувача. Вони можуть бути різних типів, залежно від призначення та форми подання. Основні типи вихідних повідомлень:

1. Машинограми, що видаються на друк - це документи, які формуються автоматично системою та містять інформацію про покупку автомобілю, замовлення на сервісне обслуговування, резервування автомобілю за клієнтом та інші повідомлення. Ці записи зберігаються в базі даних і можуть бути використані як посібник для звітності та аналізу функціонування системи.

2. Вихідні файли, що формуються як результат вирішення розглянутих завдань, призначені для обміну інформацією з іншими завданнями. Ці файли містять результати виконання розглянутих завдань і можуть бути використані іншими системами для подальшої обробки. Наприклад, це можуть бути файли з даними про продажі автомобілів, які можуть бути імпортовані в систему обліку бухгалтерських операцій.

3. Відеокадри, що відтворюються на екрані за запитом користувачів, є результатом вихідних повідомлень, що містять відеозаписи про автомобілі та обслуговування, і можуть бути показані на екрані користувача за його запитом. Ці повідомлення можуть використовуватися для демонстрації автомобілів у салоні або для надання користувачеві додаткової інформації про сервісне обслуговування.

Перелік та опис вихідних повідомлень системи наведено у табл. 2.1.

Таблиця 2.1 – Перелік та опис вихідних повідомлень системи

Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувач інформації
1	2	3	4	5	6
Повідомлення про успішну реєстрацію	MES-REG	Електронний лист, SMS	Щоденно	Термін видання - негайно, допустимий час затримки – 5 хвилин	Користувач
Повідомлення про успішну покупку	MES-ORD1	Електронний лист, SMS	Щоденно	Термін видання - негайно, допустимий час затримки – 5 хвилин	Користувач
Повідомлення про невдачу покупки	MES-OED0	Електронний лист, SMS	Щоденно	Термін видання - негайно, допустимий час затримки – 5 хвилин	Користувач

Продовження таблиці 2.1 – Перелік та опис вихідних повідомлень системи

1	2	3	4	5	6
Повідомлення про вихід нових відеоматеріалів	MES-NEW-VID	Електронний лист, SMS	Щотижня	Термін видання - негайно, допустимий час затримки – 5 хвилин	Користувач
Повідомлення про зміну ролі	MES-ROL	Електронний лист, SMS	Раз на місяць	Термін видання - негайно, допустимий час затримки – 5 хвилин	Користувач
Трансляція відео користувачу	TRA-VID	Відео плеєр	Щоденно	Термін видання - негайно, допустимий час затримки – 15 секунд	Користувач
Обсяг прибутку	PRO-FIT	Таблиця	Раз на місяць	На протязі дня	Бухгалтер
Кількість проданих відеоматеріалів	QUAN-VID	Таблиця	Раз на місяць	На протязі дня	Бухгалтер, менеджер з продажу
Популярність відеоматеріалів	POP-VID	Таблиця	Раз на місяць	На протязі дня	Менеджер з закупівель

2.2.3. Вхідна інформація

Реєстрація користувача - це процес, коли новий користувач заповнює необхідну інформацію для створення облікового запису веб-платформи для організації роботи онлайн-кінотеатру. Це повідомлення передає дані, введені користувачем, в базу даних для подальшого створення облікового запису.

Заповнення інформації про користувача - це дія, коли користувач веб-системи оновлює свою особисту інформацію, таку як ім'я, адреса, контактні дані тощо. Це повідомлення оновлює інформацію про користувача в базі даних, забезпечуючи актуальні дані про нього.

Додавання нових категорій - це дія, коли адміністратор або відповідальна особа системи створюють нові категорії для класифікації відеоматеріалів, наприклад, "Комедія", "Драма", "Фантастика" тощо. Це повідомлення додає нові категорії в базу даних, полегшуючи класифікацію відеоматеріалів і пошук для користувачів.

Додавання нової продукції - це дія, коли адміністратор або відповідальна особа системи додають новий відеоматеріал, такий як фільм, серіал або відео курс. Це повідомлення додає новий запис про відеоматеріал в базу даних, зберігаючи його характеристики, такі як назва, жанр, тривалість, режисер тощо.

Придбання продукції - це дія, коли користувач здійснює покупку відеоматеріалу і отримує доступ до його перегляду або завантаження. Це повідомлення оновлює дані про купівлю в базі даних, реєструючи придбання відеоматеріалу і надаючи користувачу відповідний доступ до контенту.

Додавання нових менеджерів - це дія, коли адміністратор або відповідальна особа системи створюють облікові записи для нових менеджерів, які будуть відповідати за керування продажами, закупівлями або іншими аспектами системи. В рамках цього повідомлення до бази даних додаються нові облікові записи менеджерів. Це дає їм можливість отримувати доступ до веб-платформи та виконувати свої службові обов'язки.

Перелік та опис вхідних повідомлень системи наведено у табл. 2.2.

Таблиця 2.2 – Перелік та опис вхідних повідомлень системи

Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
Реєстрація користувача	REG	Форма на веб-сайті	Щоденно	Користувач
Заповнення інформації про користувача	MY-INFO	Форма на веб-сайті	Щоденно	Користувач
Додавання нових категорій	NEW-CAT	Форма на веб-сайті	Раз на місяць	Менеджер
Додавання нової продукції	NEW-PROD	Форма на веб-сайті	Щотижня	Менеджер
Придбання продукції	ORD	Форма на веб-сайті	Щоденно	Користувач
Додавання нових менеджерів	NEW_MNG	Форма на веб-сайті	Раз на місяць	Адміністратор

2.2.4. Математичне забезпечення та алгоритм функціонування системи

Математичний опис системи:

$$R_i = \frac{\sum_{k=0}^{N_i} R_{ik}}{N_i} \quad (1)$$

, де R_i – середня оцінка i -ї контенту (від 0 до 5);

N_i – кількість відгуків про i -ї контенту;

R_{ik} – оцінка i -ї контенту k -ого коментаря.

$$I_{jg} = \frac{T_{jg}}{T_{avgg}} \quad (2)$$

, де I_{jg} – індекс зацікавленості j -го користувача в g -ній категорії;

T_{jg} – сумарний час проведений j -м користувачем за переглядом g -го фільму;

T_{avgg} – середній сумарний час проведений користувачами за переглядом g -го фільму.

Алгоритм розв'язання задачі подано на рис 2.11.

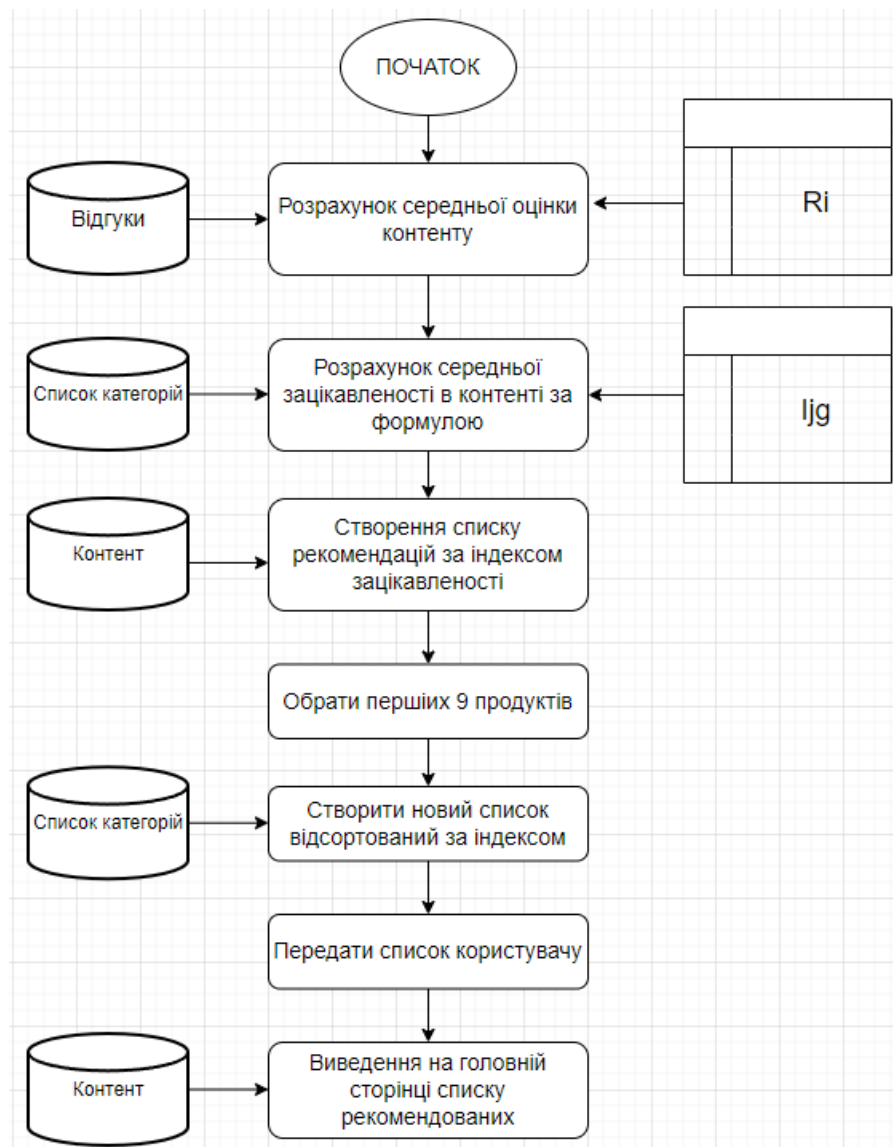


Рисунок 2.11 – Алгоритм розв'язання задачі з пошуку рекомендованих для користувача ігор

Діаграма взаємодії описує два процеси: клієнта та адміністратора. Процес адміністратора, у свою чергу, поділяється на два підпроцеси: менеджера та адміністратора. Процес клієнта включає два підпроцеси: авторизацію та вибір товару для покупки. Важливо зазначити, що вибір товару може повторюватися. На діаграмі також представлені послідовні дії та бази даних, які використовуються в кожному з процесів..

Після вдалої оплати клієнт переходить до бібліотеки для перегляду придбаного контенту. Якщо на етапі оплати або на етапі переходу до бібліотеки виникнуть проблеми він може звернутись до технічної підтримки для вирішення своїх проблем. Після звернення менеджер спробує вирішити цю проблему за допомогою внесення змін до бази даних якщо в нього це не вийде він звертається до адміністратора який спробує вирішити цю проблему редагуючи базу даних або редагуючи код back end та front end частини інформаційної системи. На рис.2.12 представлена діаграма взаємодії.

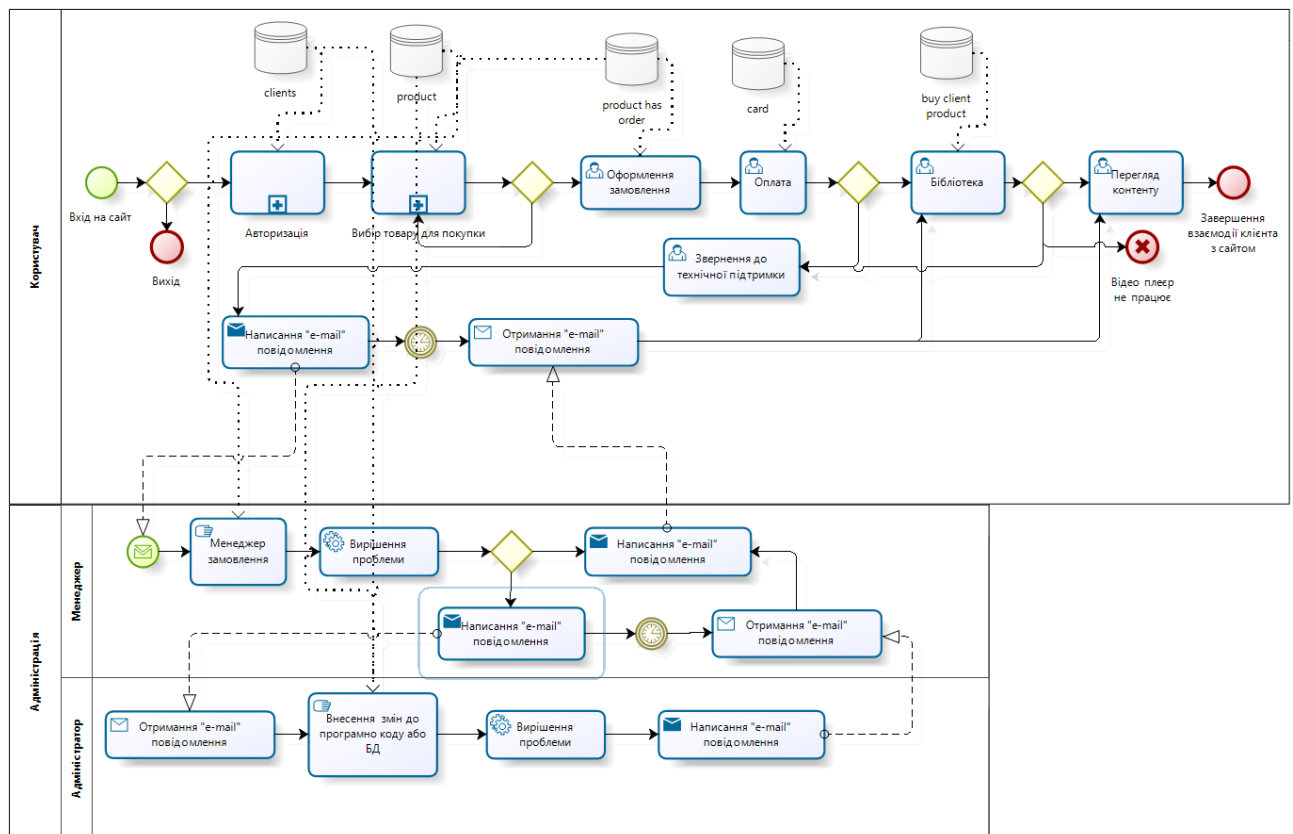


Рисунок 2.12 – Діаграма взаємодії

Діаграма відображає процес входу користувачів у систему. Користувач може увійти за допомогою існуючого облікового запису або створити новий. Після успішного входу в систему користувач отримує відповідні права відповідно до своєї ролі. Діаграма також враховує можливі помилки під час входу або реєстрації, такі як неіснуючі дані (наприклад, неправильний логін або пароль), а також можливість використання вже зайнятих даних для реєстрації. На рис.2.13 представлена діаграма процесу авторизації.



Рисунок 2.13 – Діаграма процесу авторизації

Клієнт, перейшовши на головну сторінку вже побачить найпопулярніші товари після чого в нього є два варіанти знайти те що йому потрібно перший це ввести у поле пошуку назву того що він шукає або ж за допомогою відсортувати контент за категорією після чого обирається потрібний контент і відкривається нова сторінка з цим контентом для його подальшої купівлі. На рис.2.14 представлена діаграма процесу вибору товару для покупки.

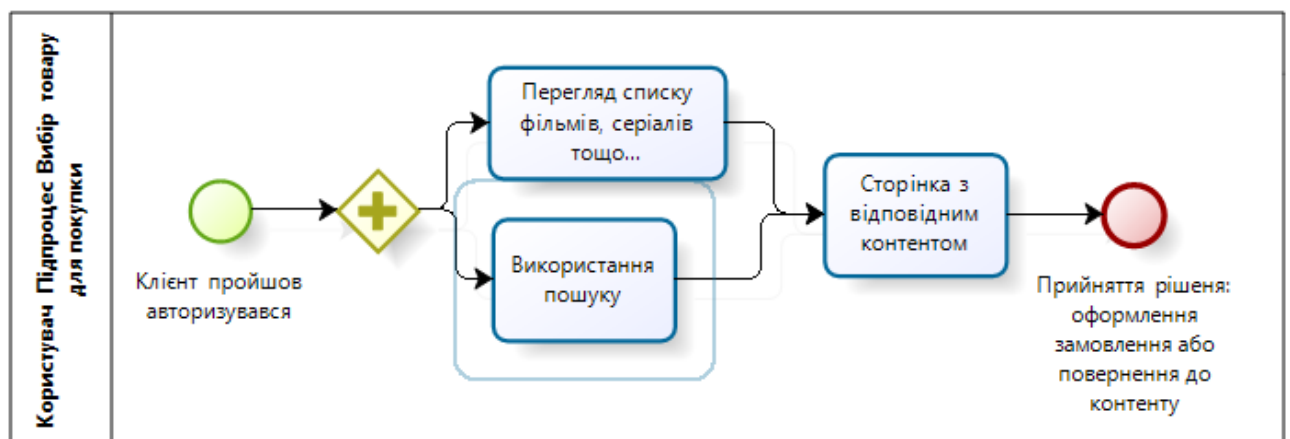


Рисунок 2.14 – Діаграма процесу вибору товару для покупки

На діаграмі діалогів продемонстровано, що адміністратор поєднаний з менеджером а менеджер в свою чергу з клієнтом. Отже якщо у клієнта з'являються проблеми він спочатку шукає їх вирішення у довідковому центрі, якщо він не знаходить вирішення своєї проблеми там він звертається до співробітників (менеджер та адміністратор) якщо і там не змогли вирішити проблеми то для вирішення будуть залучатися експерти. На рис.2.15 представлена діаграма діалогів.

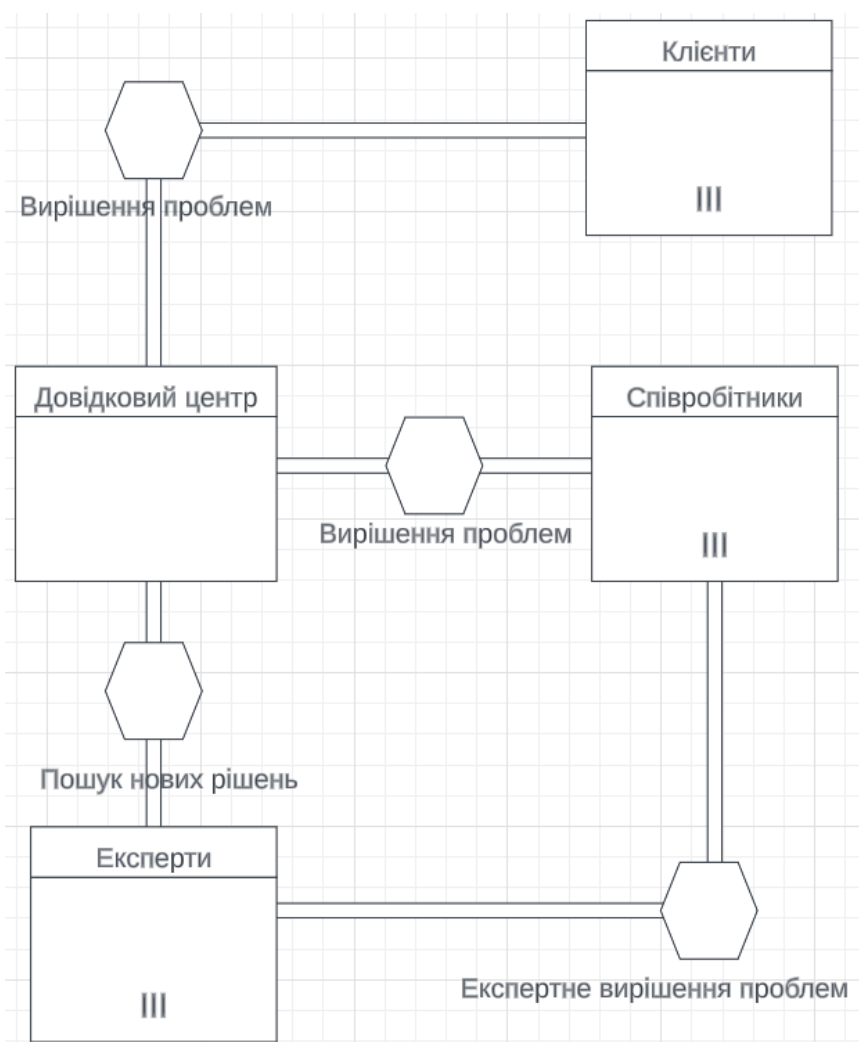


Рисунок 2.15 – Діаграма діалогів

2.3. Моделювання інформаційної підсистеми

2.3.1. Моделювання поведінки системи

За допомогою UML-діаграми прецедентів можна чітко окреслити ролі користувачів та функціональні можливості системи на рис. 2.16.

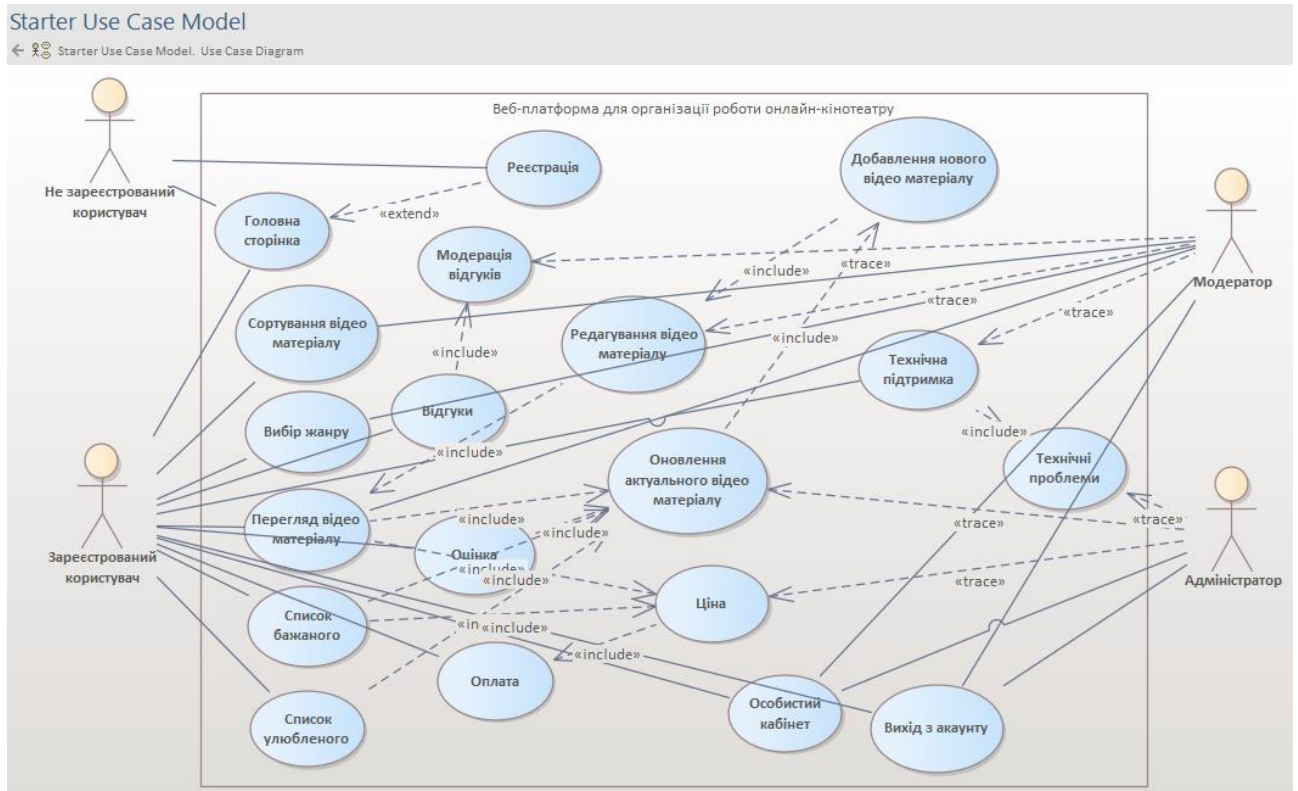


Рисунок 2.16 – Діаграма прецедентів

Послідовність виклику та взаємодії користувача із системою розпочинається з реєстрації користувача на сайті потім входить в особистий кабінет далі через параметри пошуку шукає фільм після чого бачить список відео матеріалу після чого користувач додає фільми що сподобалися до списку бажаного і потім додає платіжну картку та оплачує відео матеріал, що сподобався в результаті чого цей відео матеріал додається на акаунт користувача як придбаний і також користувач отримає квитанцію про оплату після чого він зможе переглянути цей фільм та додати його до списку улюбленого на рис. 2.17 показано UML-діаграму послідовності.

