ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра системного аналізу

Пояснювальна записка

до кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: **«СИСТЕМНИЙ АНАЛІЗ ЕЛЕКТРОННОГО СЕРВІСУ ПІДБОРУ ГАРДЕРОБУ»**

Виконав: студент 4 курсу, групи САД-41
спеціальності 124, Системний аналіз
Когут Д.Р.

Керівник Золотухіна О.А.
(прізвище, ініціали)

Рецензент _____
(прізвище, ініціали)

Нормоконтроль Ставицька Ю.В.
(прізвище, ініціали)

Київ – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Системного аналізу
Ступінь вищої освіти бакалавр
Спеціальність 124, Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедри
системного аналізу
_____
_____
_____ О.А.Золотухіна
“____” _____2020 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Когут Дмитро Русланович
(прізвище, ім’я, по батькові)

1. Тема роботи: **Системний аналіз електронного сервісу підбору гардеробу**

Керівник роботи Золотухіна Оксана Анатоліївна, кандидат технічних наук
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від“ _____” 2020 року №____
2. Строк подання студентом роботи
3. Вхідні дані до роботи: Науково технічна література з питань, пов’язаних з побудовою інформаційних систем

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Аналіз бізнес-процесів з підбору гардеробу
4.2 Визначення недоліків

4.3 Розробка нової моделі взаємодії стиліста та користувачів
4.4 Розробка бази даних електронного сервісу підбору гардеробу

5. Перелік графічного матеріалу
1. Об’єкт, предмет, мета бакалаврської роботи
2. Актуальність бакалаврської роботи
3. Завдання бакалаврської роботи
4. Контрактна модель взаємодії
5. Схема бази даних електронного сервісу
6. Апробація роботи
7. Висновок

6. Дата видачі завдання 08.05.2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз науково-технічної літератури	01.03.2020	
2	Опис предметної області	15.03.2020	
3	Постановка задачі	12.04.2020	
4	Аналіз бізнес-процесів з підбору гардеробу	20.04.2020	
5	Визначення недоліків та проблемних місць	05.05.2020	
6	Розробка нової моделі взаємодії користувачів та стилістів	15.05.2020	
7	Розробка бази даних	31.05.2020	
8	Розробка обов’язкових демонстраційних матеріалів	02.06.2020	
9	Попередній захист роботи	04.06.2020	
10	Здача роботи в деканат		

Студент _____ Когут Д.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Золотухіна О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 65 с., 17 рис., 11 джерел.

ЕЛЕКТРОНИЙ СЕРВІС ПІДБОРУ ГАРДЕРОБУ, WEB-СЕРВІСИ, БАЗИ ДАНИХ, REST-АРХІТЕКТУРА, МІКРОСЕРВІСИ

Об’єкт дослідження – системний аналіз електронного сервісу підбору гардеробу.

Предмет дослідження – удосконалена модель електронного сервісу підбору гардеробу.

Мета роботи – розробка моделі електронного сервісу підбору гардеробу використовуючи контрактну основу для взаємодії.

У роботі проведено аналіз методів розробки електронних сервісів із використанням досягнень у сфері Web-технологій. Проведено аналіз аналогів електронного сервісу підбору гардеробу використовуючи інформацію із електронних ресурсів та запропоновано нову бізнес-модель для взаємодії користувача із електронним сервісом.

На основі аналізу сучасних практик для нової бізнес-моделі розроблено архітектуру електронного сервісу підбору гардеробу, описано процеси взаємодії користувача із сервісом використовуючи UML-діаграми послідовності та реалізовано схему бази даних.

ЗМІСТ

Стор.

ВСТУП

Розвиток інформаційних технологій надзвичайно змінив повсякдений образ життя сучасної людини. Те що декілька десятиліть тому люди й уявити не могли сьогодні стало частиною нашого звичного життя. Компаній введуть бухгалтерський облік використовуючи спеціалізоване програмне забезпечення, замість паперових документів використовують системи електронного документообігу, а спілкування відбувається у соціальних мережах. В реальному світі для вирішення питання «Що мені одягти сьогодні?» чи «Що мені вдягнути щоб виглядати привабливо?» люди звертаються за консультацією до стилістів. Проте цей аспект людського життя також адаптований до використання в ньому сучасних технологій, що в загальному називається електронним сервісом підбору гардеробу.