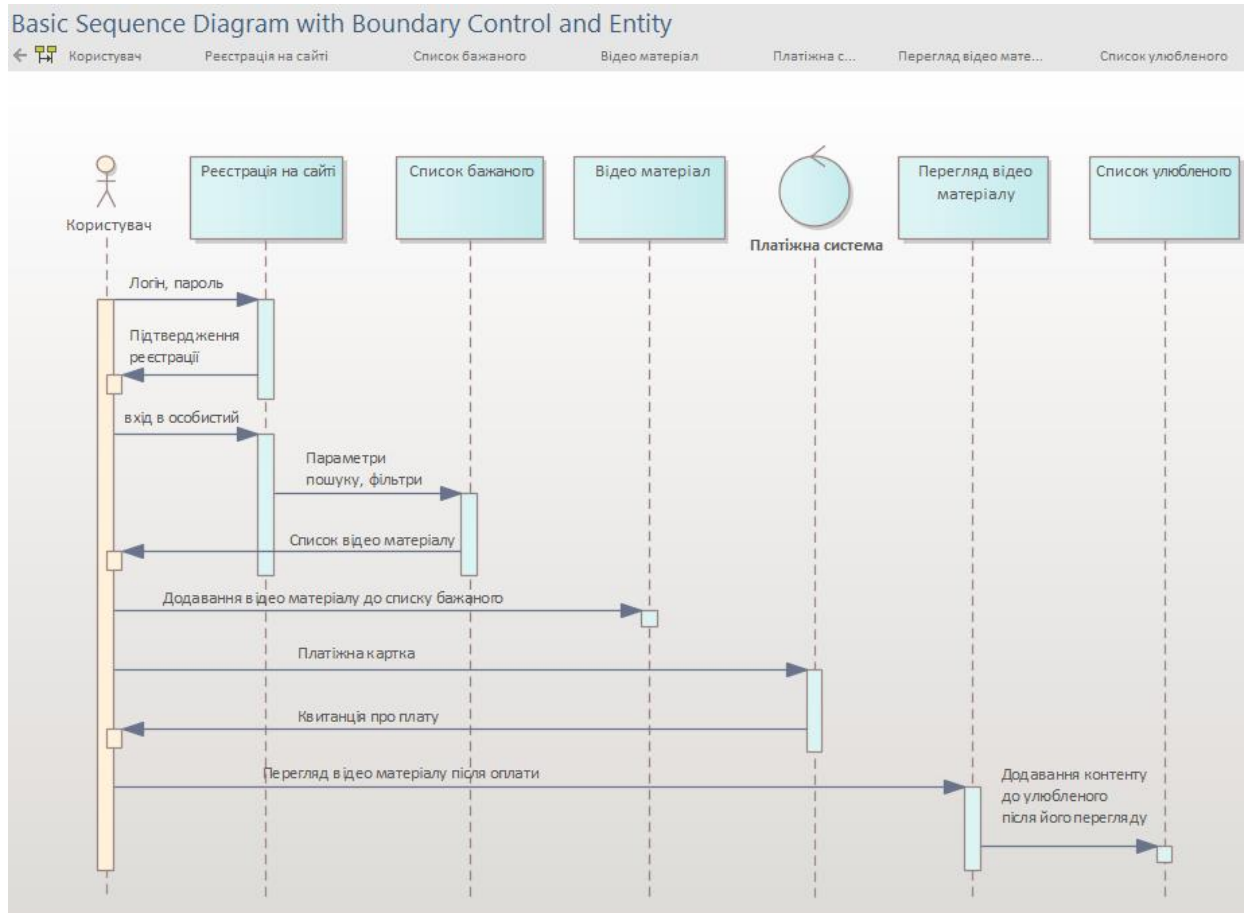


Рисунок 2.17 – Діаграми послідовності

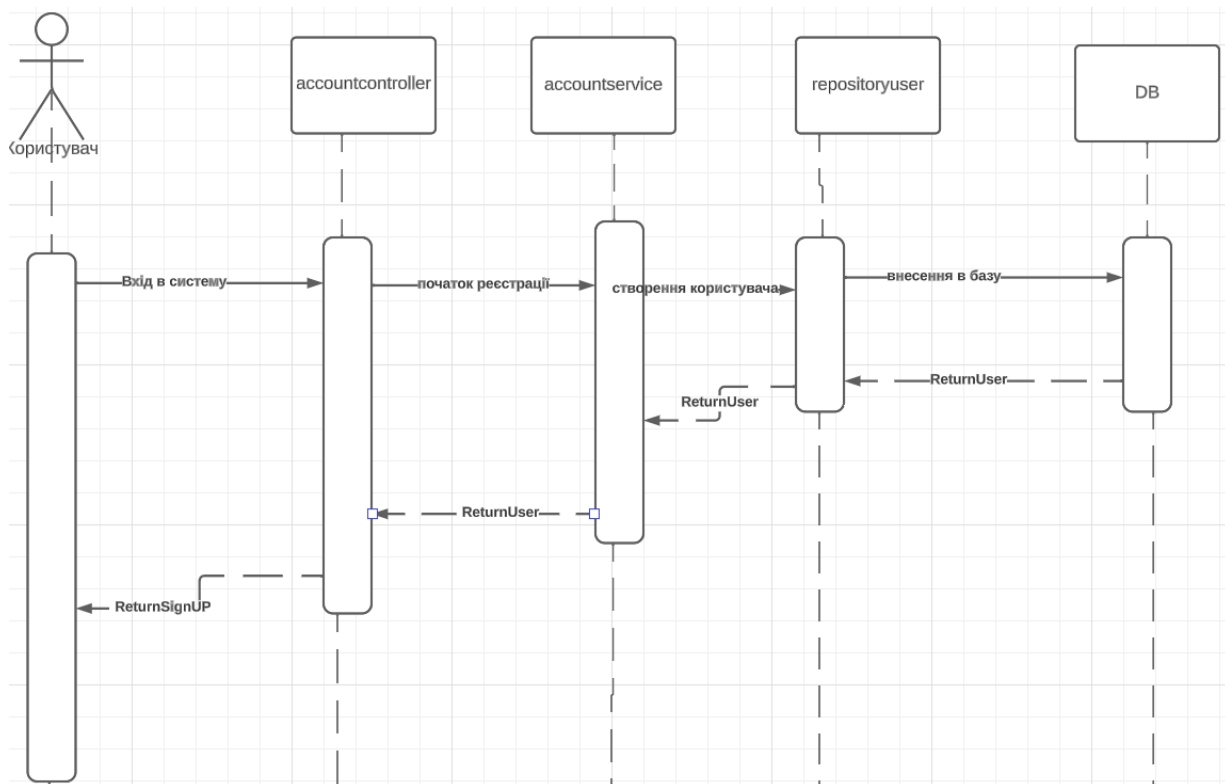


Рисунок 2.18 – Діаграми послідовності реєстрація користувача в системі

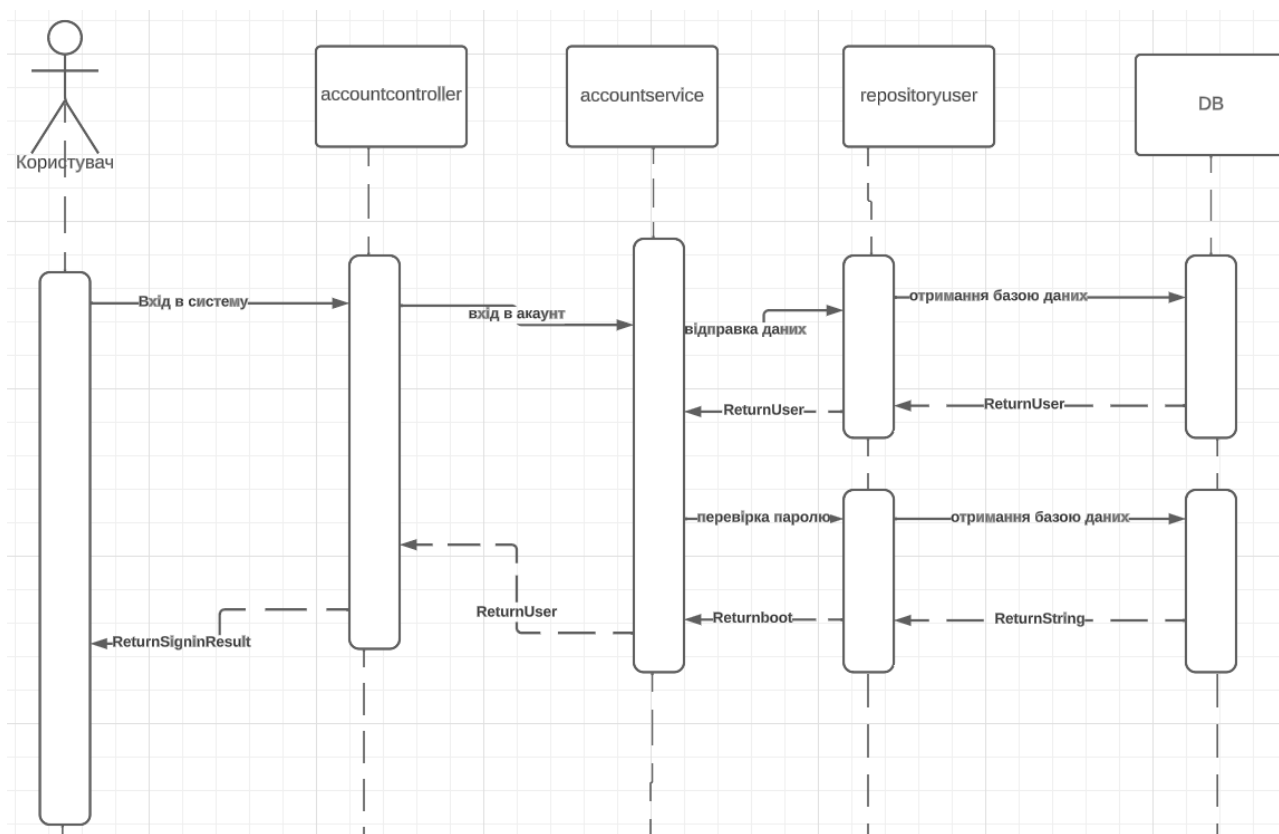


Рисунок 2.19 – Діаграми послідовності вхід користувача на сайт

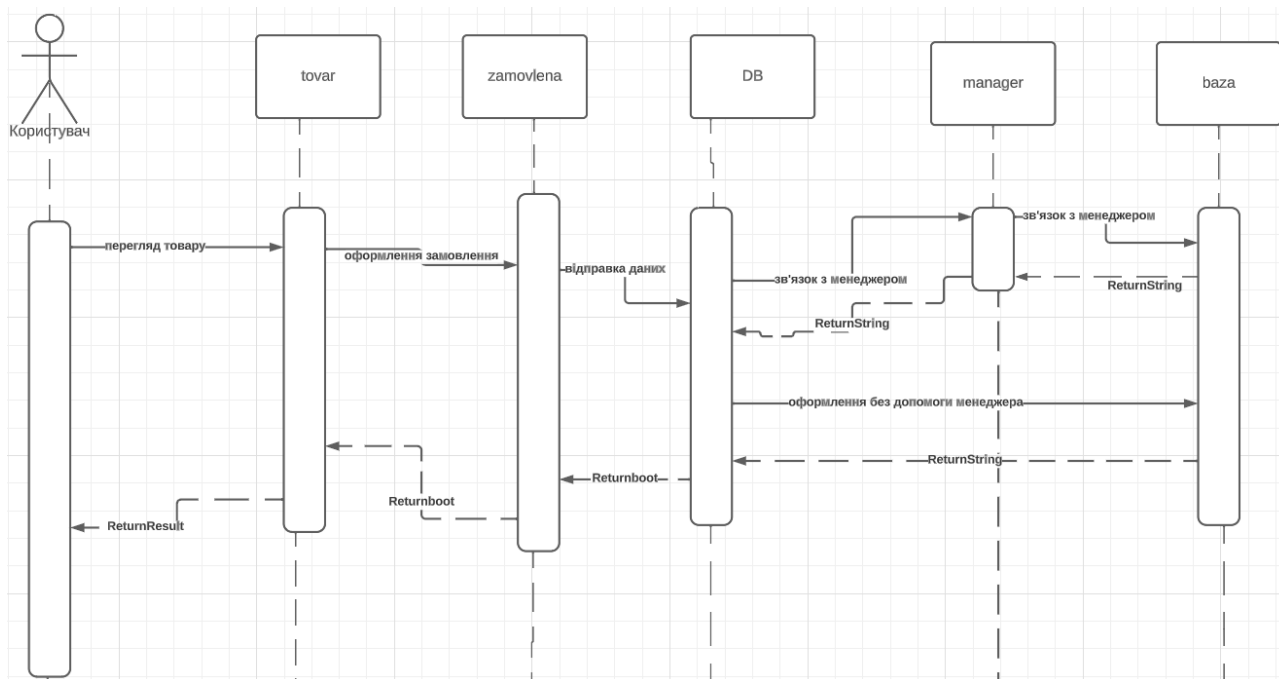


Рисунок 2.20 – Діаграми послідовності оформлення замовлення користувачем

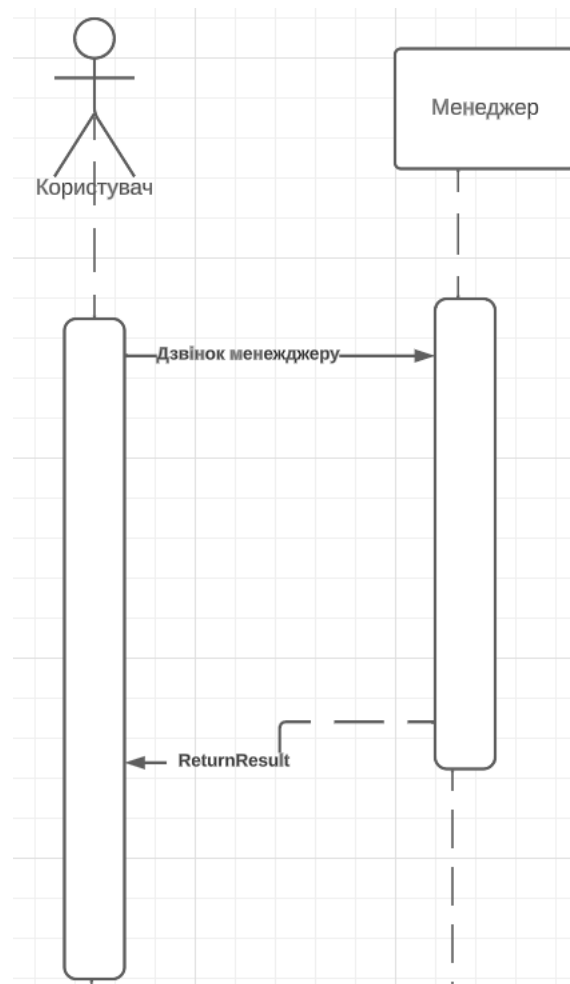


Рисунок 2.21 – Діаграми послідовності зв'язок для консультації з менеджером

2.3.2. Моделювання структури системи

Веб-платформа для організації роботи онлайн-кінотеатру має презентацію, бізнес-логіку, аутентифікацію, доступ до даних, Payment Gateway (оплата) та система управлінням відео матеріалом, що представлені на рис. 2.22.

Презентація описує взаємодію з користувачем тобто інтерфейси взаємодії на сайті та забезпечення безпеки даних веб-системи.

Бізнес-логіка описує сервіси та їх компоненти, валідацію та обробку даних, тестування, інтеграцію з зовнішніми системами, механізмами обробки даних, управління обробкою даних та реалізацію бізнес-логіки.

Аутентифікація описує форми входу та реєстрацію, зберігання та управління даними, авторизацію та керування доступом, захист від атак та небажаних дій, інтеграцію з соціальними мережами і логування та аудит.

Доступ до даних описує систему управління базами даних, моделі даних, механізм звертання до даних, оптимізація та індексування структури бази даних, захист даних їх резервування, моніторинг і аналіз продуктивності.

Payment Gateway (Оплата) описує інтеграцію з платіжними системами форми та механізми оплати, безпека та захист даних, обробка транзакцій, обробка повернень та скасування, відслідковування та звітність та підтримка міжнародних операцій.

Система управління відео матеріалом описує управління фільмами та відео, управління контентом статичних сторінок, управління контентом для користувача, розклад сеансів та управління місяцями, мультимедійним контентом, пошук фільтрацій контенту та мультиплатформеність та адаптивність.

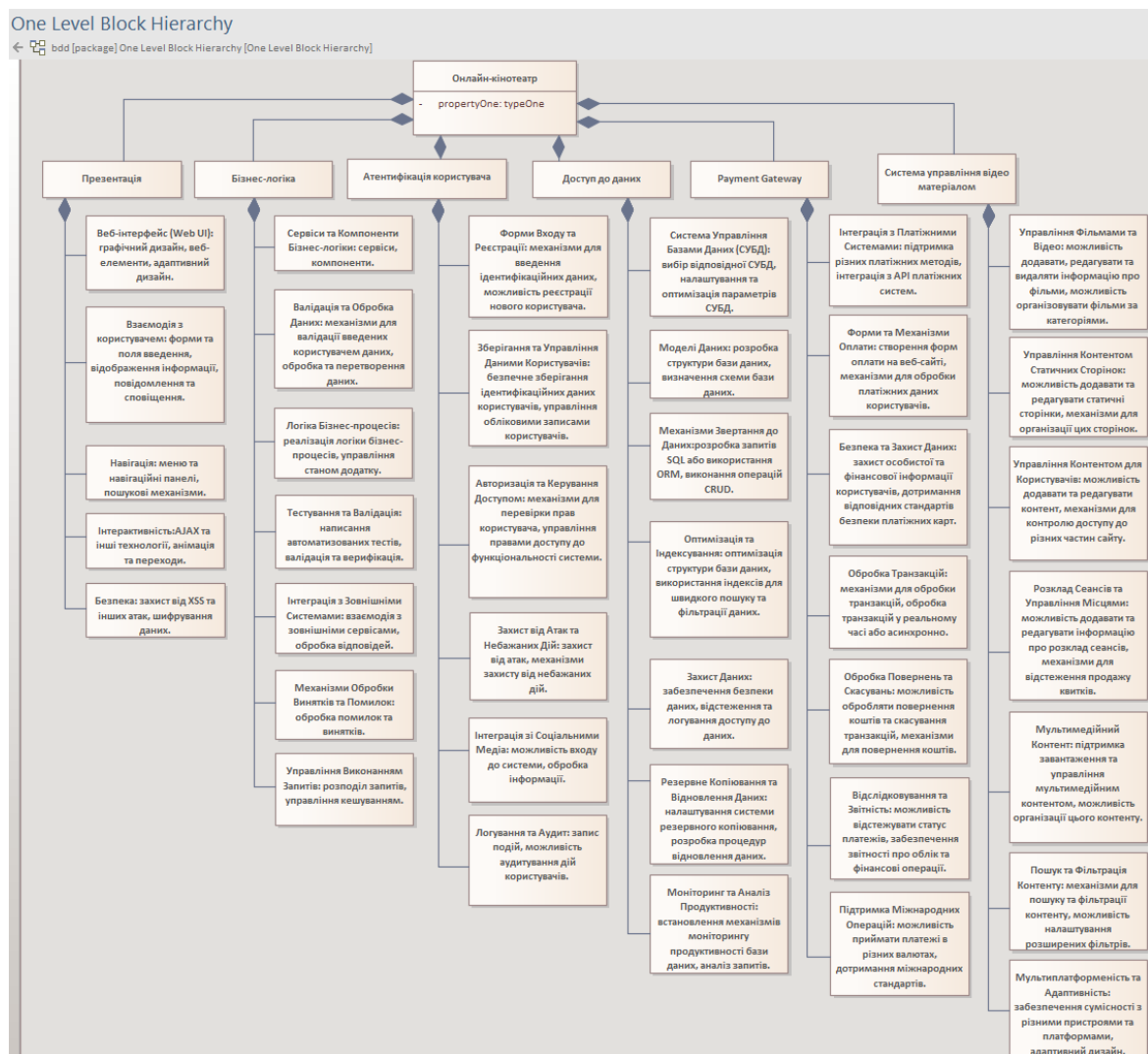


Рисунок 2.22 – Діаграма визначення блоків

Тепер опишемо купівлю відео матеріалу, для цього користувач обирає відео матеріал на новій сторінці та підтверджує його придбання після чого обирає спосіб оплати та вводить платіжні данні картки та підтверджує оплату після чого отримує чек та можливість переглядати придбаний відео матеріал. Цей опис діаграми внутрішніх блоків на рис. 2.23.

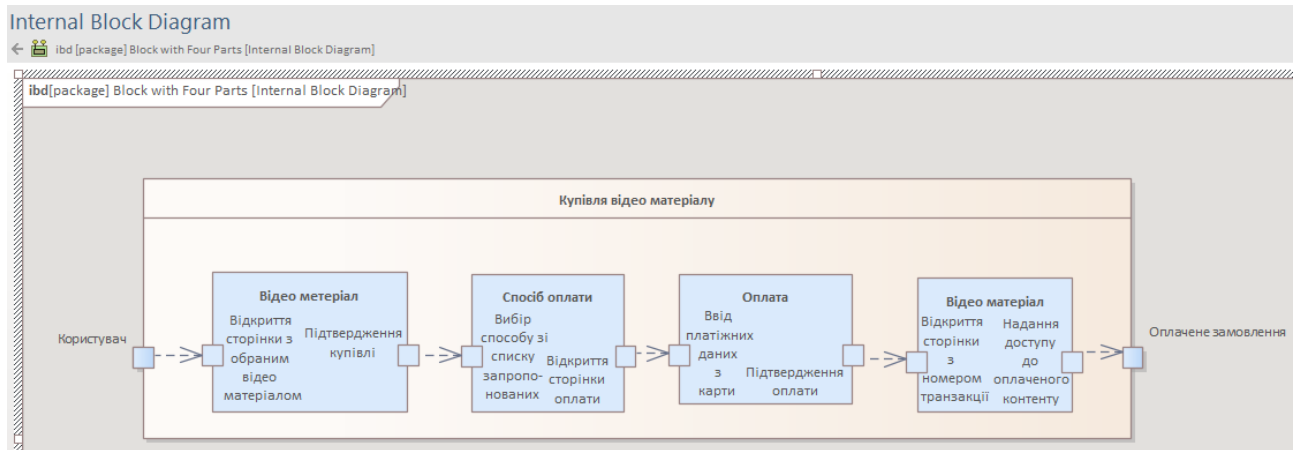


Рисунок 2.23 – Діаграма внутрішніх блоків

Було побудовано діаграму класів продемонстровано на рис 2.24 серверної частини системи. Згідно з MVC, діаграма включає моделі, контролери та сервіси, але відсутні представлення (View). Класи на діаграмі можна розділити на три основні групи:

1. "Дані" - ця частина відповідає за зберігання даних;
2. "Бізнес-логіка" - в цій частині розміщені класи, що відповідають за обчислення та виконання алгоритмів обробки даних;
3. "Інтерфейси" - ця група включає класи, які отримують інформацію з системи та передають її користувачам (клієнтам).

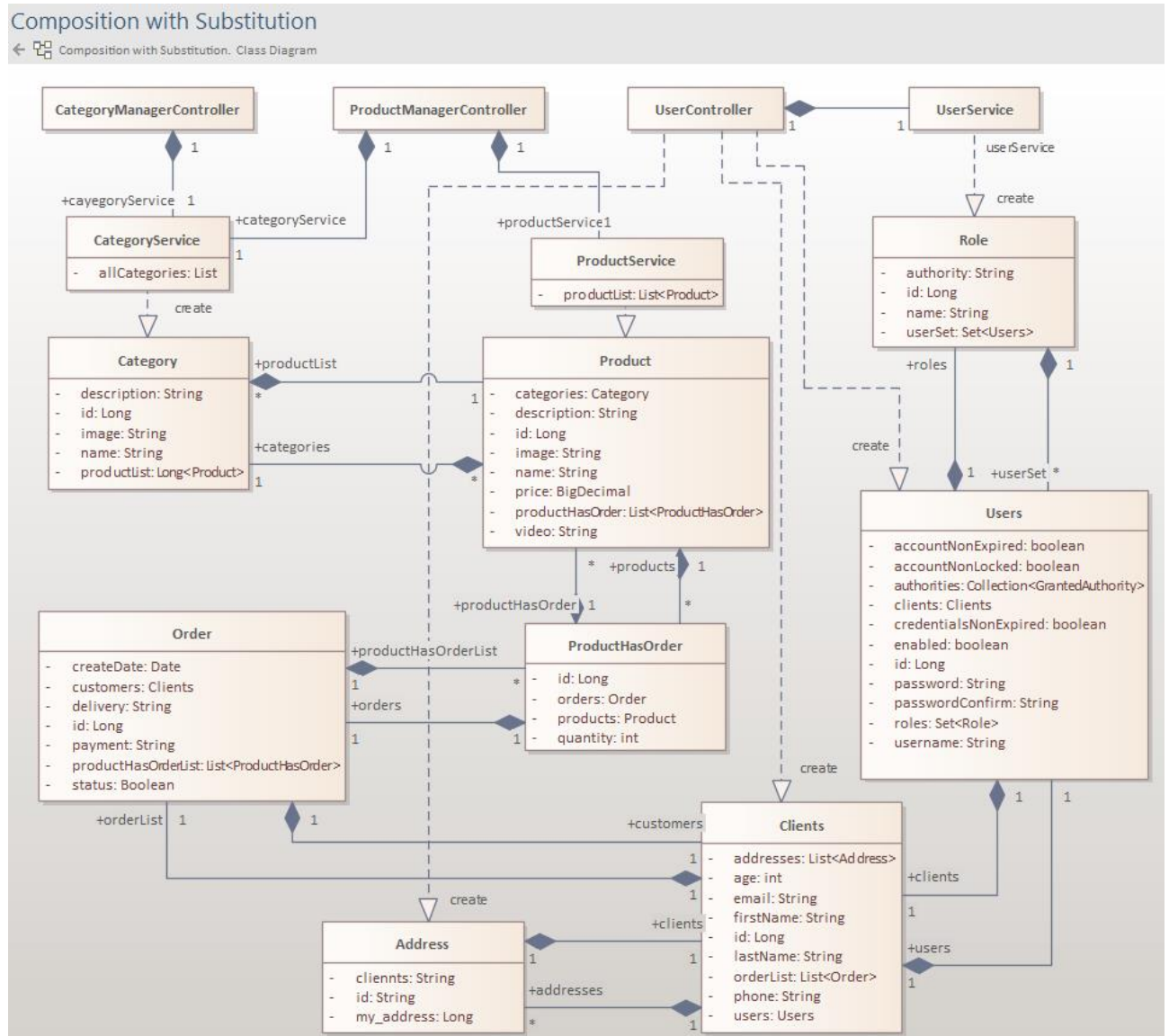


Рисунок 2.24 – Діаграма класів

2.3.3. Розробка тест-кейсів

Розроблені тест-кейси представлені на рис. 2.25 – 2.27.

TESTING
 Standard Unit test. Вхід на головну сторінку авторизованого користувача Description 1. Відкрити веб-систему "Онлайн-кінотеатр". 2. Натиснути кнопку "Вхід". 3. Ввести коректний логін та пароль. 4. Натиснути кнопку "Увійти". Input Посилання на веб-сайт: http://localhost:8080/index2 Acceptance Criteria Results Відкривається головна сторінка сайту для зареєстрованого користувача: http://localhost:8080/index2 07.04.2024 - Pass Run by: yboxuser Checked by: yboxuser [Last run Pass at 07.04.2024 by yboxuser and checked by yboxuser.]
 Standard Unit test. Відгук користувача про фільм Description 1. Авторизація на сайті. 2. Натиснути на випадковий фільм із запропонованих на головній сторінці. 3. Натиснути на поле для написання коментаря. 4. Написати коментар. 5. Натиснути на кнопку "Залишити відгук". 6. Оновити сторінку натиснувши на кнопку "Оновлення сторінки". Input Login: "User" Password: "12345" Acceptance Criteria Results Коментар залишається у списку коментарів. 11.04.2024 - Pass Run by: yboxuser Checked by: yboxuser [Last run Pass at 11.04.2024 by yboxuser and checked by yboxuser.]

Рисунок 2.25 – Тестові сценарії «Вхід на головну сторінку авторизованого користувача» та «Відгук користувача про фільм»

Standard Unit test. Додавання фільму до списку бажаного Description 1. Авторизуватися на сайті. 2. Натиснути на випадковий фільм. 3. Натиснути на зображення зірочки. 4. Натиснути на кнопку "Список бажаного". 5. Натиснути на назву фільму. Input Login: "User" Password: "12345" Acceptance Criteria Results Відкриється сторінка з фільмом що був доданий до списку бажаного. 07.04.2024 - Pass Run by: vboxuser Checked by: vboxuser [Last run Pass at 07.04.2024 by vboxuser and checked by vboxuser.]
Standard Unit test. Покупка фільму Description 1. Авторизація на сайті. 2. Натиснути на випадковий фільм із запропонованих варіантів на головній сторінці. 3. Натиснути на кнопку "Придбати". 4. Натиснути на список банківських карток. 5. Натиснути на будь яку з запропонованих карток. 6. Натиснути кнопку оплати. 7. Натиснути на "Бібліотека". 8. Перевірити чи співпадає назва фільму з оплаченим фільмом. Input Login: "User" Password: "12345" Acceptance Criteria Results Придбаний фільм додався до бібліотеки. 11.04.2024 - Pass Run by: vboxuser Checked by: vboxuser [Last run Pass at 11.04.2024 by vboxuser and checked by vboxuser.]

Рисунок 2.26 – Тестові сценарії «Додавання фільму до списку бажаного» та «Покупка фільму»

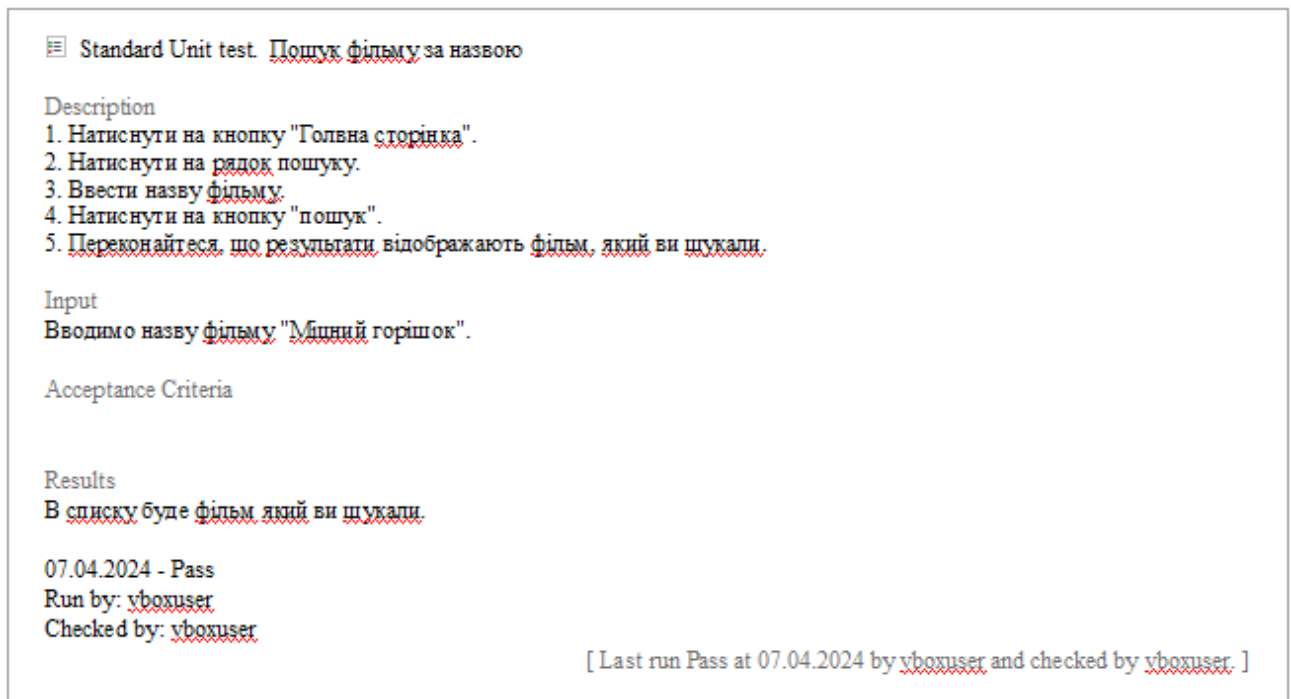


Рисунок 2.27 – Тестовий сценарій «Пошук фільму за назвою»

2.3.4. Трасування вимог до системи

Тепер розглянемо діаграму трасування де з правої сторони представлено абстрактний клас моделі користувача тобто там вказано всі типи даних користувачів, таких як, не зареєстрований користувач, зареєстрований користувач, модератор та адміністратор, ці дані використовуються у веб-системі, по центру прецеденти, що є користувацькими інтерфейсами і до яких також додано тест-кейси для перевірки роботи веб-системи а з лівої сторони представлені запити до веб-системи в них вказано які дані куди передаються і також можна зрозуміти послідовність запитів тобто якщо викликати один випадковий запит то можна зрозуміти який буде наступним і так дійти до головних запитів до веб-системи тому на рис. 2.28 представлено діаграму трасування а на рис. 2.29 – 2.30 представлено матриці взаємозв'язків, що доповнюють діаграму трасування.

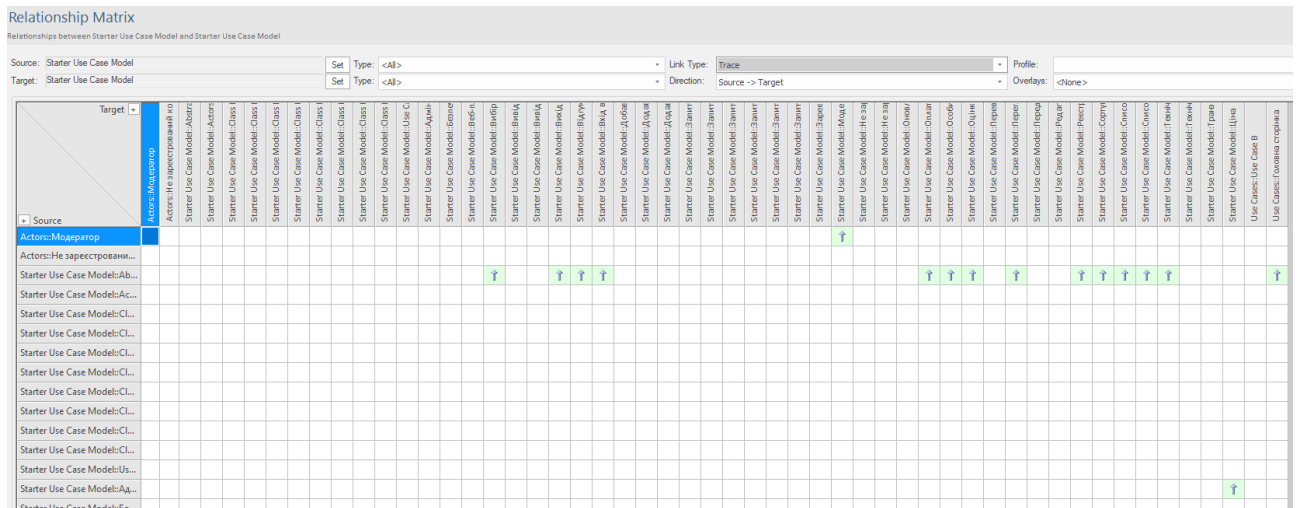


Рисунок 2.30 – Матриця взаємозв’язків для типу «Trace»

2.4. Моделювання користувацького інтерфейсу

Для демонстрації користувацьких інтерфейсів було створено прототип веб-платформи у векторному онлайн-сервісі Figma. На рис.2.31 представлено головний проект який також можна переглянути за посиланням:

<https://www.figma.com/proto/VmUoksgNXs7ZswxQqYg6l2/%D0%9A%D0%9F-%D0%91%D0%B0%D1%80%D0%B0%D0%B1%D0%B0%D1%88-%D0%9E.%D0%9C.-%D0%86%D0%9D-404?page-id=0%3A1&type=design&node-id=1-4&viewport=341%2C518%2C0.29&t=1MyG41Yk8qu7X3L9-1&scaling=scale-down&starting-point-node-id=1%3A4&mode=design>

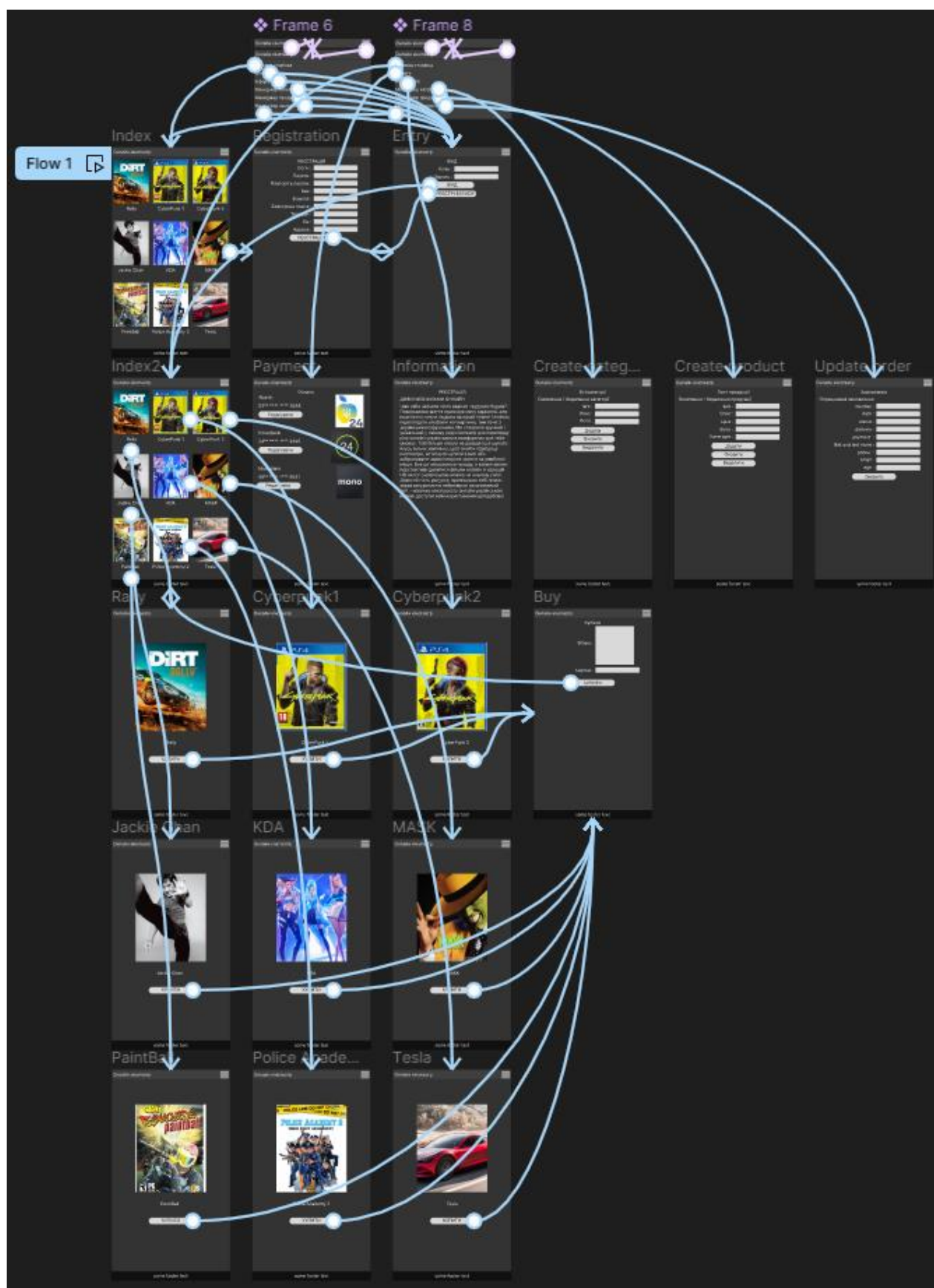


Рисунок 2.31 – Головний проект на якому демонструються зв'язки між полотнами тобто переходи з однієї сторінки на іншу

Тепер розглянемо прототипи сторінок веб-платформи для організації роботи онлайн-кінотеатру, що представлені на рис. 2.32 – 2.48.

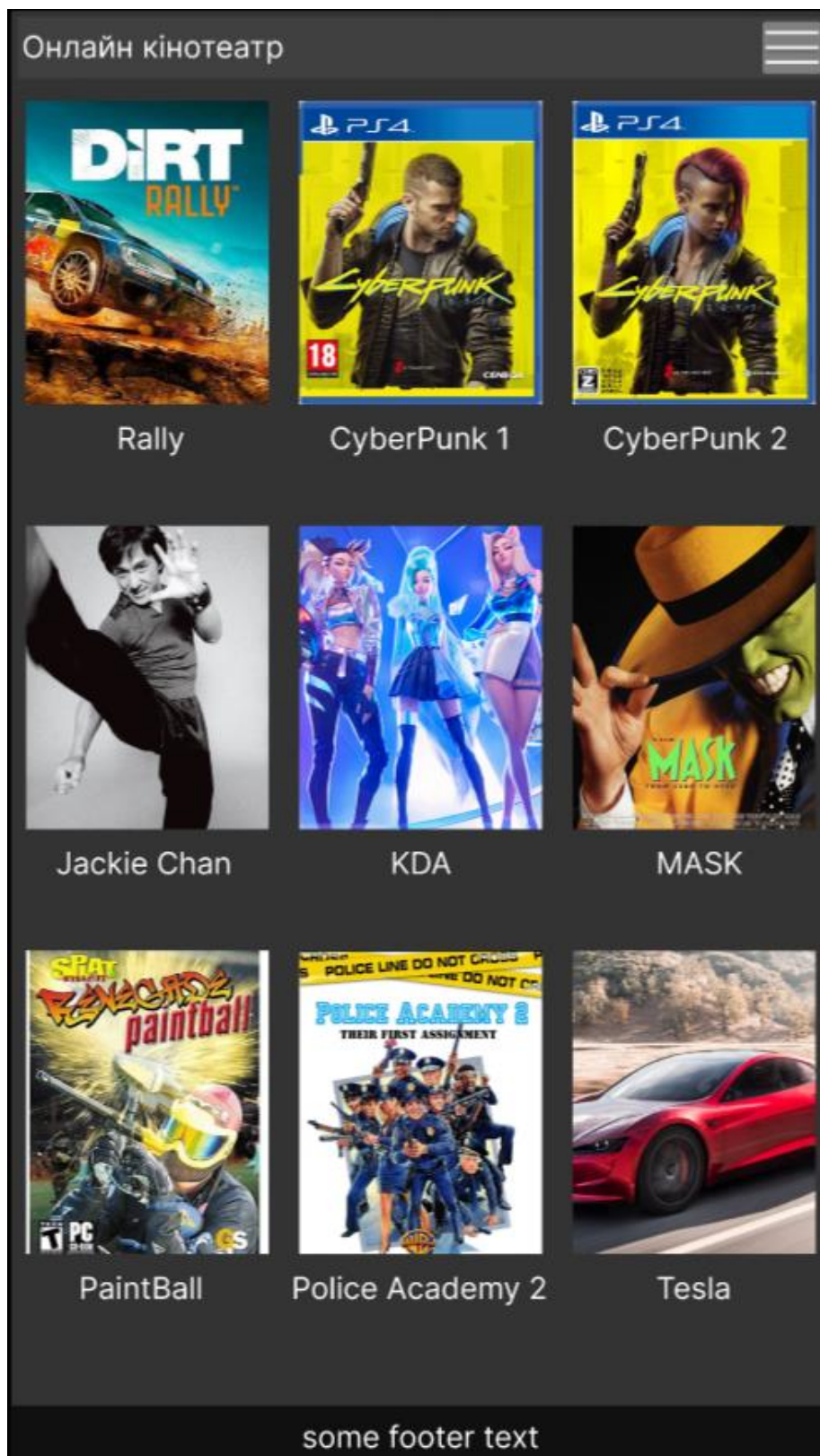


Рисунок 2.32 – Полотно Index, що є головною сторінкою для незареєстрованого користувача

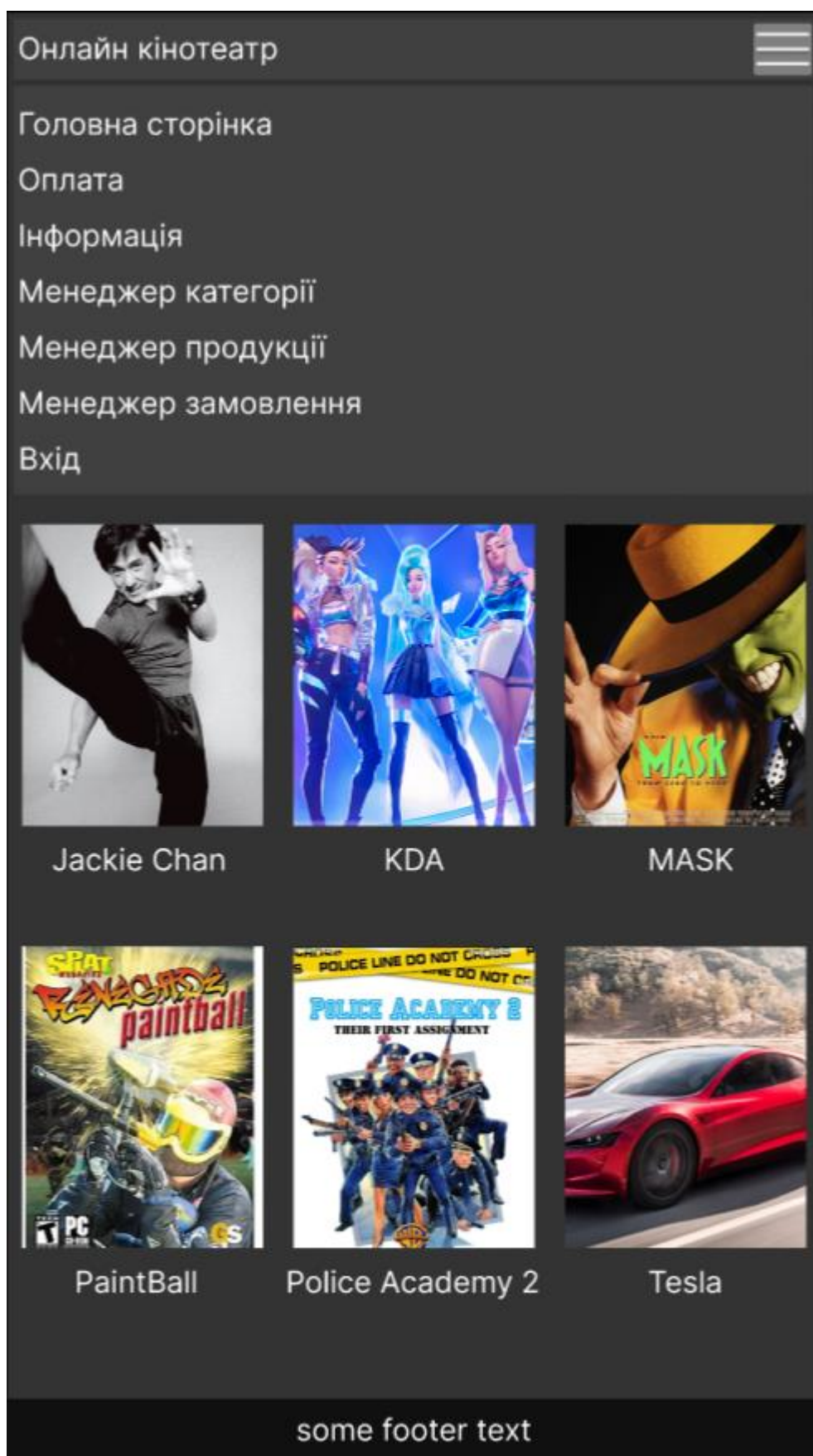
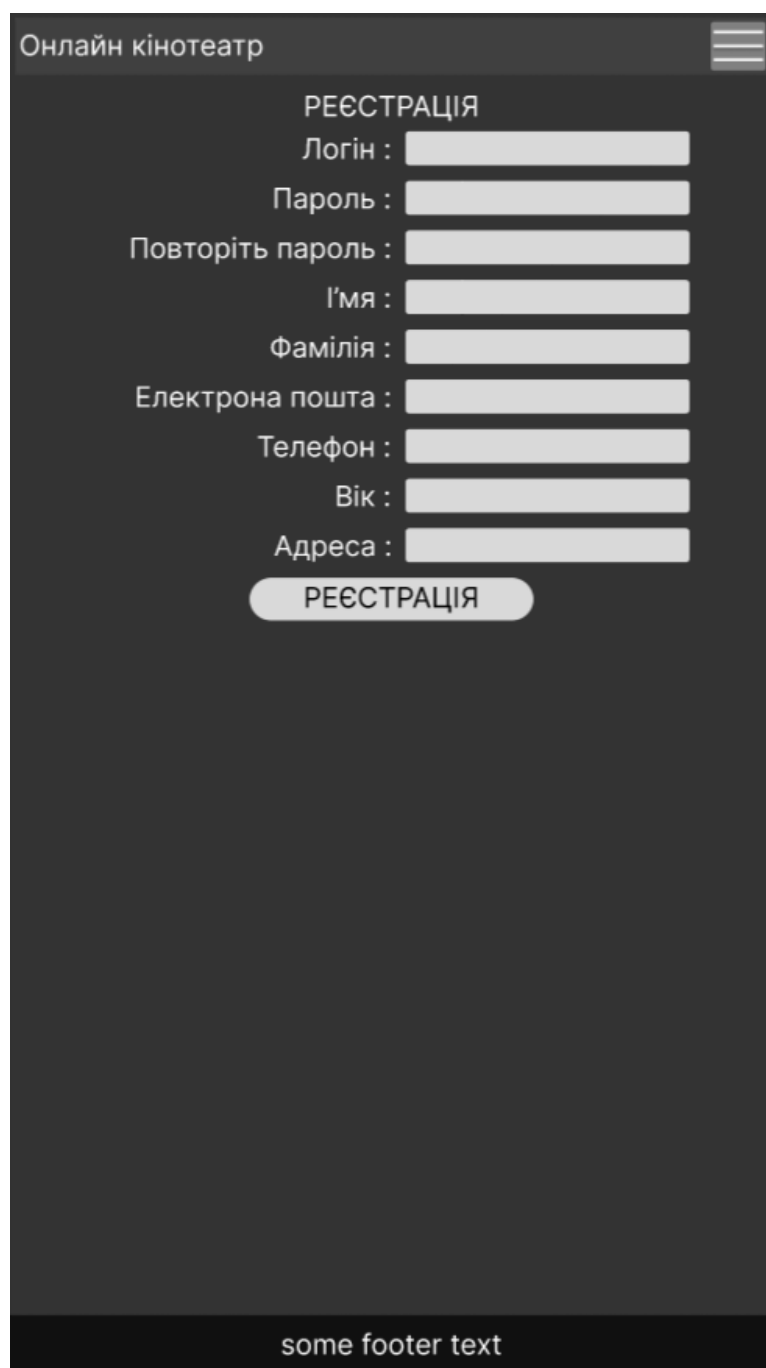


Рисунок 2.33 – Полотно index1 на якому відкритий список посилань на веб-сторінки

На головній сторінці знаходяться список фільмів доступних для перегляду і також тут можна відкрити список головного меню з переходом на такі сторінки як Оплата, Інформація, Менеджер категорії, Менеджер продукції та Менеджер замовлення але так як ці сторінки не для зареєстрованого користувача буде переадресація на сторінку реєстрації але якщо натиснути на кнопку входу відкриється сторінка з вводом логіну та паролю також якщо натиснути на Головна сторінка то відкриється головна сторінка.