Електронний сервіс підбору гардеробу вирішує проблему із якою стикається кожна людина коли думає про те, що одягнути і як зіставити одяг таким чином щоб виглядати стильно і респектабельно. Для вирішення цієї проблеми вже існують аналоги проте усі вони використовують бізнес-модель, що не завжди може бути зручною для їх клієнта.

У даній бакаларській роботі проведено аналіз сучасних практик і методів розробки електронних сервісів та їх адаптація для розробки електронного сервісу підбору гардеробу із використанням контрактної моделі взаємодії стилістів та клієнтів, на противагу бізнес-моделі підписки в існуючих аналогах.

Робота над даною роботою є актуальною, оскільки недоліком існуючих рішень є модель роботи із клієнтами – підписка на послуги компаній, що забирає у клієнтів прозорість та гнучкість у роботі із стилістами.

Об’єктом дослідження даної роботи є процес підбору гардеробу із використанням електронних сервісів.

Предметом дослідження виступають моделі та технології електронного сервісу підбору гардеробу

Мета роботи передбачає розробку пропозиції щодо удосконалення електронного сервісу підбору гардеробу за рахунок впровадження нової моделі взаємодії клієнта та стиліста.

Для досягнення поставленої мети потрібно виконати низку завдань:

- аналіз аналогічних рішень бізнес-процесів з підбору гардеробу;
- визначення недоліку бізнес-процесу;
- розробка нового бізнес-процесу з підбору гардеробу;
- розробка архітектури інформаційної системи із використанням нової бізнес-моделі моделі;
- реалізація бази даних для нової моделі.

1 ЕЛЕКТРОННИЙ СЕРВІС ПІДБОРУ ГАРДЕРОБУ

1.1 Огляд концепції електронного сервісу підбору гардеробу

Електронний сервіс підбору гардеробу вирішує проблему із якою стикається кожна людина коли думає про те, що одягнути і як зіставити одяг таким чином щоб виглядати стильно. Класичний спосіб вирішення даного питання це консультація із стилістом чи дизайнером. Проте ця консультація із спеціалістом не обов'язково буде швидкою адже для того щоб консультація відбулася потрібно попередньо домовитися, знайти час на зустріч, спланувати зустріч таким чином щоб усім було зручно зустрітися, тобто велика кількість часу витрачається просто на те щоб зустрітися, а це ще навіть не початок безпосереднього перетворення. Використовуючи сучасні технології увесь процес консультації можна пришвидшити, навіть змінити сам спосіб проведення консультацій створивши електронний-сервіс, що буде з'єднувати стилістів та їх клієнтів.

У випадку електронного сервісу можна пропустити процес безпосередньої консультації, натомість можна створити контрактну основу підбору одягу із існуючих елементів гардеробу. Суть контрактного способу в тому, що користувачі, використовуючи можливості сервісу, шукають стиліста та пропонують йому взятися за роботу «перетворення» свого образу. Відповідно, для того щоб користувач зміг вибрати до якого стиліста звернутися, стилістам потрібно завойовувати прихильність показавши свої роботи. Для цього у кожного стиліста є своєму профілі є портфоліо – альбоми образів чи колекції фотографій, що були створені для користувачів.

Отже, користувач, переглянувши роботи стилістів має змогу вибрати того чиї роботи йому найбільше сподобалися. Вибравши стиліста, користувач пропонує йому взятися за контракт – підібрати образ для клієнта. Контракт, в контексті сервісу, це повідомлення, що надсилається стилісту та може містити окрім текстового посилання колекцію особистих фотографій, для візуального

ознайомлення із клієнтом. Отримавши контракт, стиліст вирішує чи братися йому за його виконання. Якщо стиліст береться за виконання контракту, він підтверджує його натиснувши відповідну кнопку в графічному інтерфейсі.

Процес вибору стиліста та створення контракту користувачем, описану у вигляді UML діаграми діяльності, можна побачити на рисунку 1.1.