Онлайн кінотеатр

РЕЄСТРАЦІЯ

Логін :

Пароль :

Повторіть пароль :

Ім'я :

Фамілія :

Електронна пошта :

Телефон :

Вік :

Адреса :

РЕЄСТРАЦІЯ

some footer text

Рисунок 2.34 – Полотно Registration, що є сторінкою реєстрації

Онлайн кінотеатр

ВХІД

Логін :

Пароль :

ВХІД

ЗАРЕЄСТРУВАТИСЯ

some footer text

The image shows a dark-themed login interface. At the top left is the title 'Онлайн кінотеатр'. At the top right is a hamburger menu icon. Below the title is the word 'ВХІД'. Then there are two input fields: 'Логін :' followed by a text input box, and 'Пароль :' followed by a password input box. Below these are two buttons: 'ВХІД' and 'ЗАРЕЄСТРУВАТИСЯ'. At the very bottom is a footer area with the text 'some footer text'.

Рисунок 2.35 – Полотно Entry, що є сторінкою входу

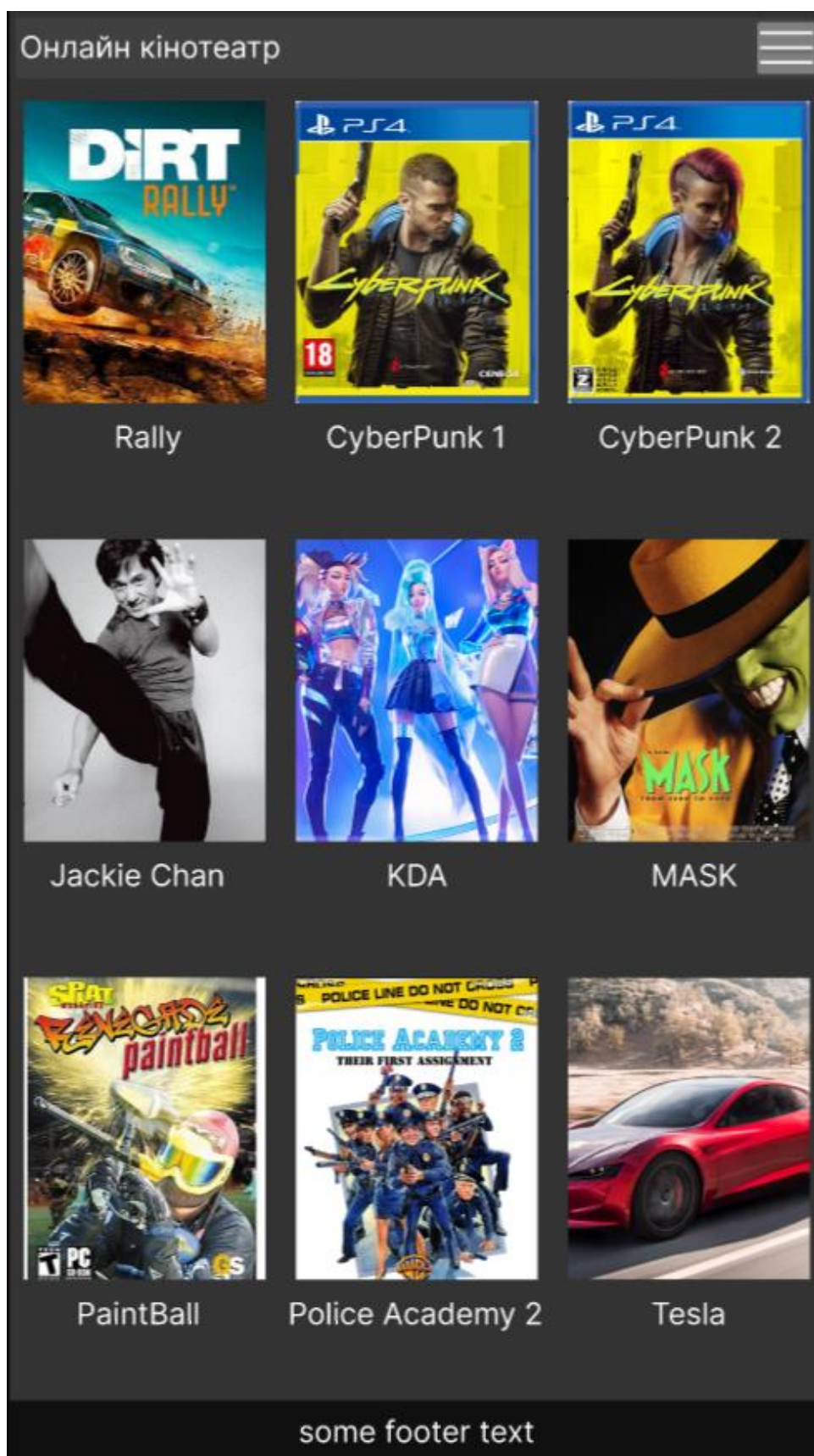


Рисунок 2.36 – Полотно Index2, що є головною сторінкою для зареєстрованого користувача тут також при натисканні на фільм відкриється сторінка для його перегляду

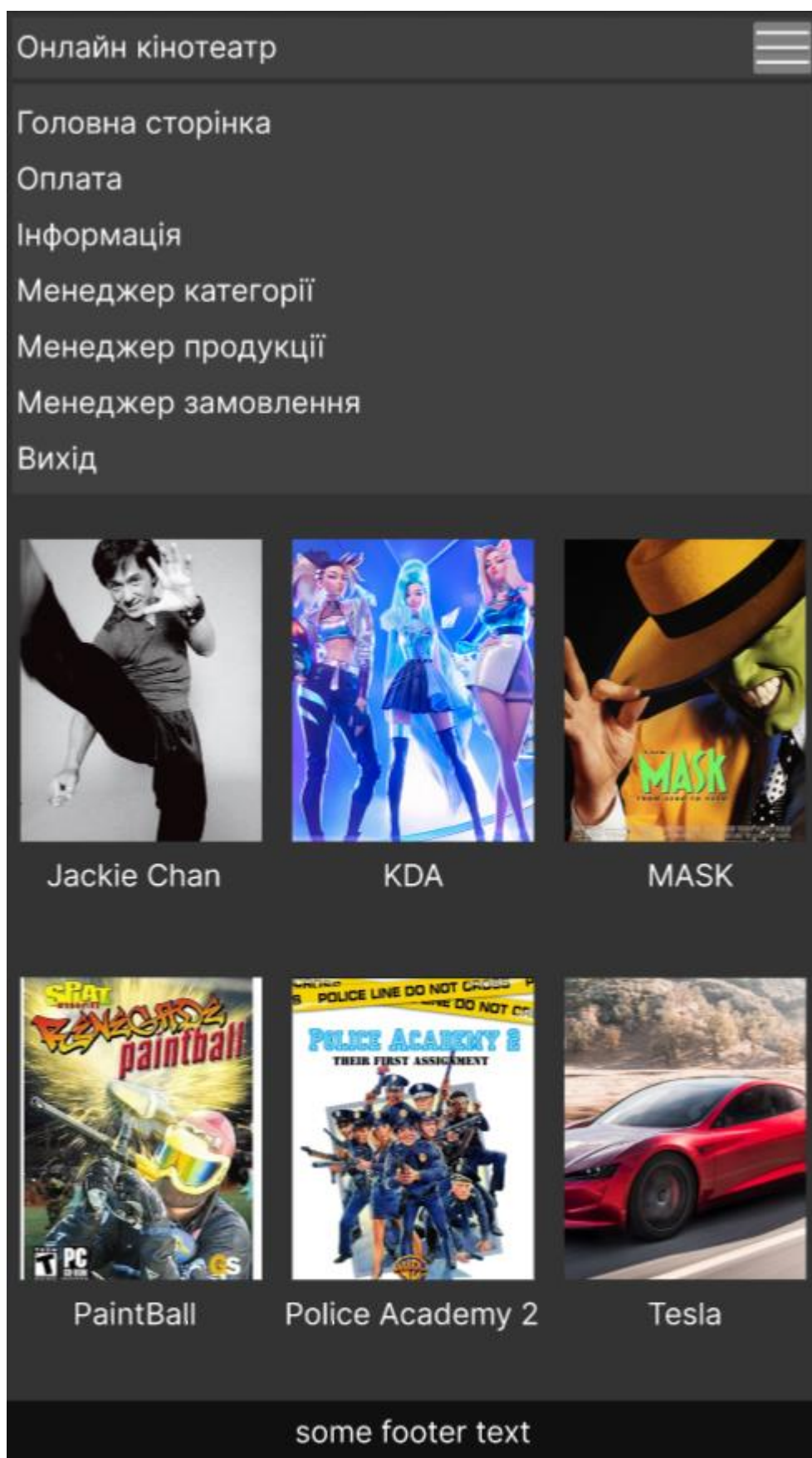


Рисунок 2.37 – Полотно Index2 на якому відкритий змінений список посилань на веб-сторінки

На головній сторінці знаходяться список фільмів доступних для перегляду і також тут можна відкрити список головного меню з переходом на такі сторінки як Головна сторінка Оплата, Інформація, Менеджер категорії, Менеджер продукції, Менеджер та Вхід.

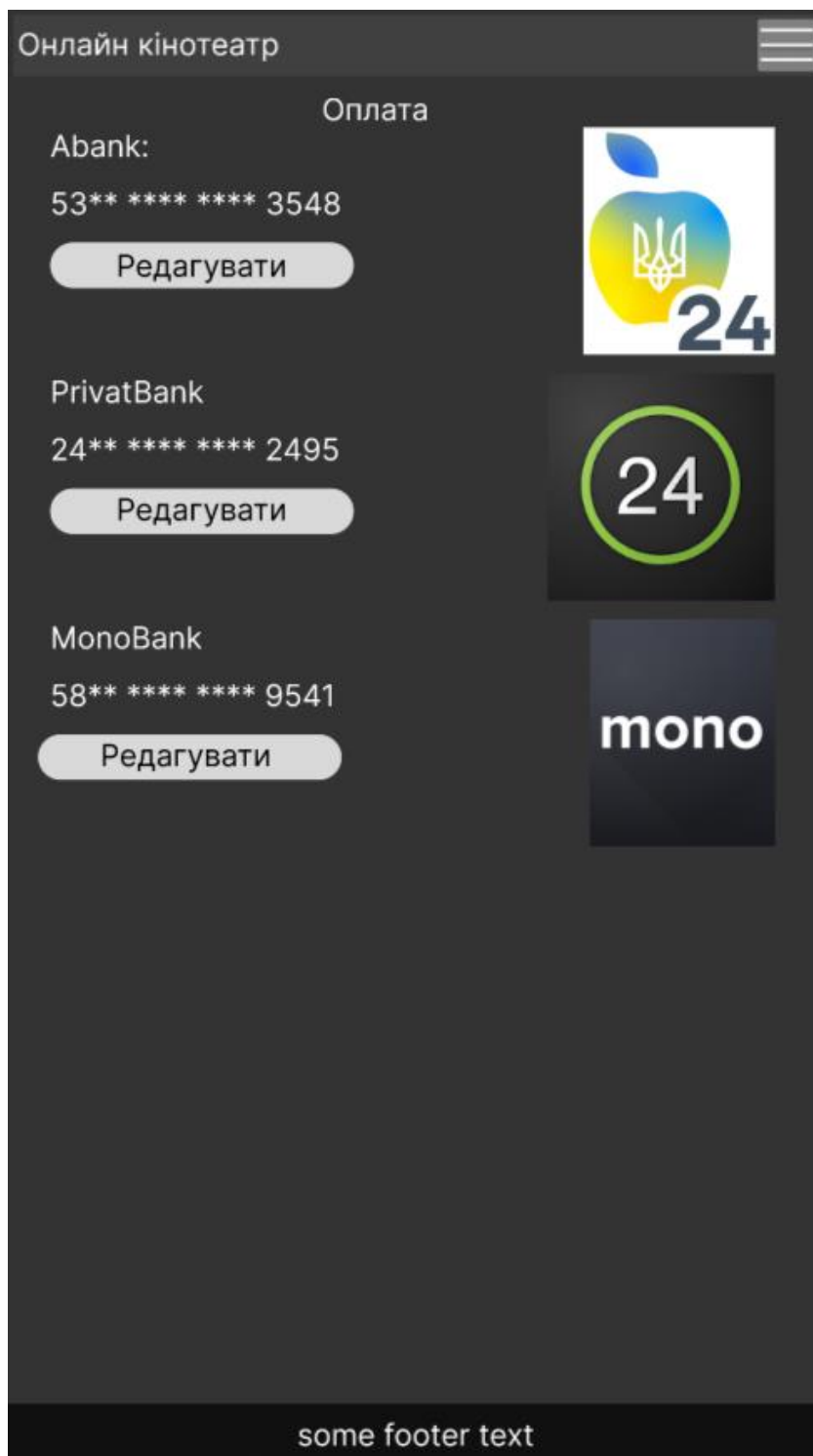


Рисунок 2.38 – Полотно Payment, що є сторінкою для оплати, де можна відредагувати дані карток банків для оплати

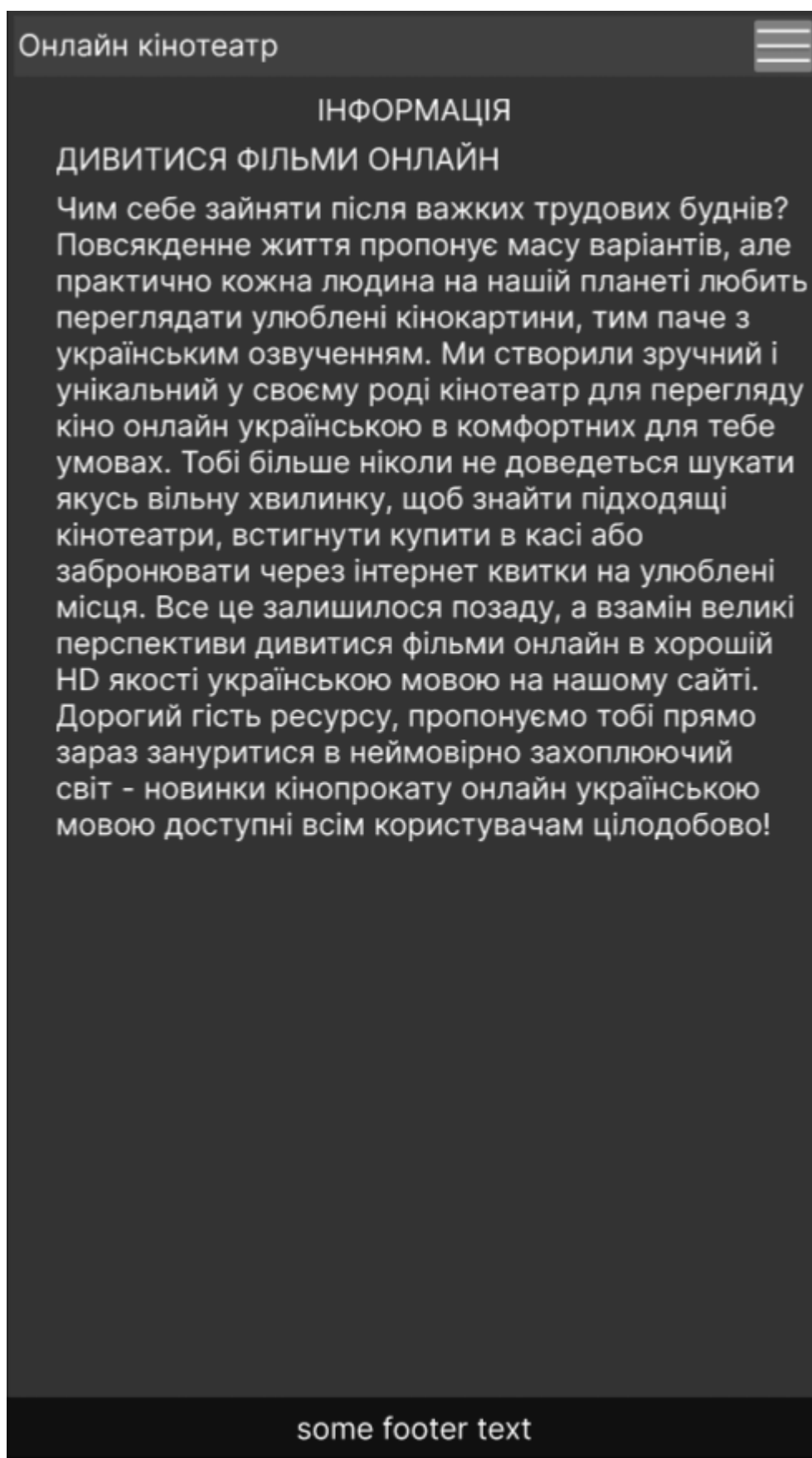


Рисунок 2.39 – Полотно Information, що є сторінкою Інформація з інформацією про сайт

Онлайн кінотеатр

Всі категорії

Оновлення / Видалення категорії

Ім'я :

Опис :

Фото :

Додати

Оновити

Видалити

some footer text

The image shows a mobile application interface for an online cinema. At the top, there is a header bar with the text 'Онлайн кінотеатр' (Online cinema) on the left and a hamburger menu icon on the right. Below the header, the main content area has a title 'Всі категорії' (All categories) and a subtitle 'Оновлення / Видалення категорії' (Update / Delete categories). There are three input fields for 'Ім'я' (Name), 'Опис' (Description), and 'Фото' (Photo). Below these fields are three buttons: 'Додати' (Add), 'Оновити' (Update), and 'Видалити' (Delete). At the bottom of the screen, there is a footer bar with the text 'some footer text'.

Рисунок 2.40 – Полотно Create category, що є сторінкою для додавання, редагування та видалення категорій

Онлайн кінотеатр

Лист продукції

[Оновлення / Видалення продукції](#)

Ім'я :

Опис :

Ціна :

Фото :

Категорія :

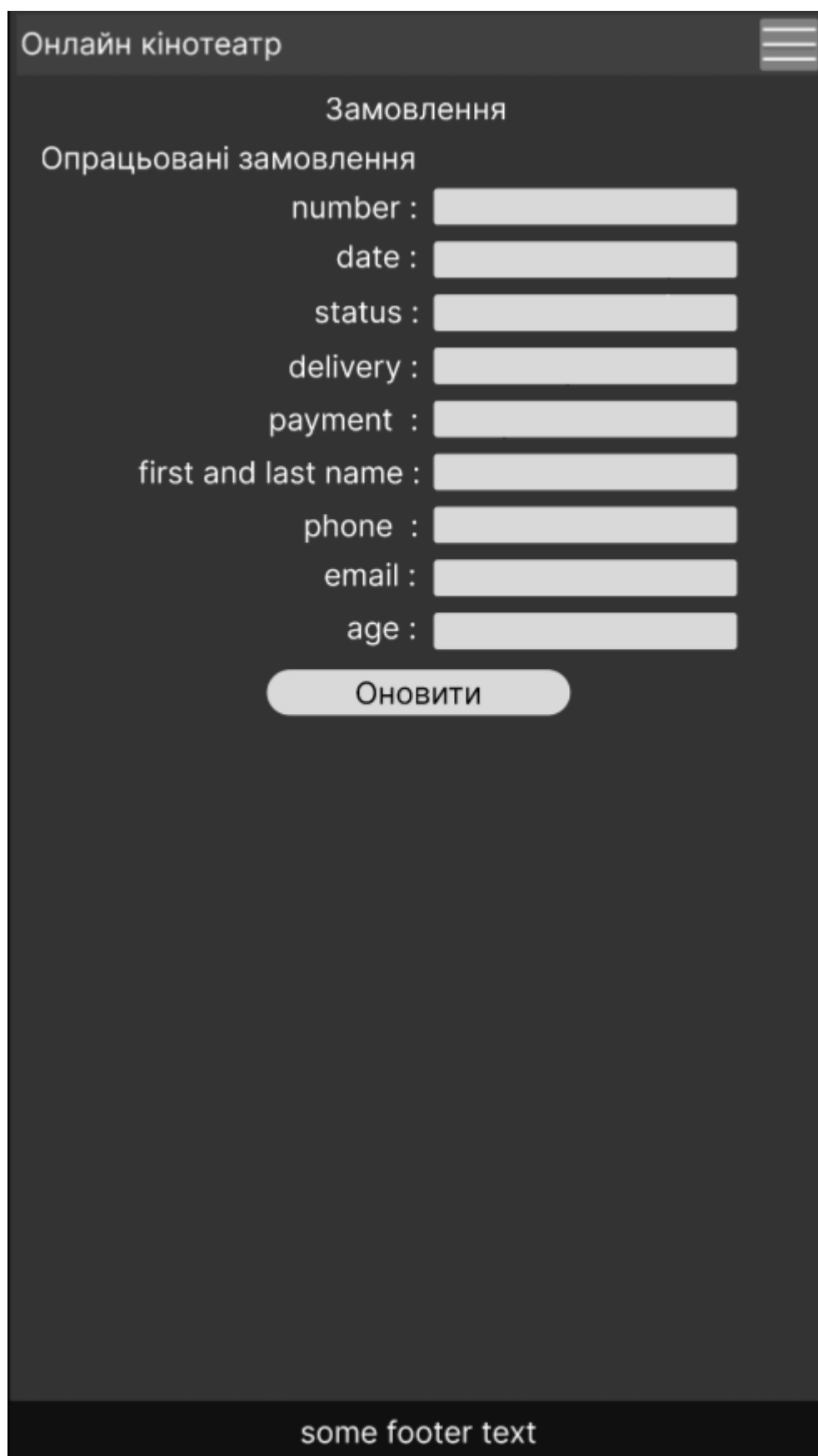
Додати

Оновити

Видалити

some footer text

Рисунок 2.41 – Полотно Create product, що є сторінкою для додавання, редагування та видалення продукції



Онлайн кінотеатр

Замовлення

Опрацьовані замовлення

number :

date :

status :

delivery :

payment :

first and last name :

phone :

email :

age :

Оновити

some footer text

Рисунок 2.42 – Полотно Update order, що є сторінкою для редагування замовлень

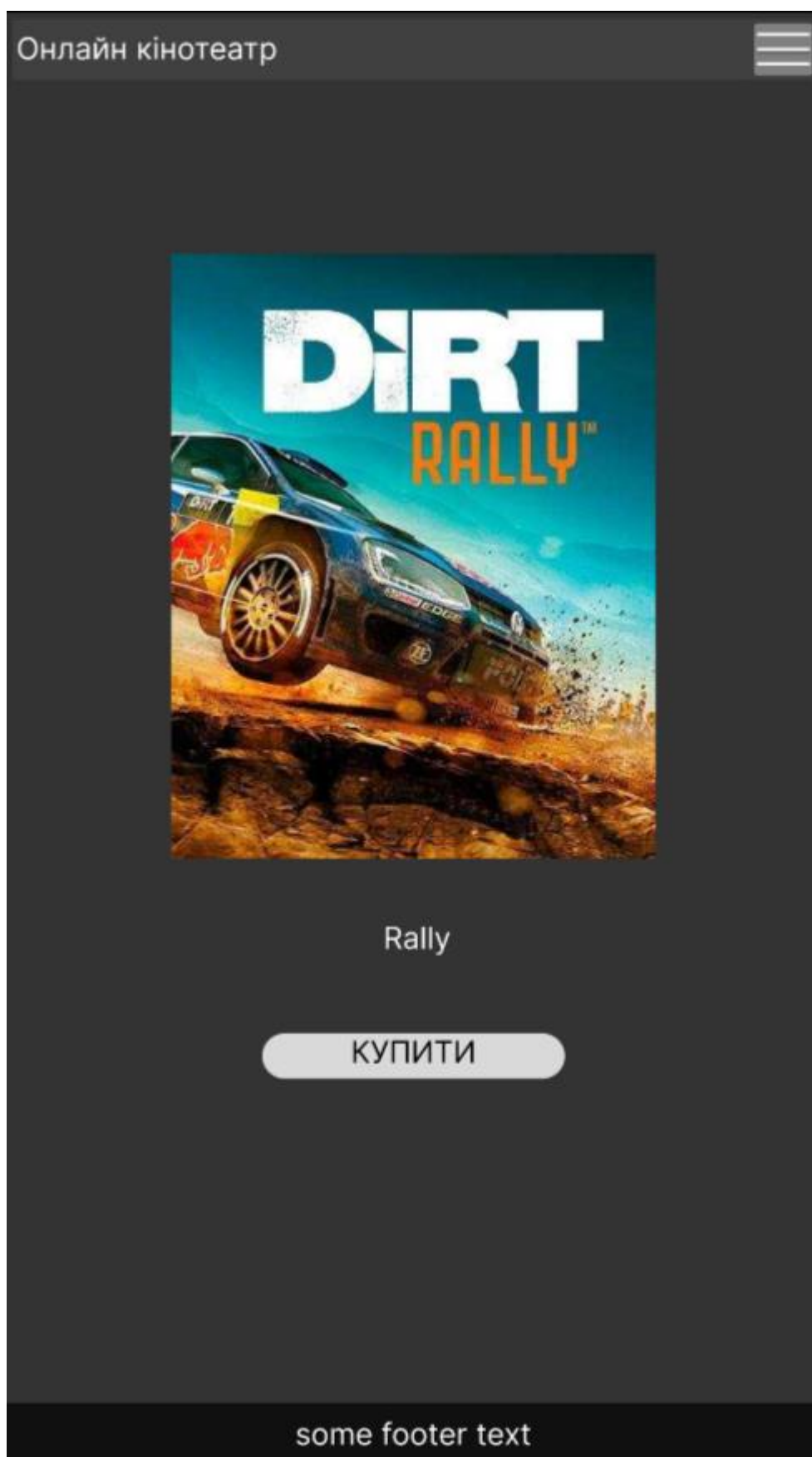


Рисунок 2.43 – Полотно Rally, що є сторінкою для перегляду з якої можна перейти на сторінку для купівлі фільму Rally

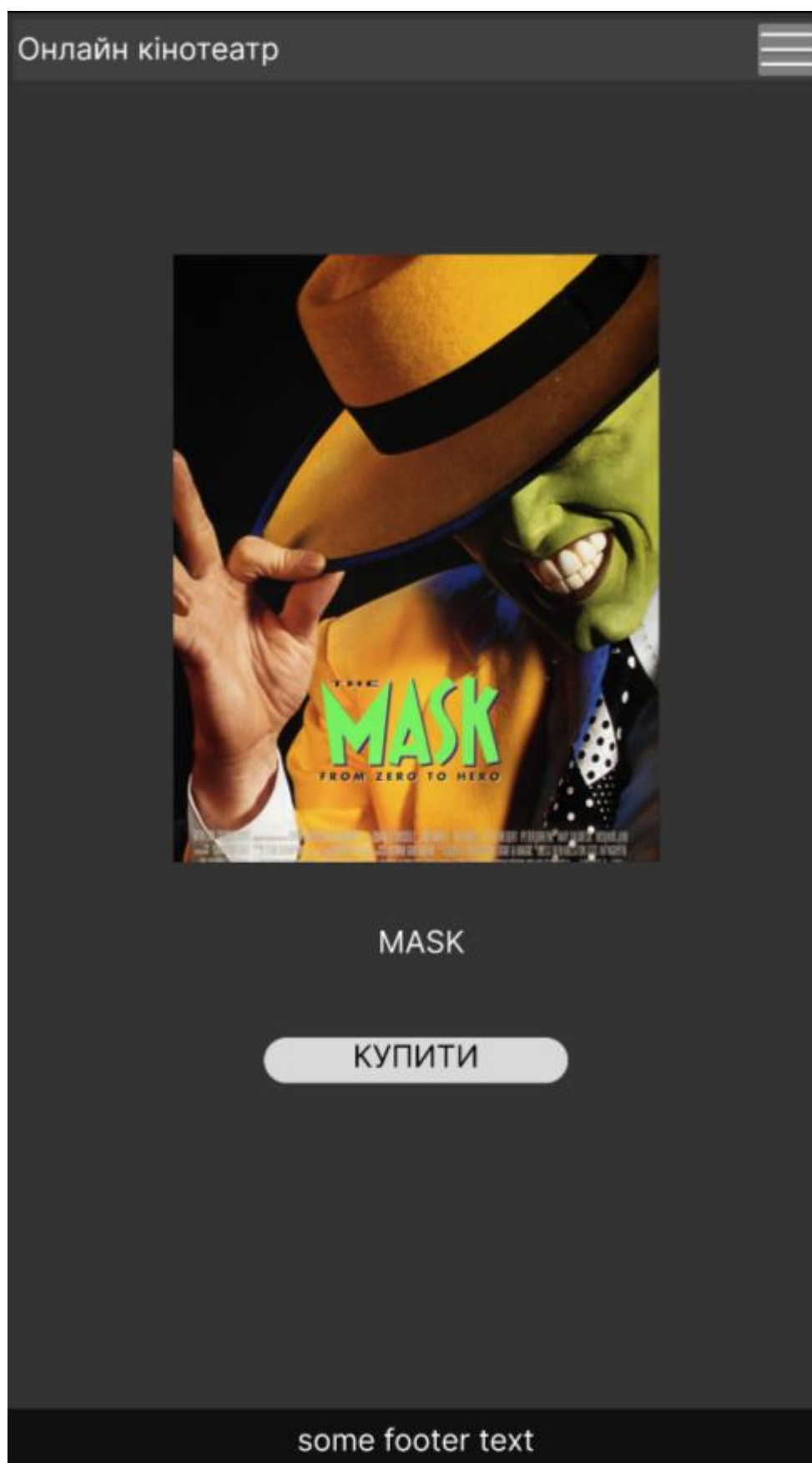


Рисунок 2.44 – Полотно Mask, що є сторінкою для перегляду з якої можна перейти на сторінку для купівлі фільму Mask



Рисунок 2.45 – Полотно Vue, що є сторінкою для купівлі фільму де для цієї операції обирається картка банку

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ПІДСИСТЕМИ

3.1. Інформаційне забезпечення

Для створення бази даних системи було обрано MySQL, вільну систему керування реляційними базами даних. Ця СКБД з відкритим кодом, створена як альтернатива платним аналогам, наразі є однією з найпоширеніших. Її часто використовують для створення динамічних веб-сторінок завдяки чудовій підтримці різних мов програмування. MySQL володіє широким спектром функцій, що забезпечують безпечне середовище для зберігання, обробки та отримання даних[10].

Окрім вищезазначеного, MySQL вирізняється високою швидкістю, стійкістю та простотою використання. Ця СКБД була розроблена для покращення обробки великих обсягів даних. Її серверний код можна компілювати на різних платформах, проте найбільш повноцінно його можливості розкриваються в UNIX-системах, де доступна багатоканальність, що значно підвищує загальну продуктивність. [11].

MySQL сервер володіє численними перевагами, які роблять його популярним вибором для керування базами даних:

- простота у налаштуванні та використанні;
- підтримка необмеженої кількості користувачів;
- масштабованість;
- швидкість виконання команд;
- надійна система безпеки:[12].

26 лютого 2008 року компанія Sun Microsystems придбала MySQL AB за 1 мільярд доларів США. Ця подія стала значною віхою в історії MySQL, адже Sun Microsystems, відомий виробник програмного забезпечення та обладнання, мала значні ресурси для розвитку та просування СКБД.

27 січня 2010 року Oracle Corporation, гігант в індустрії програмного забезпечення, придбала Sun Microsystems. Це призвело до того, що MySQL опинився під контролем Oracle, що володіє власною лінійкою СКБД[13].

Вибір MySQL як СКБД ґрунтується на низці вагомих факторів:

- масштабованість — це означає, що MySQL здатен підтримувати роботу з базами даних значних розмірів. Це підтверджується його використанням у таких компаніях, як Yahoo!, Google, HP та Associated Press. Згідно з документацією MySQL, деякі бази даних, що використовуються компанією MySQL AB (розробником MySQL), містять до 50 мільйонів записів;
- переносність — це означає, що MySQL може працювати на різних платформах, включаючи Unix, Linux, Windows, OS/2, Solaris та Mac OS X. Це робить його універсальним рішенням для різних середовищ;
- зв'язаність — це означає, що MySQL має мережеву структуру, що дозволяє одночасно отримувати доступ до бази даних з будь-якої точки Internet кільком користувачам. Ця функція робить його ідеальним вибором для веб-застосунків та інших багатокористувацьких систем. MySQL також пропонує широкий спектр програмних інтерфейсів додатків (API), які дозволяють встановлювати з'єднання з MySQL з додатків, написаних на таких мовах програмування, як C, C++, Perl, PHP, Java та Python;
- безпека — це означає, що MySQL має систему контролю доступу до даних, яка забезпечує захист даних від несанкціонованого доступу.;
- швидкість функціонування — це означає, що MySQL відомий своєю високою швидкістю виконання команд SQL, що забезпечує швидке та ефективне опрацювання даних.;
- зручність експлуатації — це означає, що MySQL досить легко встановлюється та налаштовується, а також його просто адмініструвати.;

- відкритий код — це означає, що MySQL є програмним забезпеченням з відкритим кодом, що дає користувачам доступ до його вихідного коду та можливість модифікувати його відповідно до своїх потреб. Це робить MySQL гнучким та налаштовуваним рішенням[14].

Після вибору бази даних було розроблено модель, що описує її структуру та взаємозв'язки між елементами. Ця модель представлена на рисунку 3.1.

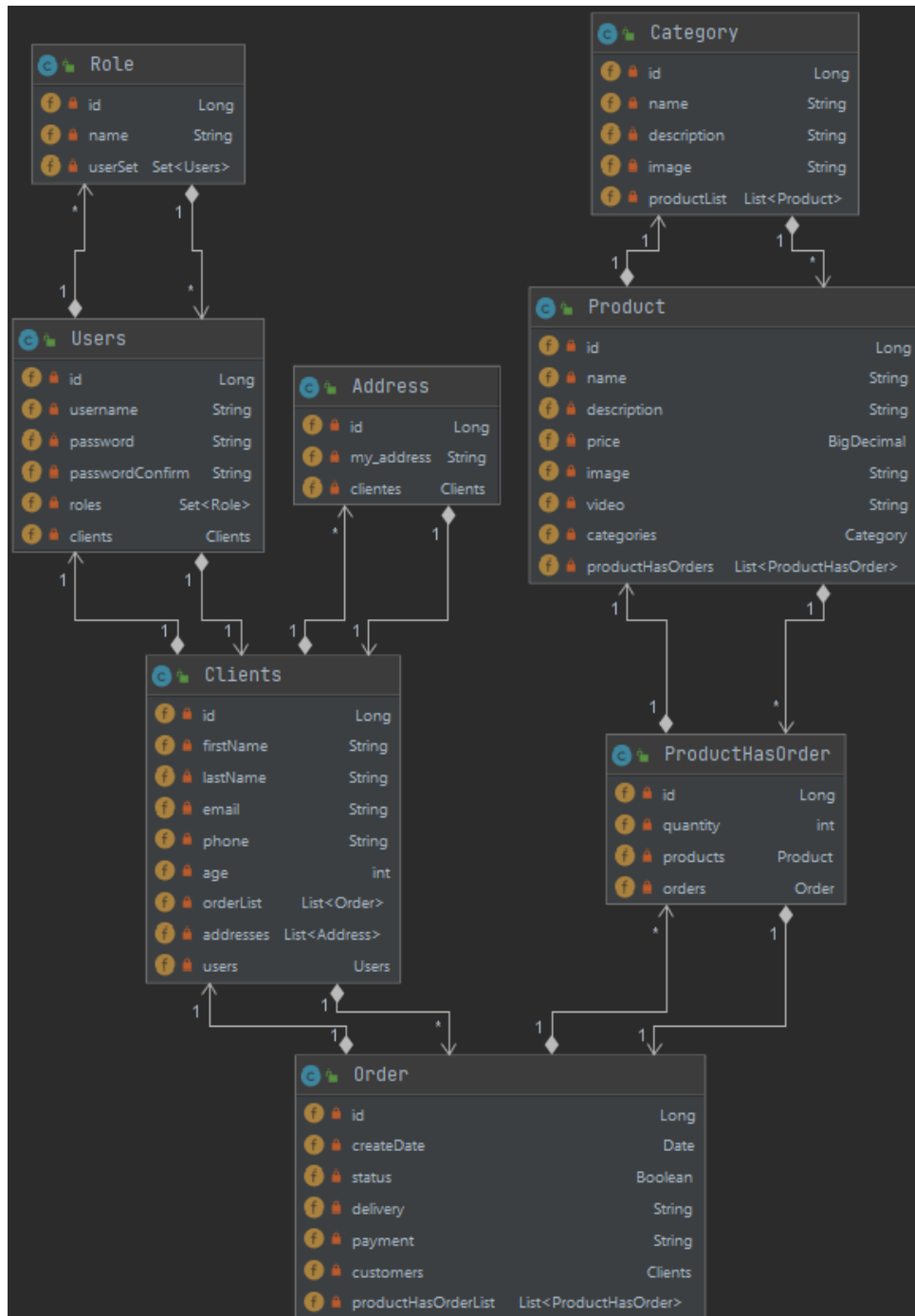


Рисунок 3.1 – Схема бази даних

Використовуючи схему як основу, база даних була створена в MySQL Workbench. Ця база даних містить всі необхідні колекції, деякі з яких є вкладеними, тобто є частиною інших. Структура отриманої бази даних представлена на рисунку 3.2.

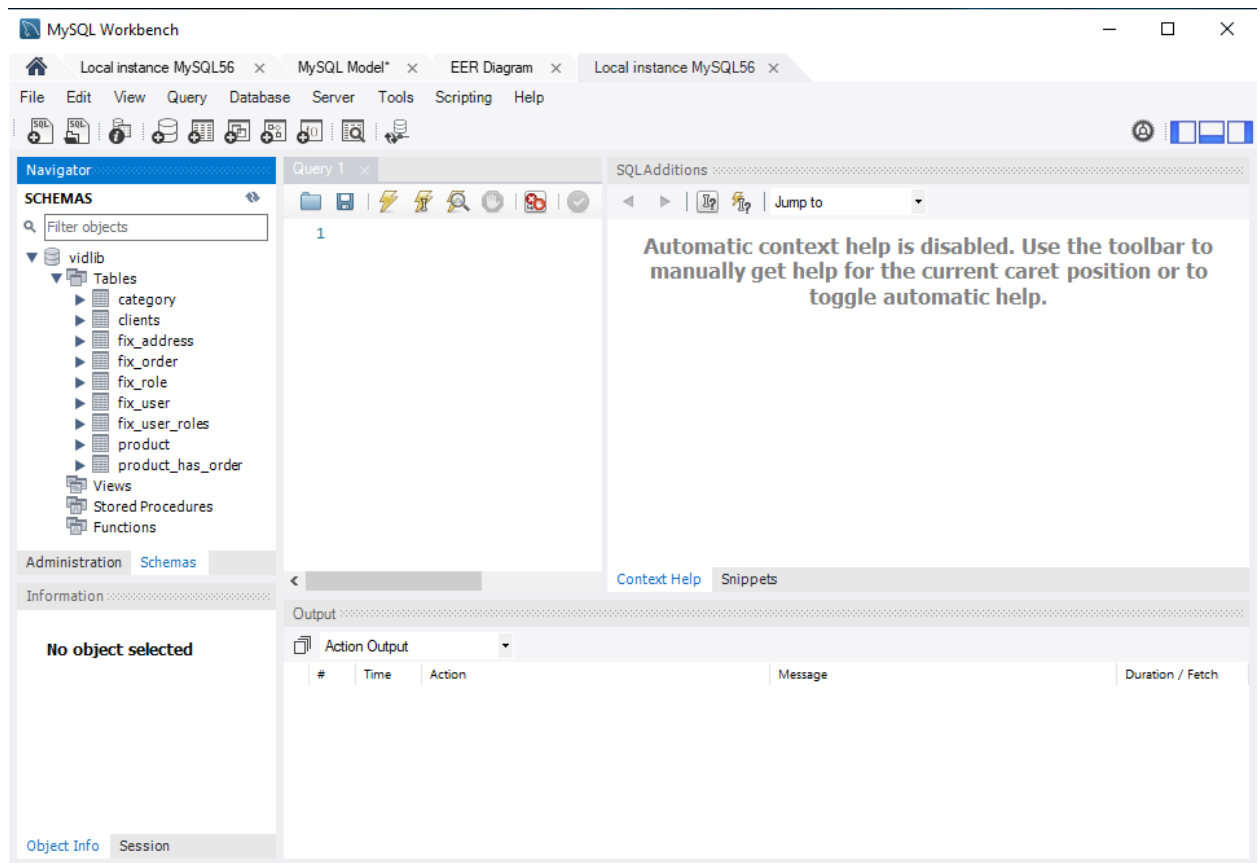


Рисунок 3.2 – База даних побудована в MySQL Workbench

На рисунку 3.3 представлена діаграма класів серверної частини системи, побудована згідно з принципами патерну MVC. Відповідно до MVC, на діаграмі представлені моделі, контролерів та класів сутностей, але на ній відсутні View, переглянути програмний код контролерів можна в «Додатку А» а класів сутностей в «Додатку Б». Класи на діаграмі можна поділити на 3 частини:

1. дані — це зберігання даних;
2. бізнес-логіка — це розрахунки та обробка даних;
3. інтерфейси — це збір та передача інформації користувачу.

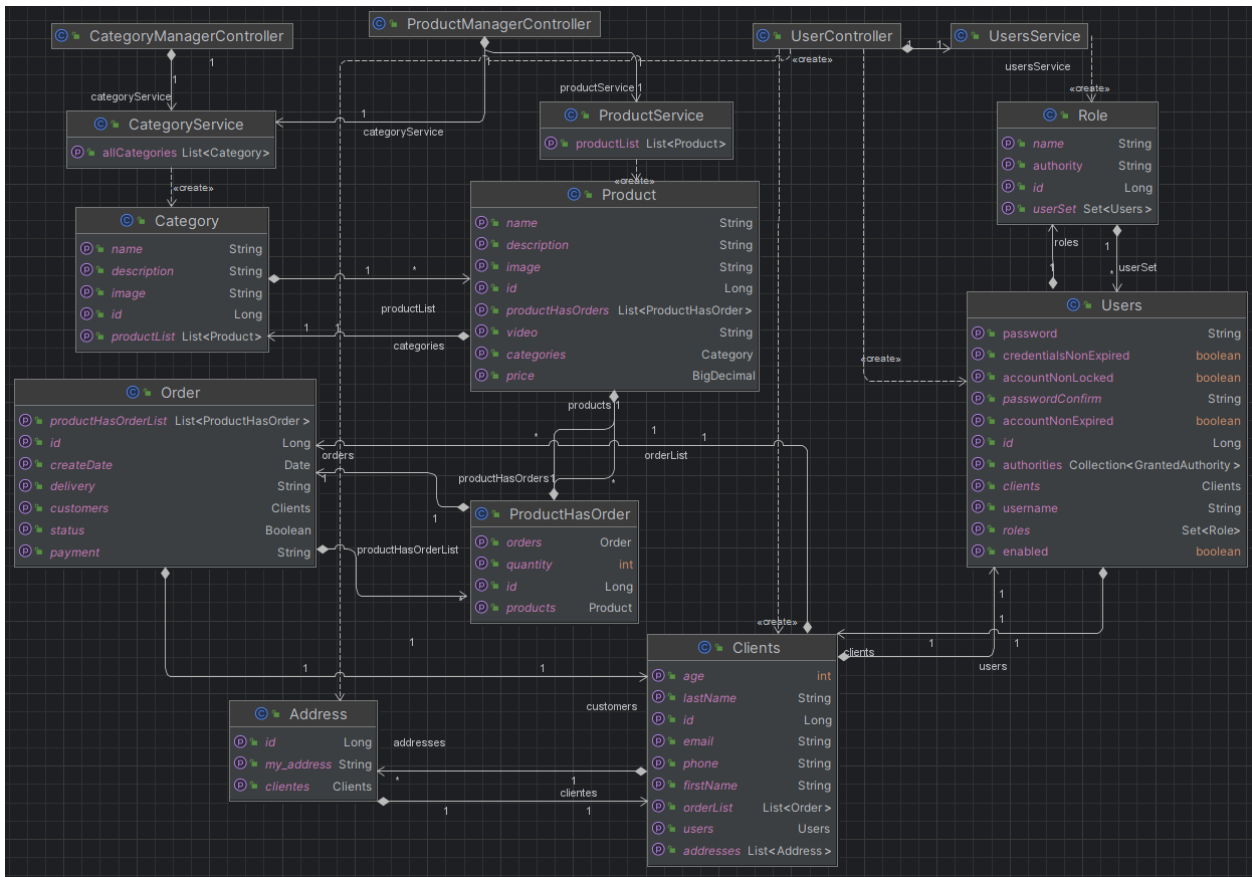


Рисунок 3.3 – Діаграма класів

Для захисту інформації в базі даних застосовуються криптографічні ключі.

У «Додатку В» представлено результат перетворення ER-моделі в конкретну схему бази даних для MySQL, а також вміст файлу згенерованої моделі даних.

Специфікація вхідних та вихідних даних веб-платформи для організації роботи онлайн-кінотеатру представлена на рис. 3.4 – 3.12.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
description	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
image	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 3.4 – Специфікація таблиці «category»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
💎 age	INT	☐	☒	☐	☐	☐	☐	☐	☐	
💎 email	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 first_name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 last_name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 phone	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 users_id	BIGINT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.5 – Специфікація таблиці «clients»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
💎 my_address	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 user_id	BIGINT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.6 – Специфікація таблиці «fix_address»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
💎 create_date	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 delivery	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 payment	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 status	BIT(1)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 client_id	BIGINT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.7 – Специфікація таблиці «fix_order»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
💎 name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.8 – Специфікація таблиці «fix_role»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
💎 password	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
💎 username	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.9 – Специфікація таблиці «fix_user»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 user_set_id	BIGINT	☒	☒	☐	☐	☐	☐	☐	☐	
🔑 roles_id	BIGINT	☒	☒	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.10 – Специфікація таблиці «fix_user_roles»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
🔹 description	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
🔹 image	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
🔹 name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
🔹 price	DECIMAL(19,2)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
🔹 video	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
🔹 category_id	BIGINT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.11 – Специфікація таблиці «product»

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 id	BIGINT	☒	☒	☐	☐	☐	☐	☒	☐	
🔹 quantity	INT	☐	☒	☐	☐	☐	☐	☐	☐	
🔹 order_id	BIGINT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
🔹 product_id	BIGINT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.12 – Специфікація таблиці «product_has_order»

3.2. Технічне забезпечення

Архітектура будувалась таким чином щоб якнайкраще відповідати вимогам до системи яка створюється відповідно до принципу "форма відповідає функції".

Вона ґрунтується на принципі розділення функціональності між двома типами компонентів:

- сервери надають ресурси та послуги клієнтам;
- клієнти ініціюють запити до серверів, щоб отримати доступ до ресурсів або послуг.

Основні характеристики архітектури клієнт-сервер:

- незалежність — це означає, що сервери та клієнти функціонують незалежно один від одного. Це робить систему гнучкою та масштабованою.

- розподіленість — це означає, що сервери та клієнти можуть бути фізично розташовані в різних місцях, що робить систему доступною з будь-якої точки мережі.
- балансування навантаження — це означає, що один сервер може обробляти запити від декількох клієнтів, що покращує продуктивність системи.
- відсутність взаємодії клієнтів — це означає, що клієнти не мають прямого зв'язку один з одним, а взаємодіють лише з серверами.

На рис. 3.13 представлена архітектура Клієнт-серверу.

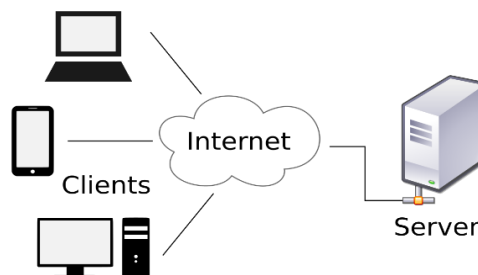


Рисунок 3.13 – Архітектура Клієнт-серверу

На представленій діаграмі розгортання, відображеній на рисунку 3.14, видно, що ІС складається з трьох взаємодіючих блоків. Блок "Клієнт" зображено як програмне забезпечення, яке встановлюється на комп'ютер користувача, і складається з двох компонентів: програмної логіки і користувацького інтерфейсу.

Блок «Сервер» включає в себе кілька компонентів, включаючи "Базу даних", де зберігаються дані системи. Також в цьому блоку присутній API, призначений для взаємодії з клієнтами, які звертаються до сервера для обміну інформацією. Додатково, у цьому блоку передбачено API для проведення оплати, і саме через цей компонент клієнти взаємодіють із сервером для проведення фінансових транзакцій.