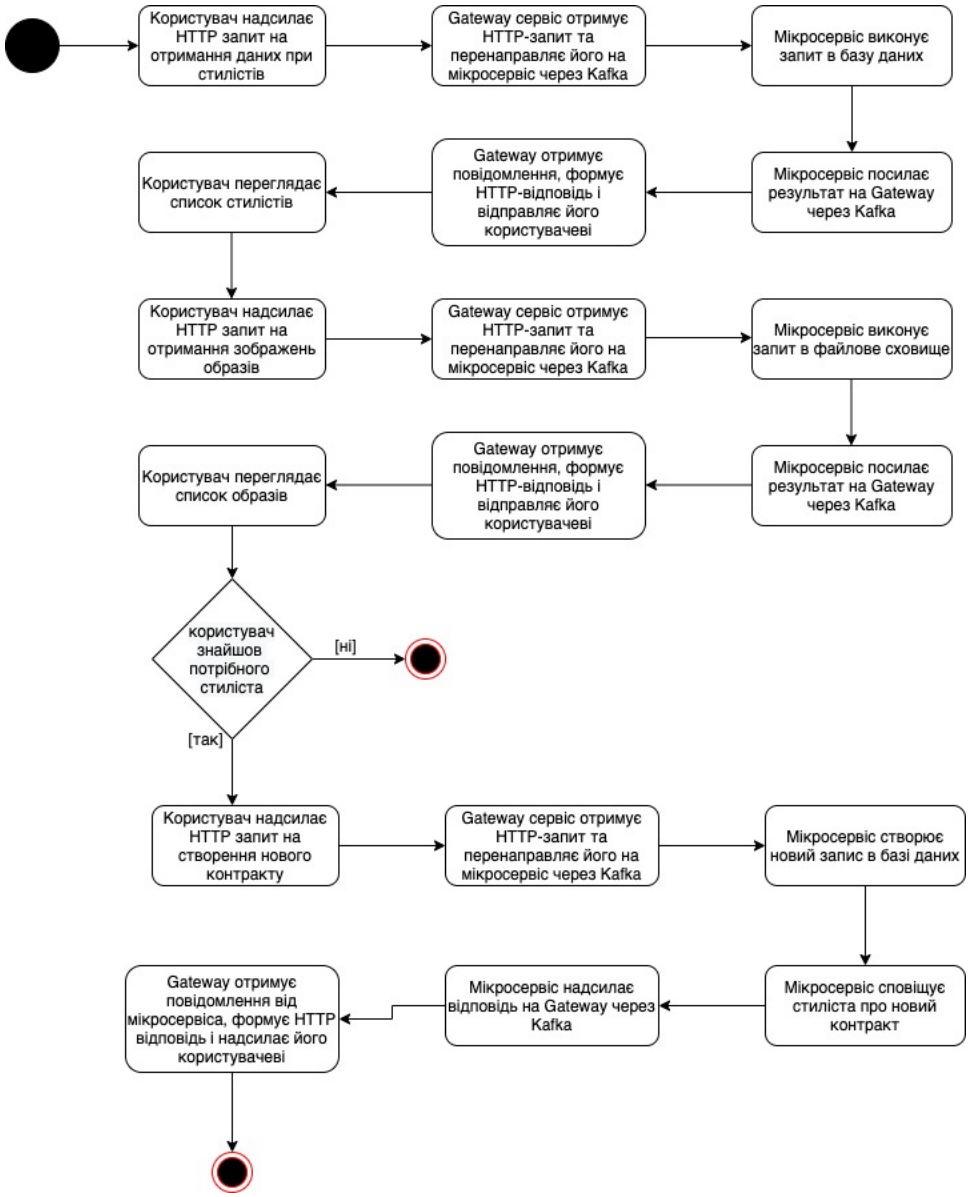


Рисунок 1.1 – Процес вибору стиліста та створення контракту

Після підтвердження контракту, клієнтові приходить відповідне сповіщення після чого обидві сторони зв’язуються використовуючи способи комунікацій

всередені сервісу чи використовуючи стороні засоби комунікації. Після виконання контракту обидві сторони контракту повинні підтвердити, що контракт закрито, для цього користувачеві та стилісту потрібно натиснути відповідну кнопку в графічному інтерфейсі сервісу. Існують декілька можливих варіантів закриття контракту.

Ініційований контракт – контракт був ініційований клієнтом, але ще не підтверджений стилістом. Контракт знаходиться в процесі формування, доки обидві сторони не досягнуть компромісу.

Відхилений контракт – контракт був ініційований клієнтом, але відхилений стилістом.

Контракт виконується – контракт був прийнятий до виконання стилістом.

Успішне виконання контракту – обидві сторони погоджуються, що контракт можна закрити і всі зобов’язання виконанні, тобто клієнт отримав бажаний результат і виконавець, тобто стиліст, отримав винагороду.

Неуспішне виконання контракту – обидві сторони погоджуються, що виконання контракту не можливе в результаті деяких подій. У такому випадку жодна із сторін не зобов’язана виконувати обумовлені зобов’язання контракту.

Конфліктний контракт – одна із сторін вирішила, що контракт виконаний, проте інша сторона із цим твердженням не згодна.

Невизначений стан – одна із сторін або обидві сторони контракту протягом тривалого часу не змінювали стану контракту, тобто контракт в процесі виконання дуже довгий час і обидві сторони не роблять для того щоб контракт був завершений.

Щоб зменшити кількість махінацій зі сторони клієнтів, даний електронний сервіс передбачає використання системи репутації засновану на статистичних даних виконання умов контракту зі сторони клієнта. Якщо стиліст виконавши роботу, згідно із обумовленими умовами, не отримав передбаченої контрактом винагороди він може поскаржитися на користувача сервісу. Ця скарга

відображатиметься в профілі користувача для того щоб стилісти мали змогу вирішувати чи варто працювати із даним клієнтом. У випадку регулярних скарг зі сторони стилістів, аккаунт користувача буде заблоковано.

Подібну систему репутацій також передбачено і для стилістів. Якщо стиліст виконав свою роботу не достатньо якісно і вимагає за виконання свої роботи винагороду, користувач також має право поскаржитися на стиліста. Так само як і у випадку із користувачем у профілі стиліста також будуть відображатися скарги користувачів поруч із статистикою виконаних робіт. Це потрібно щоб клієнт при виборі стиліста мав змогу оцінити відгуки та вирішити чи варто звертатися до спеціаліста.

Розробивши даний електронний сервіс можна досягти певних позитивних результатів:

- практично відсутні витрати часу на домовленості – даний електронний сервіс являється з'єднуючою ланкою між стилістами та їх безпосередніми клієнтами це дозволяє майже відразу знайти спеціаліста, що готовий допомогти, а контрактна система надання послуг дозволяє стилістові майже відразу почати роботу;
- розширений вибір спеціалістів – аналогічні сервіси не дають можливості вибору спеціаліста, що працюватиме над образом. Оскільки ці аналоги працюють із користувачами моделлю підписки самі компанії мають штат стилістів які працюють на компанію. Проте у випадку співпраці із стилістом використовуючи контракти – можна самим вибирати стилістів та ініціювати співпрацю;
- система рейтингу – для забезпечення кращого досвіду використання сервісу, та запобіганню шахрайства в сервісі використовується системи рейтингів для клієнтів та стилістів, ціль рейтингових систем – допомогти користувачам сервісу у виборі надійних партнерів у процесі роботи, а також блокування аккаунтів зловмисників;

Так як користувачі працюватимуть із електронним сервісом не менш важливою і вже класичною проблемою є питання доступу інформації та її конфіденційність. Для того щоб користувачам надавати доступ до потрібної їм інформації та в обмежувати їм доступ до інформації яку вони не повині отримати в даному сервісі використовується метод авторизації/аутентифікації із використанням JWT-токену [2]. JWT – це відкритий стандарт RFC 7519 [3], що використовується для безпечної передачі даних між двома акторами. Сам JWT-токен генерується та підписується секретним ключем на сервері після чого передається клієнтові, що клієнт надалі використовував цей токен для підтвердження своєї особи. Для створення JWT токену потрібно пройти 4 кроки:

- створити заголовок, заголовок (англ. – header) це перша із трьох частин токену, містить інформацію про те як потрібно вираховувати токен;
- створення корисної інформації (англ. – payload), це друга із трьох частин токену, містить в собі важливу для сервісу інформацію про користувачів;
- створення підпису (англ. – signature), це об’єкт, що вираховується із перших двух частин, спершу заголовок та корисна інформація кодуються в Base64 форматі, після чого обидві частини конкатенуються використовуючи символ крапка (‘.’), та підписуються секретним ключем;
- об'єднання всіх трьох компонентів в один токен. Це процес конкатенації всіх трьох частин, із використанням символу крапки як роздільника (‘.’).

Процес авторизації доволі простий і зображений у вигляді UML діаграми діяльності на рисунку 1.2. Основна перевага використання JWT-токену – це його самодостатність. Токен містить в собі всю потрібну інформацію про себе та про користувача, а отже нам не потрібно щоразу для аутентифікації користувача зчитувати дані із бази даних, натомість токен повинен надсилатися у кожному HTTP-запиті до сервісу в заголовку “Authorization”.