Останній блок, «Сервіс оплати», представляє собою зовнішню систему, що відповідає за обробку запитів на оплату. Важливо відзначити, що на діаграмі відображено лише взаємодію сервера з компонентом "API для проведення оплати" у межах цього блоку, хоча, очевидно, в сервісі оплати може бути більше компонент, ніж зображено на діаграмі.

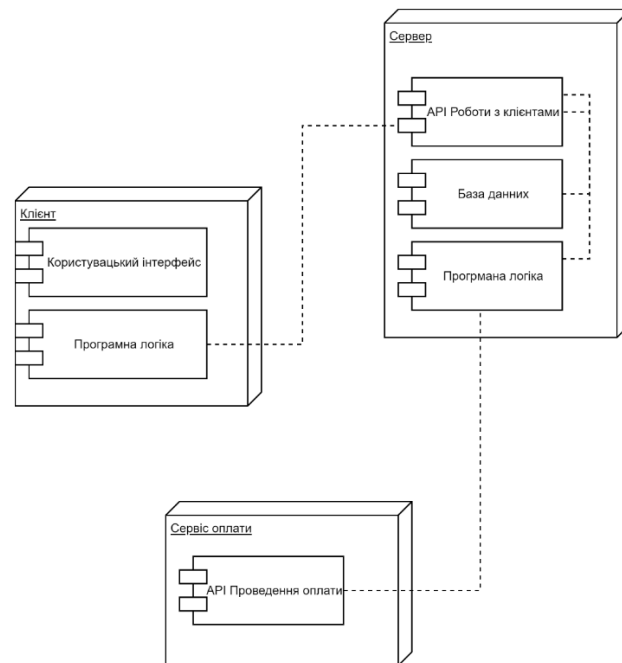


Рисунок 3.14 – Діаграма розгортання

3.3. Програмне забезпечення

Рисунок 3.15 ілюструє діаграму компонентів системи для процесу входу в акаунт. Ця діаграма візуалізує взаємодію між компонентами системи та їх розміщення в контексті Spring Boot, що може бути описане наступним чином:

1. login.ftl: Графічний інтерфейс, який отримує введені користувачем дані (ім'я користувача та пароль) та взаємодіє з логікою для обробки введених даних.
2. Entry (Компонент входу): Компонент, який отримує дані від графічного інтерфейсу (login.ftl) та починає обробку введеної інформації.
3. UserControll: Логічний компонент, який управляє операціями, пов'язаними із користувачем. Отримує дані від Entry та може ініціювати взаємодію з іншими компонентами.
4. ClientRepository: Компонент, відповідальний за взаємодію із базою даних (DataBase). UserControll може звертатися до ClientRepository для отримання чи збереження інформації про клієнтів.

5. DataBase (База даних): Система для зберігання та управління даними. ClientRepository взаємодіє з базою даних для виконання операцій збереження та витягування даних.

Також слід зауважити, що Spring Boot, як фреймворк, надає можливості для розробки веб-додатків та підтримує інтеграцію з різними шарами додатка, включаючи веб-рівень (контролери), службовий рівень (сервіси), та рівень доступу до даних (репозиторії). Таким чином, вищезгадані компоненти можуть відповідати конкретним шарам або класам додатка в рамках архітектури Spring Boot[15].

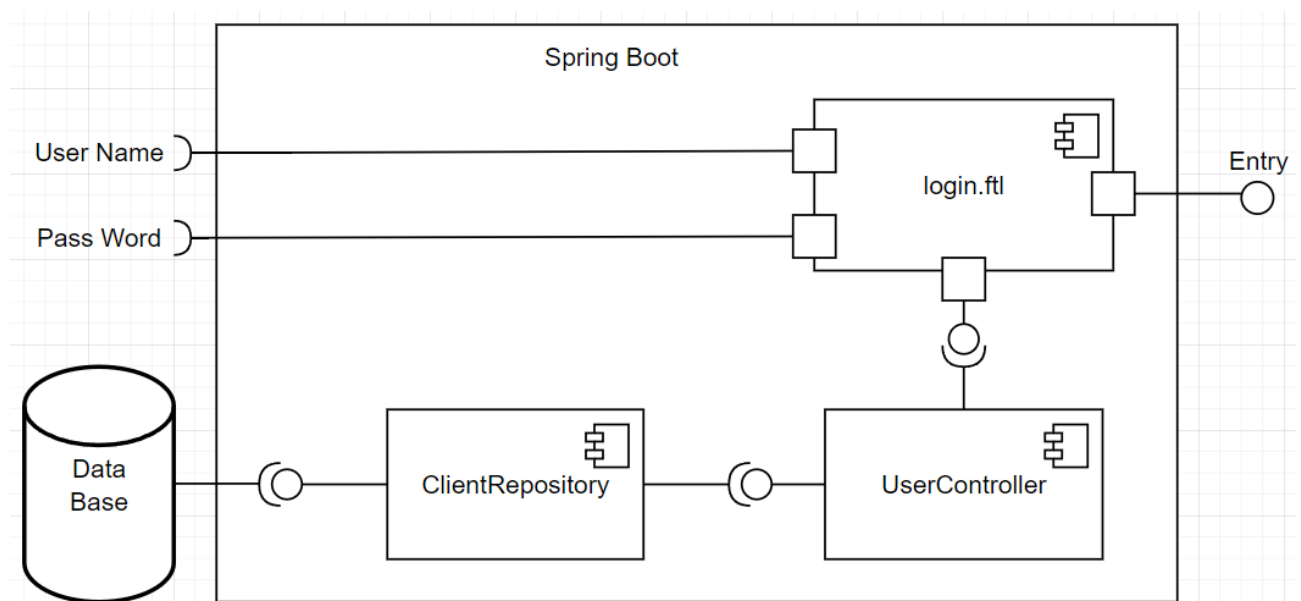


Рисунок 3.15 – Діаграма компонентів, що використовуються при вході в акаунт

Архітектура веб-платформи для організації роботи онлайн-кінотеатру включає до себе наступні компоненти:

- інтерфейси для забезпечення зручної взаємодії користувачів з її функціоналом у вигляді сторінок сайту: головна, авторизація, реєстрація, сторінки контенту, особистий кабінет, сторінка покупки, а також в подальшому їх можна буде збільшувати і покращувати;
- база даних, в якій зберігається інформація про користувачів, контент, жанри контенту, відгуки клієнтів, списки бажаних товарів, списки

улюблених товарів, сплачений контент, платіжні картки, відгуки та оцінки;

- серверна частина, яка включає велику кількість компонентів, основним з яких є файл з назвою `romp.xml`, який зберігає в собі всі базові і додані бібліотеки які в подальшому збільшують функціональність нашої системи.

Архітектурою веб-платформи для організації роботи онлайн-кінотеатру є Model View Controller що представлена на рис. 3.16.

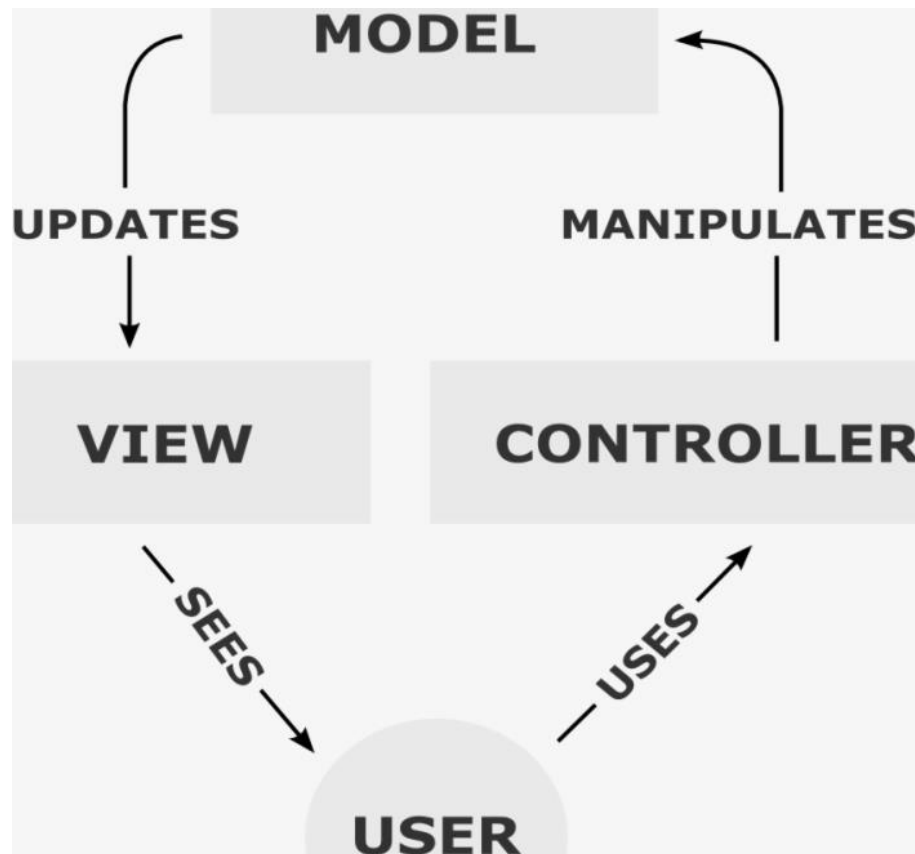


Рисунок 3.16 – Принцип роботи MVC

Для розробки веб-платформи для організації роботи онлайн-кінотеатру було обрано RAD.

RAD вирізняється трьома ключовими перевагами[16]:

1. Швидкість розробки: Завдяки інкрементному підходу та використанню прототипів, RAD дозволяє створювати програмні продукти значно

швидше, ніж традиційні методи розробки. Це робить його ідеальним вибором для проектів з обмеженими термінами.

2. Низька вартість: RAD зазвичай потребує менших команд розробників та менш тривалого періоду розробки, що веде до значного зниження витрат на розробку програмного забезпечення.
3. Висока якість: RAD робить акцент на постійному тестуванні та вдосконаленні продукту протягом усього циклу розробки. Це гарантує, що кінцевий продукт відповідатиме всім функціональним та нефункціональним вимогам замовника, а також буде мати високу якість коду та документації.

Завдяки чіткій документації та зручному дизайну системи, її експлуатація та модифікації в майбутньому стають значно простішими та дешевшими[17]. Поетапна розробка по методу RAD зображена на рис. 3.17.

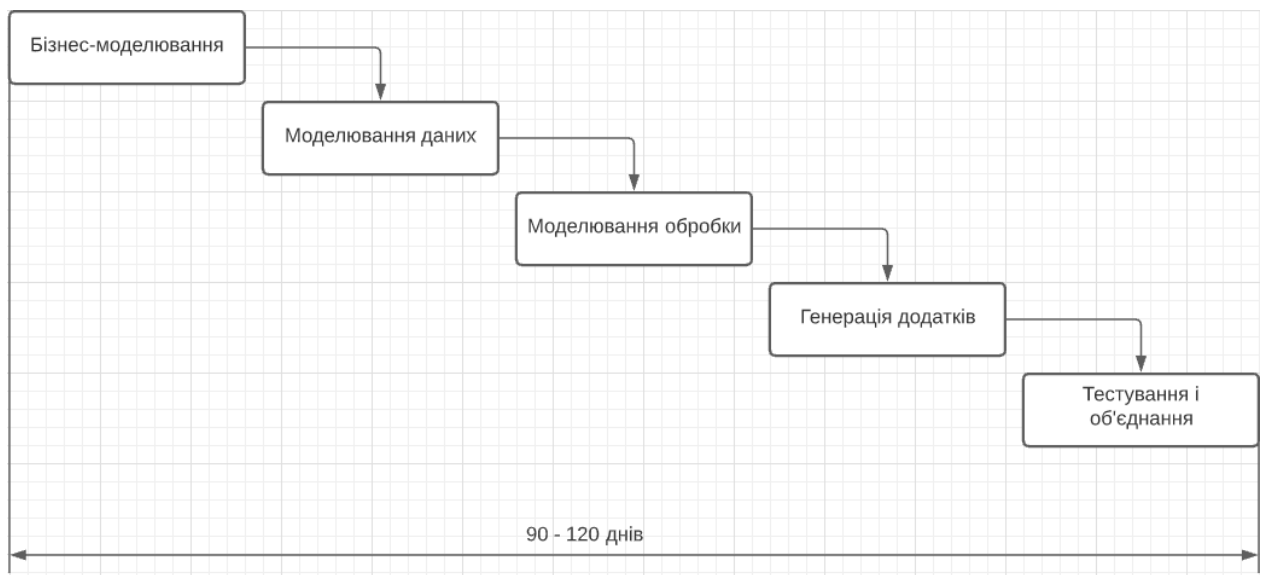


Рисунок 3.17 – Розробка програми за методом RAD

Для розробки веб-платформи для організації роботи онлайн-кінотеатру було обрано мову програмування JAVA.

Java - це потужна мова програмування, розроблена компанією Sun Microsystems у 1995 році. З 2009 року її підтримує Oracle, яка придбала Sun Microsystems[18].

"Oracle" надає компілятор Java та віртуальну машину Java, які відповідають вимогам Java Community Process і працюють під ліцензією GNU General Public License[19]. Мова суттєво використовує синтаксис з C і C++, зокрема, вона базується на об'єктній моделі C++, але з певними модифікаціями. Ці зміни спрямовані на усунення потенційних конфліктів, що можуть виникати через помилки програмістів, та полегшення процесу розробки об'єктно-орієнтованих програм. Деякі завдання, які зазвичай виконує програміст в C/C++, перепадають на віртуальну машину[20]. Основна мета розробки Java - створення платформи-незалежної мови програмування, що призвело до обмеження її можливостей роботи з апаратним забезпеченням порівняно з C++. Тим не менш, Java дозволяє викликати підпрограми, написані на інших мовах програмування, за потреби. Для реалізації Frontend були використанні мови програмування CSS та JSP, також була використовувалась розмітка FTL[21].

CSS – це мова стилю веб-сторінок, що застосовується при описі їх вигляду. Самі сторінки написані мовами розмітки даних[22].

Чому саме CSS:

- усі стильові дані для веб-сайту або його елементів можна зберігати в одному файлі CSS, що спрощує зміни в дизайні та презентації сторінок;
- різні стилі можуть бути застосовані для різних типів користувачів, таких як великий розмір шрифту для тих, хто має проблеми зі зором, стилі для друку або мобільних пристроїв;
- відокремлення стилів від вмісту дозволяє сторінці стати більш структурованою, оскільки правила застосування стилів відокремлені від самого тексту;
- завантаження сторінок може бути прискорено, а обсяг передаваної інформації зменшений завдяки можливості кешування стилів в браузері, що дозволяє використовувати їх для всіх сторінок без повторного завантаження.

JSP – це технологія, яка дозволяє створювати динамічні веб-сторінки, такі як HTML, XML та інші. Розробка JSP почалась у 1997 році, а згодом цю

технологію включили до складу Java EE – платформи для розробки веб-додатків. JSP дозволяє вставляти Java-код у статичний контент сторінки, а також використовувати бібліотеки JSP тегів для їх включення у JSP-сторінки[23]. Сторінки JSP компілюються в сервлети за допомогою JSP-компілятора, що перетворює їх у Java-класи, які виконуються на сервері. Ці технології доповнюють одна одну, і JSP є важливою складовою платформи Java EE[24].

Ось причини використання JSP:

- швидкість, JSP-сторінки виконуються швидше, ніж інші динамічні сторінки;
- простота використання, JSP ґрунтується на Java, тому розробникам, які знайомі з Java, буде легко її використовувати;
- гнучкість, JSP можна використовувати для створення різноманітних веб-додатків;
- масштабованість, JSP можна використовувати для створення масштабованих веб-додатків, які можуть обробляти велику кількість трафіку;
- підтримка, JSP має велику спільноту розробників та багато ресурсів для навчання[25].

FreeMarker – це безкоштовний механізм шаблонів на основі Java , який спочатку фокусувався на динамічній генерації веб-сторінок з архітектурою програмного забезпечення MVC. Тим не менше, це універсальний механізм шаблонів, без залежності від сервілетів або HTTP або HTML і таким чином , часто використовується для джерела генерування коду, файлу конфігурації, або по електронній пошті[26].

FreeMarker простий у використанні, працює з будь-якими текстовими файлами і до того ж, цей потужний інструмент, безкоштовний[27].

Для розробки веб-платформи для організації роботи онлайн-кінотеатру було обрано середовище розробки IntelliJ IDEA Ultimate.

IntelliJ IDEA спочатку була розроблена для мови програмування Java, але згодом отримала підтримку таких мов, як JavaScript, TypeScript та інші[28]. Вона

надає всі необхідні інструменти для розробки як клієнтських, так і серверних додатків[29]. Також у неї є засоби для перегляду більшості відомих СКБД і інтеграція з системами контролю версій, такими як Git, а також для зручного використання інструментів, наприклад, Maven та npm[30].

IntelliJ IDEA - це не просто IDE для Java, а й багатофункціональний інструмент для розробників, який підтримує широкий спектр мов програмування. Окрім Java, IntelliJ IDEA чудово підходить для розробки на таких мовах, як JavaScript, TypeScript, Python, C++, Go, Ruby, Kotlin, Scala та багато інших[31].

IntelliJ IDEA пропонує все необхідне для розробки як веб-додатків, так і back-end систем. Середовище має вбудовані інструменти для роботи з більшістю популярних СКБД, таких як MySQL, PostgreSQL та Oracle Database. IntelliJ IDEA також інтегрується з системами контролю версій, такими як Git та SVN, та пропонує зручне використання інструментів автоматизації, таких як Maven, Gradle та npm. [32].

Для розробки веб-платформи для організації роботи онлайн-кінотеатру було обрано систему проектування та розробки баз даних MySQL.

3.4. Результати реалізації інформаційної підсистеми

Тепер розглянемо результати програмної реалізації веб-платформи для організації роботи онлайн-кінотеатру, що представлені на рис. 3.18 – 3.27. Програмний код цих веб-сторінок представлено у «Додатку Г».

На рисунку 3.18 представлено головну сторінку для незареєстрованого користувача на якій є можливість перейти до інших сторінок або безпосередньо до перегляду самого відео контенту але при спробі це зробити користувача перекине на сторінку входу.

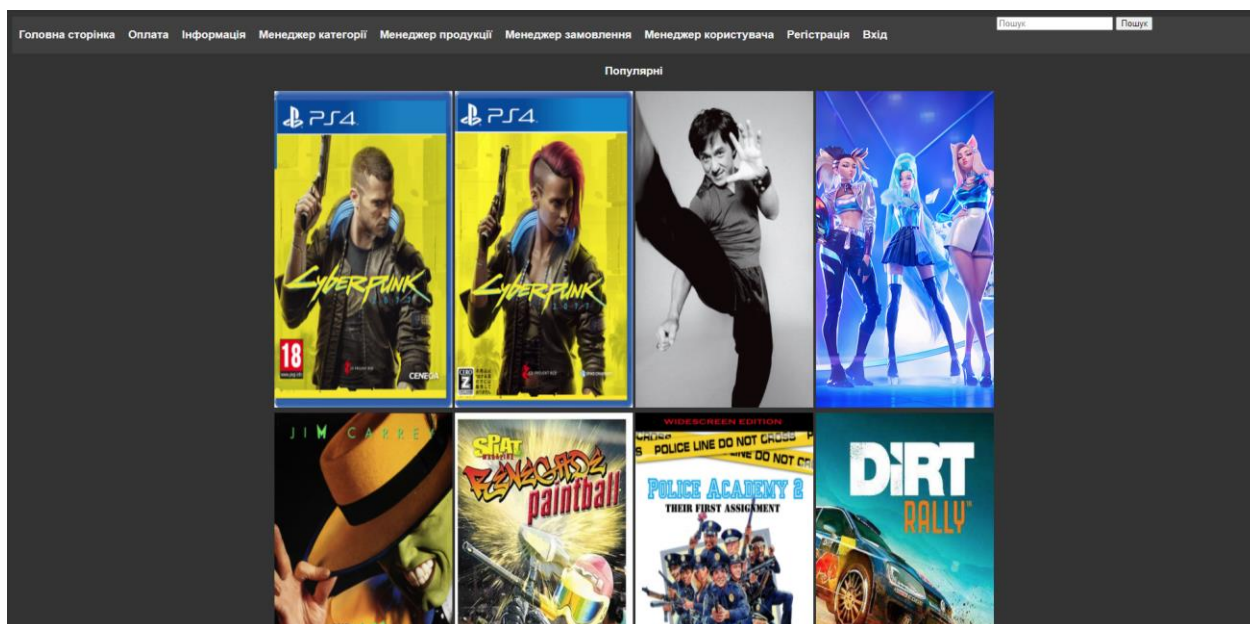


Рисунок 3.18 – Зображена головна сторінка для не зареєстрованого користувача

На рисунку 3.19 продемонстрована сторінка для входу користувачу на якій доступно два варіанти: увійти на сайт або перейти на сторінку реєстрації.

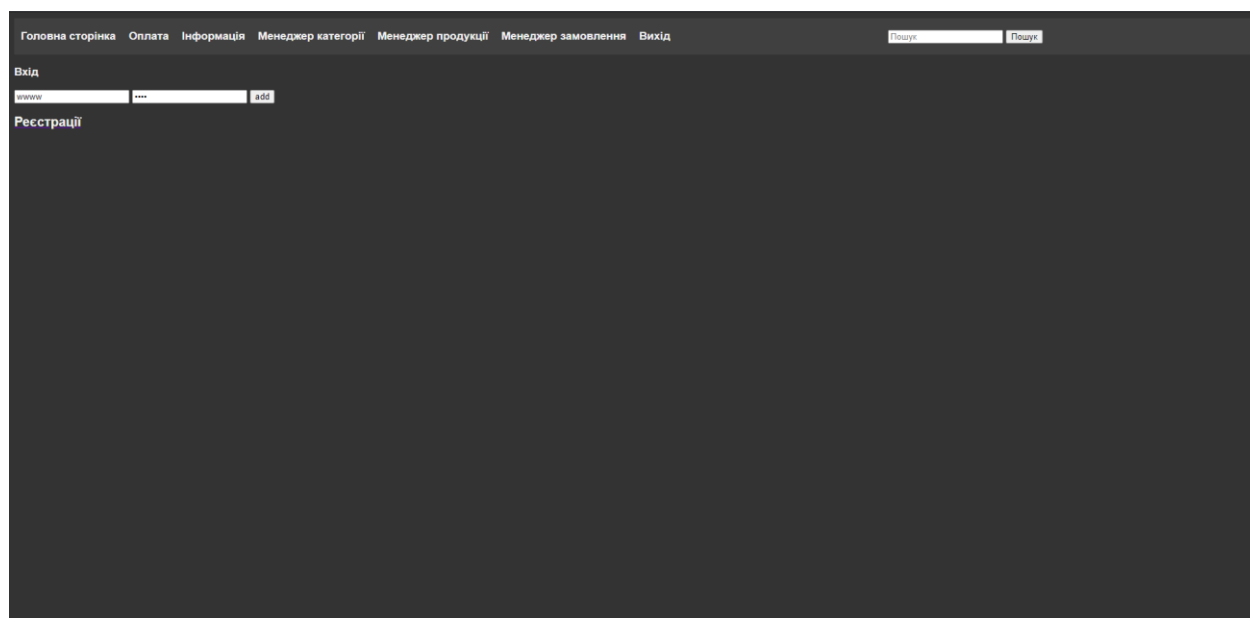


Рисунок 3.19 – Сторінка входу

На рисунку 3.20 та 3.21 продемонстрована сторінка реєстрації користувача на якій потрібно ввести для реєстрації на сайті: логін, пароль, ім'я, фамілію, електронну адресу, номер телефону, вік та адресу проживання.

Головна сторінка Оплата Інформація Менеджер категорій Менеджер продукції Менеджер замовлення Вихід

Пошук Пошук

Регістрація

Логін

Пароль

Повторіть пароль

Ім'я

Фамілія

Електронна пошта

Телефон

Вік

Рисунок 3.20 – Сторінка реєстрації

Пароль

Повторіть пароль

Ім'я

Фамілія

Електронна пошта

Телефон

Вік

Адреса

add

Рисунок 3.21 – Сторінка реєстрації

На рисунку 3.22 продемонстрована головна сторінка для зареєстрованого користувача з якої можна перейти на такі сторінки ,а саме, перегляд контенту, оплата, інформація, менеджер категорій, менеджер продукції та менеджер замовлення також на сторінці є кнопка для виходу з акаунту та поле для пошуку відео контенту на сайті.

На рисунку 3.23 продемонстрована сторінка для перегляду контенту на якій можна перегляду придбаний відео контент на сайті.

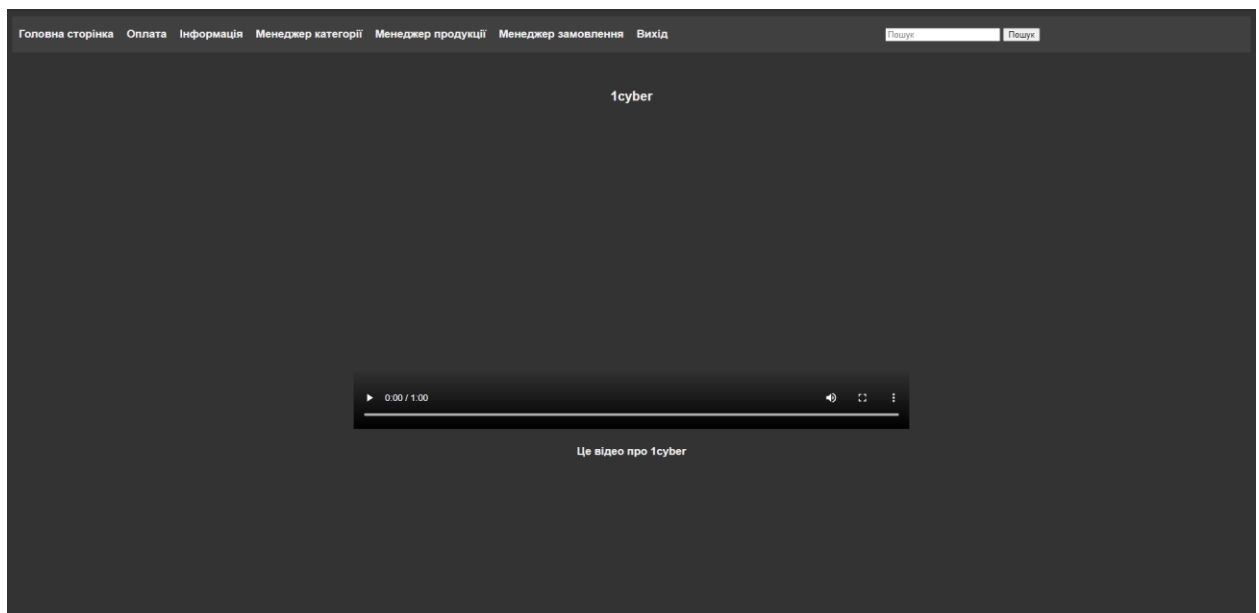


Рисунок 3.23 – Перегляд контенту «cyber1»

На рисунку 3.24 та 3.25 продемонстрована сторінка менеджера категорій на якій можна створювати, редагувати та видаляти категорії для відео контенту.

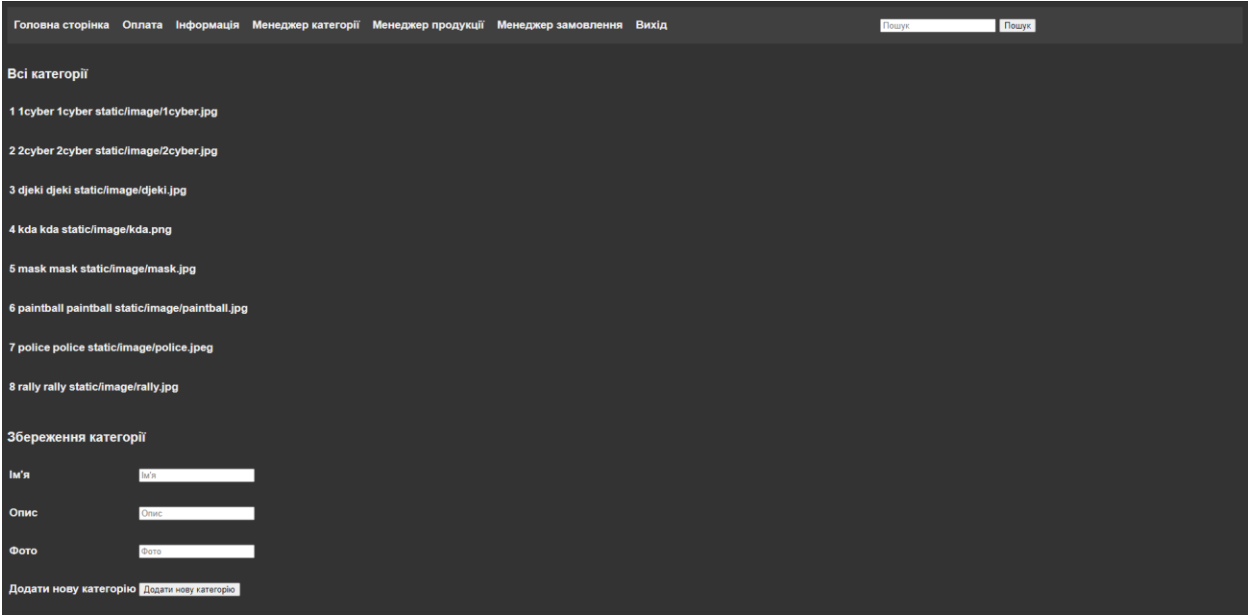


Рисунок 3.24 – Сторінка менеджера категорій

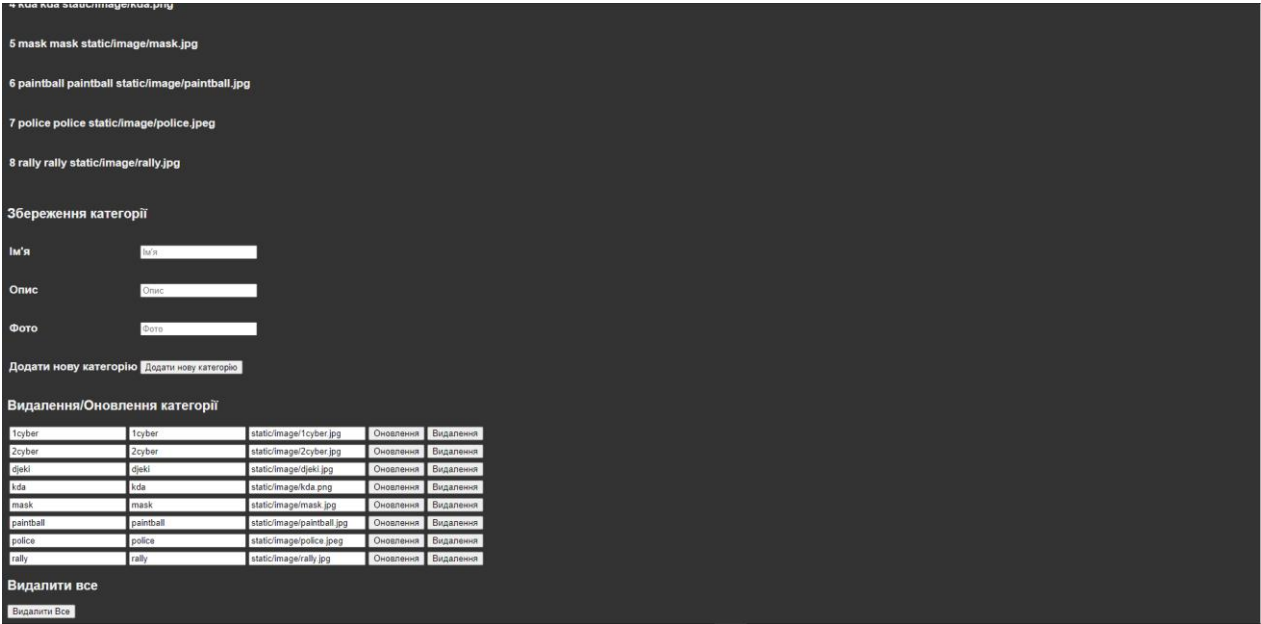


Рисунок 3.25 – Сторінка менеджера категорій

На рисунку 3.26 та 3.27 продемонстрована сторінка менеджера продукції на якій можна створювати, редагувати та видаляти відео контент.

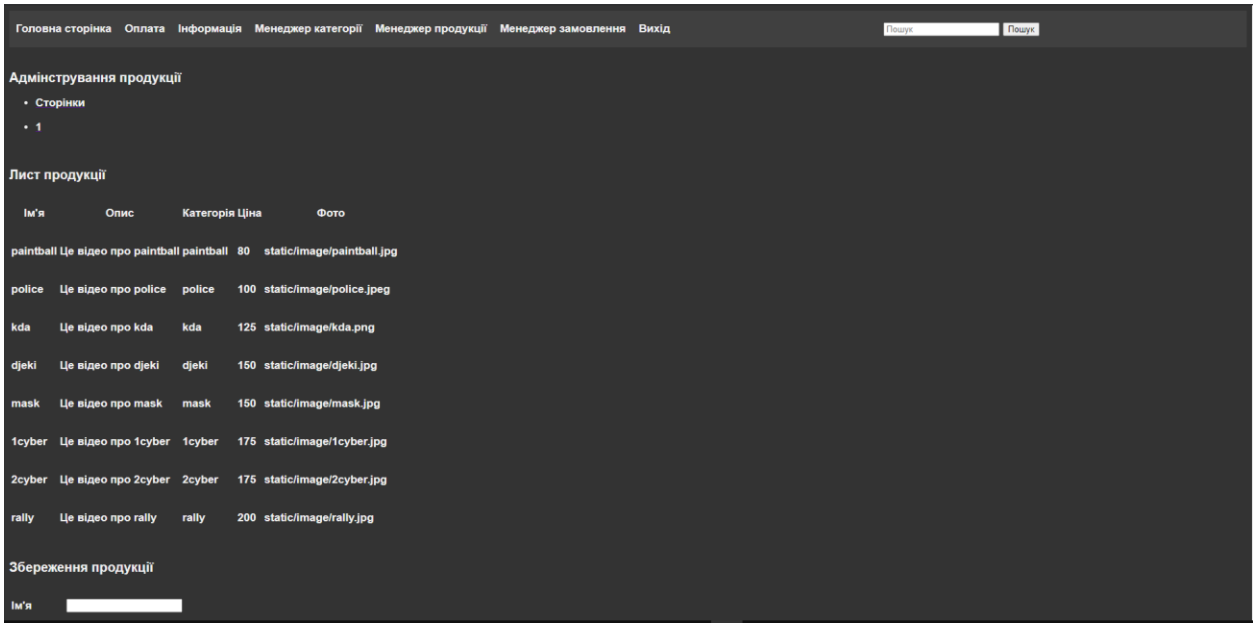


Рисунок 3.26 – Сторінка менеджера продукції

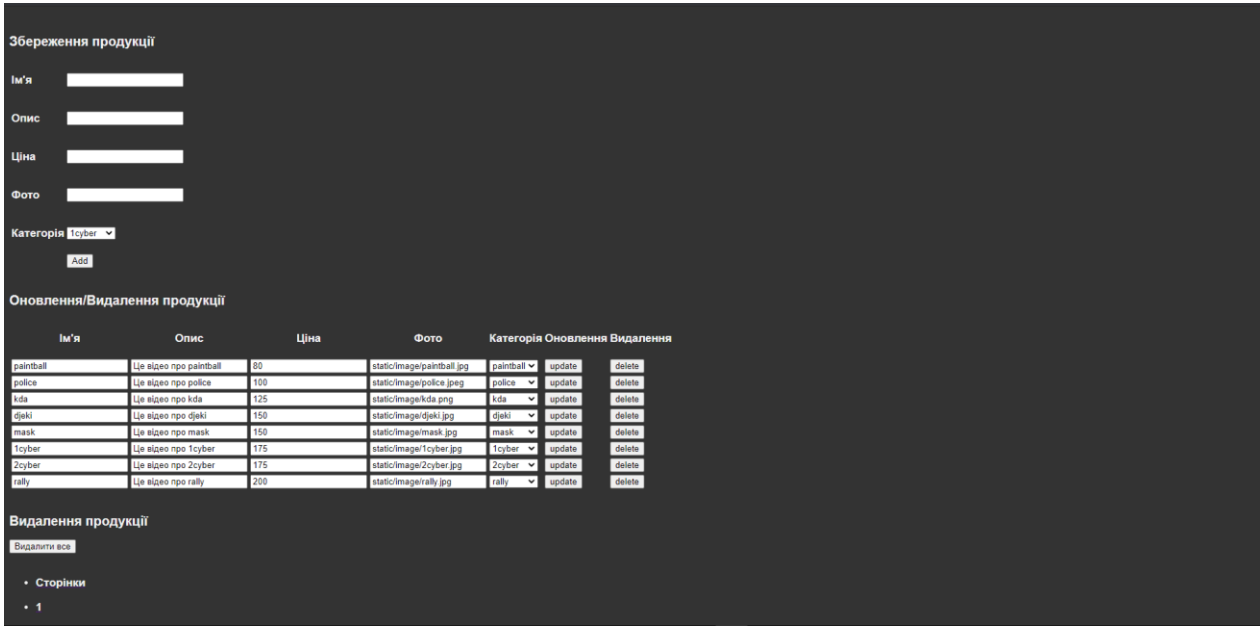


Рисунок 3.27 – Сторінка менеджера продукції

Отже, розроблена веб-платформи для організації роботи онлайн-кінотеатру працездатна і відповідає встановленим до неї вимогам.

ВИСНОВКИ

У першому розділі кваліфікаційного бакалаврського проекту проаналізовано розвиток ринку інформаційних технологій в Україні та діяльність онлайн кінотеатрів. Онлайн кінотеатри, як веб-сайти, надають послуги реклами, продажу, маркетингу та посередництва у сфері кіноіндустрії. Основними клієнтами яких є користувачі, що бажають переглядати контент онлайн. Порівняння платформ NETFLIX, MEGOGO, Apple TV+ та SWEET.TV показав, що кожна має унікальні характеристики. NETFLIX пропонує великий обсяг оригінального контенту, але в Україні може бракувати дубляжу українською. MEGOGO має велику бібліотеку і власні телеканали, але є скарги на якість контенту. Apple TV+ відомий якісним контентом і безкоштовним пробним періодом, але обсяг контенту менший. SWEET.TV – інноваційна платформа з доступом до HD-телебачення та українських фільмів.

У другому розділі обґрунтовано вимоги та моделювання інформаційної підсистеми для веб-платформи онлайн-кінотеатру. Бізнес-вимоги охоплюють різні категорії стейкхолдерів, такі як юридичний відділ, клієнти, власники кінотеатру, розробники, партнери, маркетинговий відділ і технічна підтримка. Функціональні вимоги включають у себе аутентифікацію, пошук відеоматеріалів, оплату та перегляд. Нефункціональні вимоги стосуються надійності, безпеки та продуктивності системи. Описано також і вхідну та вихідну інформацію, такі як реєстрація користувачів, заповнення інформації, додавання категорій та продукції, придбання продукції та управління ролями користувачів. Математичне забезпечення та алгоритм функціонування системи демонструють взаємодію користувачів з БД. Моделювання поведінки та структури системи продемонстровані на UML-діаграмах прецедентів, послідовності, внутрішніх блоків і класів. Розроблені тест-кейси перевіряють основні функції веб-платформи. Моделювання користувацького інтерфейсу показує зв'язки між сторінками та переходи між ними, демонструючи зручність навігації.

У третьому розділі розглянуто проектування та реалізацію компонентів інформаційної підсистеми веб-платформи для онлайн-кінотеатру. Для керування БД обрано MySQL, на основі якого створено даталогічну модель у програмі MySQL та Workbench. Архітектура включає клієнт-серверну модель та модель

Model View Controller (MVC). Створено діаграми класів, компонентів, розгортання та архітектури системи. Програмне забезпечення розроблялося за допомогою Rapid Application Development (RAD) з використанням мови програмування JAVA, а для фронтенду застосовувалися CSS, JSP та шаблони FreeMarker. Інтерфейси користувача включають в себе головну сторінку, сторінки реєстрації та входу, перегляд контенту та менеджери категорій і продукції.

Дана веб-платформа має потенціал для розвитку та вдосконалення, завдяки використанню технологій, таких як MySQL, JAVA, CSS, JSP та FreeMarker, платформа може адаптуватися до змін ринку та вимог користувачів. Архітектура, побудована на основі клієнт-серверної моделі та MVC, що забезпечує її гнучкість та можливість масштабування, що дозволяє платформі інтегрувати нові можливості, такі як підтримка нових форматів відео, інтеграція з іншими сервісами та розширення можливостей для розробників.

До переваг системи можна віднести: що веб-платформа включає всі необхідні функції для повноцінної роботи онлайн-кінотеатру, а саме, аутентифікація, пошук відеоматеріалів, оплата та перегляд контенту. Також до переваг можна віднести, використання криптографічних ключів, що забезпечує надійне зберігання даних користувачів. Ще однією перевагою є гнучкість і масштабованість архітектура MVC та клієнт-серверна модель, що дозволяють додавати нові функції та масштабувати систему. До переваг також варто віднести і зручність веб-платформи для користувачів, інтуїтивно зрозумілий інтерфейс користувача, який включає головну сторінку, сторінки реєстрації, входу, перегляд контенту, менеджери категорій і продукції.

Недоліки веб-платформи є: складність розробки та підтримки використання різних технологій та фреймворків, що ускладню процес розробки та підтримки платформи. Також веб-платформа вибаглива до ресурсів, тому для забезпечення надійної роботи та високої продуктивності платформа потребує великих обчислювальних ресурсів. Ще одним недоліком веб-платформи є залежність від технологій, що може призвести до проблем із сумісністю та оновленням системи.

Отже, розроблена веб-платформа для онлайн-кінотеатру має перспективи та переваги, проте для її успішного розвитку необхідно враховувати і потенційні недоліки, забезпечуючи підтримку та оновлення системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1) Онлайн кінотеатр [Електронний ресурс] – Режим доступу до ресурсу: <http://uzinform.com.ua/news/2014/01/05/30109.html>.
- 2) Порівнюємо Netflix, Amazon Prime Video, Apple TV+, Megogo і Sweet.TV [Електронний ресурс] – Режим доступу до ресурсу: <https://mind.ua/publications/20254602-shcho-vibrati-dlya-domashnogo-pereglyadu-v-ukrayini-porivnyuemo-netflix-amazon-prime-video-apple-tv-me>.
- 3) Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: <https://www.netflix.com/ua/>.
- 4) Легальні онлайн-кінотеатри [Електронний ресурс] – Режим доступу до ресурсу: <https://nv.ua/ukr/techno/it-industry/de-divitisya-filmi-onlayn-legalno-50075319.html>.
- 5) Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: <https://megogo.net/ua>.
- 6) Apple TV+ [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Apple_TV%2B.
- 7) Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: <https://tv.apple.com/ua>.
- 8) sweet.tv [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Sweet.tv>.
- 9) Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: <https://sweet.tv/uk>.
- 10) Система керування базами даних MySQL [Електронний ресурс] – Режим доступу до ресурсу: <http://www.znannya.org/?view=mysql>.
- 11) 1.3. Основні переваги субд MySql [Електронний ресурс] – Режим доступу до ресурсу: <https://studfile.net/preview/5607354/page:3/>.
- 12) База даних MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://promoter.net.ua/articles/baza-danix-mysql.html>.
- 13) ЩО ТАКЕ MYSQL ЯК І ДЕ ВИКОРИСТОВУЮТЬ MYSQL [Електронний ресурс] – Режим доступу до ресурсу: <http://ruszura.in.ua/ihry-i-ihrovi-konsoli/scho-take-mysql-yak-i-de-vykorystovuyut-mysql.html>.

- 14) Чому СУБД MySQL стала головною у всесвітній павутині [Електронний ресурс] – Режим доступу до ресурсу: <http://yoip.com.ua/chomu-subd-mysql-stala-golovnoyu-u-vsesvitnij-pavutini/>.
- 15) Spring Boot [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.spring.io/spring-boot/index.html>.
- 16) What is Rapid Application Development (RAD)? [Електронний ресурс] – Режим доступу до ресурсу: <https://codebots.com/app-development/what-is-rapid-application-development-rad>.
- 17) ЯК ВИГЛЯДАЄ СПЕЦИФІКАЦІЯ ВИМОГ І ЯК ЇЇ ТЕСТУВАТИ [Електронний ресурс] – Режим доступу до ресурсу: <https://training.qatestlab.com/blog/technical-articles/what-the-requirements-specification-looks-like-and-how-to-test-it/>.
- 18) JAVA [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Java>.
- 19) Історія та особливості Oracle Java [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/orakl-dzhava/>.
- 20) Освоюємо Java/Загальний огляд та історія Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikibooks.org/wiki/%D0%9E%D1%81%D0%B2%D0%BE%D1%8E%D1%94%D0%BC%D0%BE_Java/%D0%97%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4_%D1%82%D0%B0_%D1%96%D1%81%D1%82%D0%BE%D1%80%D1%96%D1%8F_Java.
- 21) Найпопулярніші мови програмування з'явилися 20+ років тому: чому вони досі у тренді? [Електронний ресурс] – Режим доступу до ресурсу: <https://robotdreams.cc/uk/blog/496-naypopulyarnishi-movi-programuvannya>.
- 22) CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://umity.in.ua/concept/?id=1401>.
- 23) Що таке CSS [Електронний ресурс] – Режим доступу до ресурсу: https://css.in.ua/article/shcho-take-html_10.
- 24) JSP в Java [Електронний ресурс] – Режим доступу до ресурсу: <https://antitutor.com.ua/jsp-%D0%B2-java/>.

- 25) Що таке JSP? Розбираємось з можливостями на практиці [Електронний ресурс] – Режим доступу до ресурсу: [https://javarush.com/ua/groups/posts/uk.2655.jshtake-jsp-rozbiramosjh-z-mozhливостями-na-practic](https://javarush.com/ua/groups/posts/uk.2655.jshtake-jsp-rozbiramosjh-z-mozhливostjami-na-practic).
- 26) https://uk.wikibooks.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BD%D0%B0_Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikibooks.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BD%D0%B0_Java.
- 27) FreeMarker [Електронний ресурс] – Режим доступу до ресурсу: <https://javarush.com/ua/quests/lectures/ua.questspring.level04.lecture29>.
- 28) What is Apache FreeMarker? [Електронний ресурс] – Режим доступу до ресурсу: <https://freemarker.apache.org/index.html>.
- 29) Створення проекту в IntelliJ IDEA [Електронний ресурс] – Режим доступу до ресурсу: <https://javarush.com/ua/groups/posts/uk.2311.stvorennja-proektu-v-intellij-idea>.
- 30) 10 корисних плагінів для IntelliJ IDEA: для новачків та досвідчених розробників [Електронний ресурс] – Режим доступу до ресурсу: <https://javarush.com/ua/groups/posts/uk.3146.10-korisnikh-plagnv-dlja-intellij-idea-dlja-novachkv-ta-dosvdchenikh-rozrobnikv>.
- 31) IntelliJ IDEA і Debug: не дайвінг, але снорклінг [Електронний ресурс] – Режим доступу до ресурсу: <https://javarush.com/ua/groups/posts/uk.621.intellij-idea--debug-ne-dayvng-ale-snorklng>.
- 32) Бібліотеки в програмуванні: потужний інструмент для створення чистого коду [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/shcho-take-biblioteka-v-prohramuvanni/>.

ДОДАТКИ

ДОДАТОК А

Програмний код контролерів.