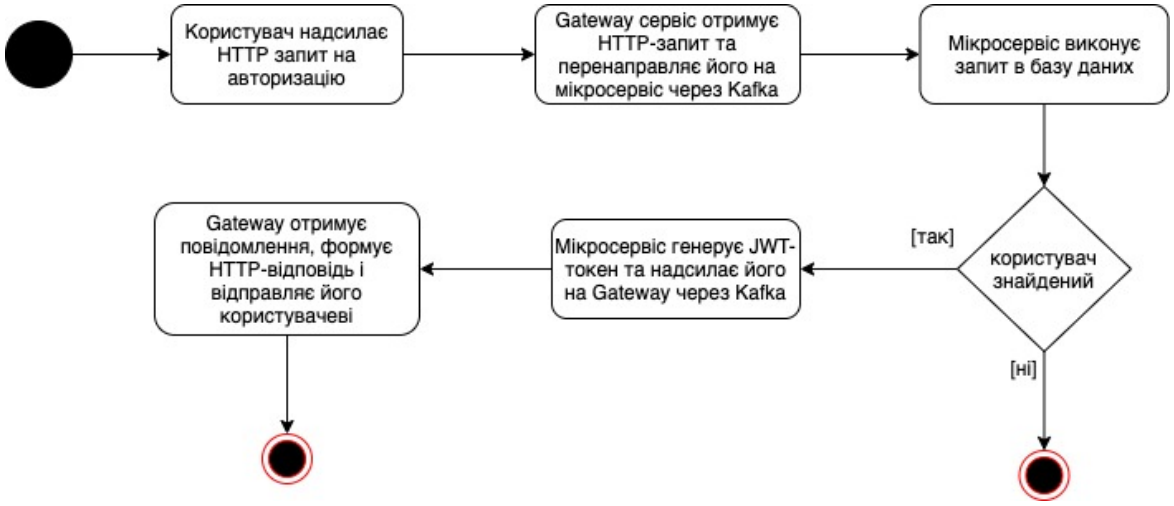


Рисунок 1.2 – Процес авторизації користувача

1.2 Огляд аналогів

1.2.1 Stitch Fix.

Stitch Fix – це персональний сервіс для стилізації, який надсилає окремо підібрані предмети одягу та аксесуарів за разову плату за послуги стиліста [10].

Процес роботи із сервісом Stitch Fix зводиться до наступних дій:

а) Клієнти заповнюють анкету в Інтернеті про свої переваги стилю.

б) Стиліст Stitch Fix вибирає п’ять предметів, які надсилає клієнтові. Предмети обираються на основі відповідей клієнта та будь-якого доступу, який клієнт надає їм у своїх соціальних мережах.

в) Клієнт планує дату отримання своїх предметів, яку називають «Виправлення» (Fix).

г) Після того, як відправка буде отримана, клієнт має три дні, щоб вибрати, чи зберігати вибрані для нього товари, або повернути деякі із них (або увесь набір одягу). Якщо клієнт зберігає щонайменше один товар, початкова плата за послуги стиліста зараховується до вартості вибраного товару. Окрім того, що нараховується плата за послуги стиліста, якщо клієнт вирішить зберегти всі п’ять предметів, він отримує 20% знижки від загальної вартості товарів.

ґ) Клієнти вибирають частоту доставки, наприклад, кожні два тижні, раз на місяць або кожні два місяці. Компанія також підтримує інтеграцію з дошками Pinterest, що дозволяє клієнтам додавати фотографії образів, які їм подобаються. Ці дошки можуть переглядати стилістом із Stitch Fix.

Компанія використовує науку про дані (Data Science) і поєднала персональних стилістів та машинне навчання для персоналізованих рекомендацій.

1.2.2 Stylogic

Stylogic – це сервіс по стилізації, яка доставляє підібрані вбрання до вашої дому. На основі профілю стилів користувача (Style Profile), стилісти компанії надсилають вам повний «набір». Ви можете зберегти весь наряд або зберегти вподобані вам частини наряду та відправити назад те, що вам не подобається.

Stylogic працює наступним чином:

а) Клієнт розповідає про себе, створюючи та заповнюючи свій профіль стилю (Style Profile) та завантажує свою фотографію в повний зріст. Це дозволяє стилістам ознайомитися із виглядом якого клієнт намагається досягти, типом та розміром тіла, бюджетом та способом життя. Далі компанія призначає стиліста щоб знайти вбрання, яке відповідає уподобанням клієнта.