CategoryManagerController:

```
package ua.com.vl.vidlib2021.conrollers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import ua.com.vl.vidlib2021.entity.Category;
import ua.com.vl.vidlib2021.repository.ProductRepository;
import ua.com.vl.vidlib2021.service.CategoryService;

@Controller
public class CategoryManagerController {

    private final CategoryService categoryService;
    private final ProductRepository productService;

    @Autowired
    public CategoryManagerController(CategoryService categoryService, ProductRepository
productService) {
        this.categoryService = categoryService;

        this.productService = productService;
    }

    @GetMapping("/create-category")
    public String getCategoryAdmin(Model model){
        model.addAttribute("allCategories", categoryService.getAllCategories());
        return "category_manager";
    }

    @PostMapping("saveCategory")
    public String saveNewCategory(@RequestParam(value = "name") String name,
                                @RequestParam(value = "desc") String description,
                                @RequestParam(value = "image") String image){
        categoryService.saveCategory(name, description, image);
        return "redirect:/create-category";
    }

    @PostMapping("updateCategory")
    public String updateCategory(@RequestParam(value = "id") Long id,
                                @RequestParam(value = "name") String name,
                                @RequestParam(value = "desc") String description,
                                @RequestParam(value = "image") String image){
```

```

        categoryService.updateCategory(id, name, description, image);
        return "redirect:/create-category";
    }

    @PostMapping("deleteCategory")
    public String deleteCategory(@RequestParam(value = "id") Long id){
        categoryService.deleteCategory(id);
        return "redirect:/create-category";
    }

    @PostMapping("deleteAllCategory")
    public String deleteAllCategory(){
        categoryService.deleteAll();
        return "redirect:/create-category";
    }

    @GetMapping("/videos/{id}")
    public String getVideo(@PathVariable("id") Category category,
        Model model){
        model.addAttribute("video", productService.findAllByCategories(category) );
        return "videos1";
    }

    @GetMapping("/index2")
    public String getIndex(Model model){
        model.addAttribute("AllCategory", categoryService.getAllCategories());
        return "index2";
    }
}

```

ClientManagerController:

```

package ua.com.vl.vidlib2021.conrollers;

public class ClientManagerController {
}

```

OrderManagerController:

```

package ua.com.vl.vidlib2021.conrollers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import ua.com.vl.vidlib2021.entity.Address;
import ua.com.vl.vidlib2021.entity.Order;

```

```

import ua.com.vl.vidlib2021.entity.ProductHasOrder;
import ua.com.vl.vidlib2021.repository.AddressRepository;
import ua.com.vl.vidlib2021.repository.OrderRepository;
import ua.com.vl.vidlib2021.repository.ProductHasOrderRepository;

import java.util.List;
import java.util.stream.Collectors;

@Controller
public class OrderManagerController {

    private final OrderRepository orderRepository;
    private final ProductHasOrderRepository productHasOrderRepository;
    private final AddressRepository addressRepository;

    @Autowired
    public OrderManagerController(OrderRepository orderRepository, ProductHasOrderRepository
productHasOrderRepository, AddressRepository addressRepository) {
        this.orderRepository = orderRepository;
        this.productHasOrderRepository = productHasOrderRepository;
        this.addressRepository = addressRepository;
    }

    @GetMapping("/update-order")
    public String getAllOrder(Model model){

        // model.addAttribute("order1", orderRepository.findAllByStatus(false));
        // model.addAttribute("order2", orderRepository.findAllByStatus(true));

        List<Order> orderList = orderRepository.findAll();

        List<Order> orders1 = orderList.stream().filter(p -> !p.getStatus()).collect(Collectors.toList());
        List<Order> orders2 = orderList.stream().filter( p-> p.getStatus()).collect(Collectors.toList());

        model.addAttribute("orderList1", orders1);
        model.addAttribute("orderList2", orders2);

        return "order_manager";
    }

    @GetMapping("/update-order/{id}")
    public String showOrder(@PathVariable("id") Order order,
        // @PathVariable("id") Long id,
        Model model){

        List<ProductHasOrder> productHasOrders =
productHasOrderRepository.findAllByOrders(order);
        model.addAttribute("orderList",productHasOrders);

        Address address = addressRepository.findAllByClientes(order.getCustomers()).get(0);

```

```

        model.addAttribute("address", address.getMy_address());

        return "order_id";
    }

    @PostMapping("updateOrder")
    public String updateOrder(@RequestParam(value = "id") Order order){

        //    if(!order.getStatus()) {
        //        order.setStatus(true);
        //    } else {
        //        order.setStatus(false);
        //    }

        order.setStatus(!order.getStatus() ? true : false);
        orderRepository.save(order);

        return "redirect:/update-order";
    }
}

```

ProductManagerController:

```

package ua.com.vl.vidlib2021.conrollers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import ua.com.vl.vidlib2021.entity.Category;
import ua.com.vl.vidlib2021.repository.ProductRepository;
import ua.com.vl.vidlib2021.service.CategoryService;
import ua.com.vl.vidlib2021.service.ProductService;

@Controller
public class ProductManagerController {

    private final ProductService productService;
    private final CategoryService categoryService;
    private final ProductRepository productRepository;

    @Autowired
    public ProductManagerController(ProductService productService, CategoryService
categoryService, ProductRepository productRepository) {
        this.productService = productService;
    }
}

```

```

        this.categoryService = categoryService;
        this.productRepository = productRepository;
    }

```

```

// pagination
@GetMapping("/products")
public String getProducts(
    @PageableDefault(sort = {"id"},
        direction = Sort.Direction.DESC, size = 4) Pageable page,
    Model model)
{

    model.addAttribute("all_product", productRepository.findAll(page));
    model.addAttribute("url", "/products");
    return "products";
}

```

```

@GetMapping("/create-product")
public String getAllProduct(
    @PageableDefault(sort = {"price"}, direction = Sort.Direction.ASC, size = 8) Pageable page,
    Model model){

    model.addAttribute("url", "/create-product");
    model.addAttribute("all_product", productRepository.findAll(page));
    model.addAttribute("allCategory", categoryService.getAllCategories());

    return "product_manager";
}

```

```

@PostMapping("saveProduct")
public String saveProduct(@RequestParam(value = "name") String name,
    @RequestParam(value = "desc") String description,
    @RequestParam(value = "price") String price,
    @RequestParam(value = "category_id") Category category,
    @RequestParam(value = "image") String image,
    @RequestParam(value = "video") String video
){
    productService.saveProduct(name, description, image, video, Double.valueOf(price), category);
    return "redirect:/create-product";
}

```

```

@PostMapping("updateProduct")
public String updateProduct(@RequestParam(value = "id") Long id,
    @RequestParam(value = "name") String name,
    @RequestParam(value = "desc") String description,
    @RequestParam(value = "price") String price,
    @RequestParam(value = "category_id") Category category,
    @RequestParam(value = "image") String image,

```

```

        @RequestParam(value = "video") String video
    ){
        productService.updateProduct(id, name, description, image, video, Double.valueOf(price),
category);
        return "redirect:/create-product";
    }

    @PostMapping("deleteProduct")
    public String deleteProduct(@RequestParam(value = "id") Long id){
        productService.deleteProduct(id);
        return "redirect:/create-product";
    }

    @PostMapping("deleteAllProduct")
    public String deleteProducts(){
        productService.deleteAll();
        return "redirect:/create-product";
    }
}

```

UserController:

```

package ua.com.vl.vidlib2021.conrollers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import ua.com.vl.vidlib2021.entity.Address;
import ua.com.vl.vidlib2021.entity.Clients;
import ua.com.vl.vidlib2021.entity.Users;
import ua.com.vl.vidlib2021.repository.AddressRepository;
import ua.com.vl.vidlib2021.repository.ClientRepository;
import ua.com.vl.vidlib2021.repository.UsersRepository;
import ua.com.vl.vidlib2021.service.UsersService;

import javax.validation.Valid;

@Controller
public class UserController {

    private final UsersService usersService;
    private final UsersRepository usersRepository;
    private final ClientRepository clientRepository;
    private final AddressRepository addressRepository;

    @Autowired

```

```

    public UserController(UsersService usersService, UsersRepository usersRepository,
ClientRepository clientRepository, AddressRepository addressRepository) {
        this.usersService = usersService;
        this.usersRepository = usersRepository;
        this.clientRepository = clientRepository;
        this.addressRepository = addressRepository;
    }

    @GetMapping("/registration")
    public String getMyNewFormLogin(Model model) {

        model.addAttribute("users", new Users());
        model.addAttribute("clients", new Clients());
        model.addAttribute("address", new Address());

        return "registration";
    }

    @PostMapping("registration")
    public String addNewUser(
        @Valid Users users,
        BindingResult bindingResult,
        @Valid Clients clients,
        BindingResult bindingResult1,
        @Valid Address address,
        BindingResult bindingResult2,
        Model model) {

        if (bindingResult.hasErrors()) {
            return "registration";
        }

        if (bindingResult1.hasErrors()) {
            return "registration";
        }

        if (bindingResult2.hasErrors()) {
            return "registration";
        }

        if (!users.getPassword().equals(users.getPasswordConfirm())) {
            model.addAttribute("passwordError", "Паролі не співпадають");
            return "registration";
        }

        if (usersService.getUsersFromDB(users)) {
            model.addAttribute("userNameError", "Користувач з таким логіном вже існує в БД");
            return "registration";
        }

        usersService.saveUser(users);
    }

```

```

Users user = usersRepository.findByUsername(users.getUsername());

clients.setUsers(user);
Clients client = clientRepository.save(clients);
address.setClientes(client);
addressRepository.save(address);

return "redirect:/login";
}

@RequestMapping(value = "/login", method = RequestMethod.GET)
public String login(Model model, String error, String logout) {
    if (error != null) {
        model.addAttribute("error", "Username or password is incorrect.");
    }

    if (logout != null) {
        model.addAttribute("message", "Logged out successfully.");
    }

    return "login";
}
}

```

ДОДАТОК Б

Програмний код класів сутностей.

Address:

```
package ua.com.vl.vidlib2021.entity;

import lombok.*;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.Size;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table(name = "fix_address")
public class Address {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotEmpty
    private String my_address;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private Clients clientes;
}
```

Category:

```
package ua.com.vl.vidlib2021.entity;

import lombok.*;

import javax.persistence.*;
import java.util.List;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
```

```

@Entity
@Table(name = "category")
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String description;
    private String image;

    @OneToMany(mappedBy = "categories")
    private List<Product> productList;

    public Category(String name, String description, String image) {
        this.name = name;
        this.description = description;
        this.image = image;
    }
}

```

Clients:

```

package ua.com.vl.vidlib2021.entity;

import lombok.*;

import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
import java.util.List;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString

@Entity
@Table(name = "clients")
public class Clients {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;
    @NotEmpty
    @Email
    private String email;
}

```

```

private String phone;
private int age;

@OneToMany(mappedBy = "customers")
private List<Order> orderList;

@OneToMany(mappedBy = "clientes")
private List<Address> addresses;

@OneToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "users_id")
private Users users;
}

```

Order:

```

package ua.com.vl.vidlib2021.entity;

import lombok.*;

import javax.persistence.*;
import java.util.Date;
import java.util.List;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString

@Entity
@Table(name = "fix_order")
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "create_date")
    private Date createDate;

    private Boolean status;

    private String delivery;
    private String payment;

    @ManyToOne
    @JoinColumn(name = "client_id")
    private Clients customers;
}

```

```
@OneToMany(mappedBy = "orders")
private List<ProductHasOrder> productHasOrderList;

}
```

Product:

```
package ua.com.vl.vidlib2021.entity;

import lombok.*;

import javax.persistence.*;
import java.math.BigDecimal;
import java.util.List;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString

@Entity
@Table(name = "product")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String description;
    private BigDecimal price;
    private String image;
    private String video;

    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category categories;

    @OneToMany(mappedBy = "products")
    private List<ProductHasOrder> productHasOrders;

}
```

ProductHasOrder:

```
package ua.com.vl.vidlib2021.entity;

import lombok.*;

import javax.persistence.*;
```

```

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString

@Entity
@Table(name = "product_has_order")
public class ProductHasOrder {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private int quantity;

    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product products;

    @ManyToOne
    @JoinColumn(name = "order_id")
    private Order orders;
}

```

Role:

```

package ua.com.vl.vidlib2021.entity;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.Transient;

import javax.persistence.*;
import java.util.Set;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

@Transient
@Entity
@Table(name = "fix_role")
public class Role implements GrantedAuthority{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

```

```

private String name;

public Role(Long id, String name) {
    this.id = id;
    this.name = name;
}

// List
// Set

@ManyToMany(mappedBy = "roles")
private Set<Users> userSet;

@Override
public String getAuthority() {
    return getName();
}
}

```

Users:

```

package ua.com.vl.vidlib2021.entity;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.Size;
import java.util.Collection;
import java.util.HashSet;
import java.util.Set;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table(name = "fix_user")
public class Users implements UserDetails{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotEmpty
    @Size(min=4)

```

```

private String username;
@NotEmpty
@Size(min=4)
private String password;

@Transient
@NotEmpty
@Size(min=4, max = 25)
private String passwordConfirm;

@ManyToMany(fetch = FetchType.EAGER)
private Set<Role> roles = new HashSet<>();

@OneToOne(mappedBy = "users", fetch = FetchType.EAGER)
private Clients clients;

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return getRoles();
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return true;
}

@Override
public String getUsername(){
    return username;
}

@Override
public String getPassword(){
    return password;
}
}

```

ДОДАТОК В

Результат перетворення ER-моделі в модель даних СКБД MySQL.

Згенерований код:

```
CREATE TABLE `category` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `description` varchar(255) DEFAULT NULL,  
  `image` varchar(255) DEFAULT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM AUTO_INCREMENT=9 DEFAULT CHARSET=utf8
```

```
CREATE TABLE `clients` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `age` int NOT NULL,  
  `email` varchar(255) DEFAULT NULL,  
  `first_name` varchar(255) DEFAULT NULL,  
  `last_name` varchar(255) DEFAULT NULL,  
  `phone` varchar(255) DEFAULT NULL,  
  `users_id` bigint DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `FKoo3grf91x7kf8jqx59hyxavdb` (`users_id`)  
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
```

```
CREATE TABLE `fix_address` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `my_address` varchar(255) DEFAULT NULL,  
  `user_id` bigint DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `FK14112orl2ohc1r58d999ec51l` (`user_id`)  
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
```

```
CREATE TABLE `fix_order` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `create_date` datetime DEFAULT NULL,  
  `delivery` varchar(255) DEFAULT NULL,  
  `payment` varchar(255) DEFAULT NULL,  
  `status` bit(1) DEFAULT NULL,  
  `client_id` bigint DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `FKb3g2elv6y0emaqkudwcnwouu2` (`client_id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

```
CREATE TABLE `fix_role` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `name` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
```

```
CREATE TABLE `fix_user` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `name` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
```

```
`id` bigint NOT NULL AUTO_INCREMENT,  
`password` varchar(255) DEFAULT NULL,  
`username` varchar(255) DEFAULT NULL,  
PRIMARY KEY (`id`)  
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
```

```
CREATE TABLE `fix_user_roles` (  
  `user_set_id` bigint NOT NULL,  
  `roles_id` bigint NOT NULL,  
  PRIMARY KEY (`user_set_id`,`roles_id`),  
  KEY `FKpg7cutj93x5qbunn1i5v4ju3r` (`roles_id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

```
CREATE TABLE `product` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `description` varchar(255) DEFAULT NULL,  
  `image` varchar(255) DEFAULT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  `price` decimal(19,2) DEFAULT NULL,  
  `video` varchar(255) DEFAULT NULL,  
  `category_id` bigint DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `FK1mtsbur82frn64de7balymq9s` (`category_id`)  
) ENGINE=MyISAM AUTO_INCREMENT=9 DEFAULT CHARSET=utf8
```

```
CREATE TABLE `product_has_order` (  
  `id` bigint NOT NULL AUTO_INCREMENT,  
  `quantity` int NOT NULL,  
  `order_id` bigint DEFAULT NULL,  
  `product_id` bigint DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `FK41trdb727wmnxektf3mp9x7e5` (`order_id`),  
  KEY `FK7vgxccvusjj3s0afu0uiwv1r8` (`product_id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

ДОДАТОК Г

Програмний код веб-сторінок

Головна сторінка для не зареєстрованого користувача:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link href="static/css/style1.css" rel="stylesheet">
  <title>Title</title>
</head>
<body>
<div class="topnav">
  <a href="/index"><h3>Головна сторінка</h3></a>
  <a href="/payment"><h3>Оплата</h3></a>
  <a href="/info"><h3>Інформація</h3></a>
  <a href="/create-category"><h3>Менеджер категорії</h3></a>
  <a href="/create-product"><h3>Менеджер продукції</h3></a>
  <a href="/update-order"><h3>Менеджер замовлення</h3></a>
  <a href="/user-manager"><h3>Менеджер користувача</h3></a>
  <a href="/registration"><h3>Регістрація</h3></a>
  <a href="/login"><h3>Вхід</h3></a>
  <form align="center">
    <input name="search" type="text" placeholder="Пошук" aria-label="Search">
    <button type="submit">Пошук</button>
  </form>
</div>
<table align="center" border="0">
  <caption><h3>Популярні</h3></caption>
  <tr><td></td><td></td><td></td><td></td></tr>
  <tr><td></td><td></td><td></td><td></td></tr>
</table>
</body>
</html>
```

Сторінка входу:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```

    <link href="static/css/style1.css" rel="stylesheet">
    <title>Login</title>
</head>
<body>
<#include "parts/navbar.ftl">
<h3>Вхід</h3>

    <form action="login" method="post">
        <input type="text" name="username" placeholder="Username" value="www">
        <input type="password" name="password" placeholder="password" value="qqqq">
        <input type="submit" value="add">

        <a href="/registration"><h2>Реєстрація</h2></a>
    </form>

</body>
</html>

```

Сторінка реєстрації:

```

<#import "/spring.ftl" as spring />
<!DOCTYPE html>
<html lang="en"><head>
    <meta charset="UTF-8">
    <link href="static/css/style1.css" rel="stylesheet">
    <title>Registration</title></head><body>
<#include "parts/navbar.ftl">
<h2> Реєстрація </h2>
<@spring.bind "users"/>
<form action="registration" method="post">
    <h3>Логін</h3> <br>
    <@spring.formInput "users.username"/><br>
    <@spring.showErrors "<br>"/> <br>
    <#if userNameError??>
        ${userNameError}
    </#if>
    <h3>Пароль</h3> <br>
    <@spring.formInput "users.password"/><br>
    <@spring.showErrors "<br>"/> <br>
    <h3>Повторіть пароль</h3> <br>
    <@spring.formInput "users.passwordConfirm"/><br>
    <@spring.showErrors "<br>"/> <br>
    <#if passwordError??>
        ${passwordError}
    </#if>
    <@spring.bind "clients"/>
    <h3>Ім'я</h3> <br>
    <@spring.formInput "clients.firstName"/><br><br>
    <h3>Фамілія</h3> <br>
    <@spring.formInput "clients.lastName"/><br><br>
    <h3>Електронна пошта</h3> <br>
    <@spring.formInput "clients.email"/><br>

```

```

<@spring.showErrors "<br>"/> <br>
<h3>Телефон</h3> <br>
<@spring.formInput "clients.phone"/><br><br>
<h3>Вік</h3> <br>
<@spring.formInput "clients.age" /><br><br>
<@spring.bind "address"/>
<h3>Адреса</h3> <br>
<@spring.formInput "address.my_address"/><br>
<@spring.showErrors "<br>"/> <br>
<input type="submit" value="add"></form></body></html>

```

Головна сторінка для зареєстрованого користувача:

```

<#import "parts/basis_ftl.ftl" as c>

<@c.page>
  <h2 align="center">Популярні</h2>
  <#list AllCategory as category>
    <a href="/videos/${category.id}">
      
    </a>
  </#list>
</@c.page>

```

Перегляд контенту:

```

<#import "parts/basis_vid.ftl" as c>

<@c.page>
  <#if video??>
    <#list video as vid>
      <table align="center">
        <tr>
          <td>
            <h2 align="center">${vid.name}</h2>
          </td>
        </tr>
        <tr>
          <td>
            <video controls="w" name="media" poster="${vid.image}" width="854"
height="480">
              <source align="center" src="/video/${vid.name}.mp4" type="video/mp4">
            </video>
          </td>
        </tr>
        <tr>
          <td>
            <h3 align="center">${vid.description}</h3>
          </td>
        </tr>
      </table>
    </#list>
  </#if>
</@c.page>

```

```

        </tr>
    </table>
</#list>
</#if>
</@c.page>

```

Сторінка менеджера категорій:

```

<#import "parts/basis_ftl.ftl" as c>
<@c.page>
    <h2 xmlns="http://www.w3.org/1999/html">Всі категорії</h2>
    <#if allCategories??>
        <#list allCategories as category>
            <table>
                <tr>
                    <td><h3>${category.id}</h3></td>
                    <td><h3>${category.name}</h3></td>
                    <td><h3>${category.description}</h3></td>
                    <td><h3>${category.image}</h3></td>
                    <td></td>
                </tr>
            </table>
        </#list>
    </#if>
    <br>
    <h2>Збереження категорії</h2>
    <table>
        <form action="saveCategory" method="post">
            <tr>
                <td><label><h3>Ім'я</h3></label></td>
                <td><input type="text" name="name" placeholder="Ім'я"/></td>
            </tr>
            <tr>
                <td><label><h3>Опис</h3></label></td>
                <td><input type="text" name="desc" placeholder="Опис"/></td>
            </tr>
            <tr>
                <td><label><h3>Фото</h3></label></td>
                <td><input type="text" name="image" placeholder="Фото"/></td>
            </tr>
            <tr>
                <td><label><h3>Додати нову категорію</h3></label></td>
                <td><input type="submit" value="Додати нову категорію"></td>
            </tr>
        </form>
    </table>
    <h2>Видалення/Оновлення категорії</h2>
    <#if allCategories??>
        <#list allCategories as category>
            <table>
                <tr>

```

```

        <form action="updateCategory" method="post">
            <input type="hidden" name="id" value="{category.id}">
            <td><input type="text" name="Ім'я" value="{category.name}"></td>
            <td><input type="text" name="Опис" value="{category.description}"></td>
            <td><input type="text" name="Фото" value="{category.image}"></td>
            <td><input type="submit" value="Оновлення" name="button"></td>
        </form>

        <form action="deleteCategory" method="post">
            <input type="hidden" name="id" value="{category.id}">
            <td><input type="submit" value="Видалення"></td>
        </form>
    </tr>
</table>
</#list>
</#if>
<h2>Видалити все</h2>
<form action="deleteAllCategory" method="post">
    <input type="submit" value="Видалити Все">
</form>
</@c.page>

```

Сторінка менеджера продукції:

```

<#import "parts/basis_ftl.ftl" as c>
<@c.page>
<h2>Адміністрування продукції</h2>
<h3><#import "parts/pager.ftl" as p>
    <@p.pager url all_product /></h3><br>
<h2>Лист продукції</h2>
<table class="table table-bordered">
    <tr>
        <th scope="col"><h3>Ім'я</h3></th>
        <th scope="col"><h3>Опис</h3></th>
        <th scope="col"><h3>Категорія</h3></th>
        <th scope="col"><h3>Ціна</h3></th>
        <th scope="col"><h3>Фото</h3></th>
    </tr>
    <#if all_product.content??>
        <#list all_product.content as product>
            <tr>
                <td><h3>{product.name}</h3></td>
                <td><h3>{product.description}</h3></td>
                <td><h3>{product.categories.name}</h3></td>
                <td><h3>{product.price}</h3></td>
                <td><h3>{product.image}</h3></td>
            </tr>
        </#list>
    </#if>
</table>
<br>
<h2>Збереження продукції</h2>

```

```

<form action="saveProduct" method="post">
  <table class="table">
    <tr>
      <td><h3>Ім'я</h3></td>
      <td><input type="text" name="Ім'я"></td>
    </tr>
    <tr>
      <td><h3>Опис</h3></td>
      <td><input type="text" name="Опис"></td>
    </tr>
    <tr>
      <td><h3>Ціна</h3></td>
      <td><input type="text" name="Ціна"></td>
    </tr>
    <tr>
      <td><h3>Фото</h3></td>
      <td><input type="text" name="Фото"></td>
    </tr>
    <tr>
      <td><h3>Категорія</h3></td>
      <!--      <td><input type="text" name="category"></td>-->
      <td>
        <select name="category_id">
          <#if allCategory??>
            <#list allCategory as category>
              <#if "${category.id}"=="1">
                <option value="${category.id}" selected>${category.name}</option>
              <#else>
                <option value="${category.id}">${category.name}</option>
              </#if>
            </#list>
          </#if>
        </select>
      </td>
    </tr>
    <tr>
      <td></td>
      <td><input type="submit" value="Add"></td>
    </tr>
  </table>
</form>
<br>
<h2>Оновлення/Видалення продукції</h2>
<table class="table table-bordered">
  <tr>
    <th scope="col"><h3>Ім'я</h3></th>
    <th scope="col"><h3>Опис</h3></th>
    <th scope="col"><h3>Ціна</h3></th>
    <th scope="col"><h3>Фото</h3></th>
    <th scope="col"><h3>Категорія</h3></th>
    <th scope="col"><h3>Оновлення</h3></th>
    <th scope="col"><h3>Видалення</h3></th>
  </tr>

```

```

</tr>
<#if all_product.content??>
<#list all_product.content as product>
<tr>
  <form action="updateProduct" method="post">
    <input type="hidden" name="id" value="{product.id}">
    <td><input type="text" name="Ім'я" value="{product.name}"></td>
    <td><input type="text" name="Опис" value="{product.description}"></td>
    <td><input type="text" name="Ціна" value="{product.price}"></td>
    <td><input type="text" name="Фото" value="{product.image}"></td>
    <#--      Список категорій      --><td>
      <select name="category_id">
        <#if allCategory??>
        <#list allCategory as category>
        <#if "{category.id}"=="{product.categories.id}">
        <option value="{category.id}" selected>{category.name}</option>
        <#else>
        <option value="{category.id}">{category.name}</option>
        </#if>
        </#list>
        </#if>
      </select>
    </td>
    <td><input type="submit" value="update"></td>
  </form>
  <form action="deleteProduct" method="post">
    <input type="hidden" name="id" value="{product.id}">
    <td><input type="submit" value="delete"></td>
  </form>
</tr>
</#list>
</#if>
</table>
</form><br>
<h2>Видалення продукції</h2>
<form action="deleteAllProduct" method="post">
  <input type="submit" value="Видалити все">
</form><br>
<h3><@p.pager url all_product /></h3>
</@c.page>

```

Ім'я користувача:
Інформаційних систем в економіці Шкуратовська Те...

ID перевірки:
1016352921

Дата перевірки:
12.06.2024 22:38:24 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.06.2024 11:09:42 EEST

ID користувача:
100005745

Назва документа: Барабаш_O_IN_404

Кількість сторінок: 87 Кількість слів: 8883 Кількість символів: 71523 Розмір файлу: 15.29 MB ID файлу: 1016156759

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

9.34%

Схожість

Найбільша схожість: 1.73% з джерелом з Бібліотеки (ID файлу: 1015248318)

4% Джерела з Інтернету

278

Сторінка 89

8.38% Джерела з Бібліотеки

565

Сторінка 91

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%

Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи

1

Підозріле форматування

16
сторінок