б) Далі клієнт вибирає дату на яку хоче отримати свій Stylogic набір. Після того як стиліст збере наряд клієнтові, з клієнта буде стягнено комісію в розмірі 20 доларів. Ця плата використовується як кредит у випадку якщо клієнт залишає собі хоча б одну річ із підібраного набору. У випадку якщо клієнт залишає всі речі із набору він отримує 20% знижки на всю покупку. Клієнт може також оформити членство, це дозволяє клієнтові отримувати періодичні поставки одягу у вибрану клієнтом дату. Цю опцію можна увімкнути в профілі користувача і у будь який момент відключити її.

г) Клієнт зберігає речі які йому сподобалися та відправляє назад ті, що йому не підходять. Доставку в обидві сторони оплачує Stylogic.

ґ) Клієнт надсилає відгук про створений набір одягу (Stylogic Set). В своєму профілі користувач має змогу завантажити фотографії створеного образу. Написати коментарі та відгуки про те, що йому сподобалося, а що не сподобалося. Stylogic використовуватиме ці дані для подальшої роботи із клієнтами.

1.2.3 ModaBox

ModaBox – це сервіс персонального стилю, що працює за допомогою команди досвідчених стилістів. ModaBox – сервіс орієнтований на жіночий стиль одягу [11].

Modabox працює наступним чином:

а) Реєстрація. Клієнт створює свій профіль стилю, що дозволить стилісту-експерту знайти одяг, що відповідає тілу, стилю життя та особистості клієнта. Клієнт повідомляє сервіс про свій бюджет, стилі та кольори які найбільше подобаються клієнтові та про майбутні події в житті клієнта, такі як канікули чи побачення. В залежності від потреб клієнт вибирає тип послуг: Select, Premier чи Luxe. Плата стягується лише тоді коли стиліст почне створювати персоналізовану коробку.

б) Створення персоналізованої коробки. Базуючись на бюджеті і персональних перевагах стиліст Modabox створює ассортимент нарядів повністю адаптованих для клієнта, щоб клієнт приміряв на себе наряд не виходячи із дому.

в) «Спробуй вдома». Клієнт протягом встановленого часу (від 5 до 7) днів приміряє надіслані йому наряди не виходячи із дому чи офісу. У випадку якщо клієнт залишає у себе всі надіслані товари він отримує скидку. Ті товари, що клієнт залишає у себе він викреслює із чеку, інші речі клієнт складає назад у коробку із чеком та відсилає назад у Modabox.

Провівши аналіз даних сервісів можна прийти до висновку що дані сервіси надають послуги переважно у вигляді підписки на послуги сервісу. Беззаперечно підхід, що використовують дані компанії є прибутковим для самих компаній та можливо підходить багатьом користувачам. Проте очевидний недолік даного

підходу саме в тому що клієнти повинні щомісяця платити за підписку. Окрім того клієнти працюють із компанією, а не особисто із стилістом. Контрактний підхід описаний в розділі 1 даного звіту дозволяє з’єднувати клієнтів безпосередньо із стилістами. Клієнти самі обирають стилістів і працюють безпосередньо із ними.

Щоб зрозуміти переваги контрактної взаємодії та порівняти її із моделлю підписки, в таблиці 1.1 продемонстровані порівняльні характеристики цих моделей.

Таблиця 1.1 – Порівняльна характеристика моделі підписки та контрактної моделі

Характеристика	Модель підписки	Контрактна модель
Ціна	Користувачі зобов’язані поновлювати підписку із певним періодом, що може нести за собою непотрібні витрати.	Клієнти сервісу платять безпосередньо за те, що їм потрібно та коли їм потрібно, а не по тарифу компанії, пропозиція для клієнтів стає вигідною.
Гнучкість	Як і використання ресурсів, модель підписки не передбачає гнучкості у роботі із клієнтами. Відносини із клієнтами постійні, але обмежені.	Ціна та послуги стиліста залежить від умов укладеного контракту.

Продовження таблиці 1.1.

Прозорість	Клієнти працюють із компаніями без усвідомлення досвіду конкретного стиліста та без знання про його досвід який створює для них образи.	Клієнти самі мають здатність обирати стиліста стиль одягу якого їм підходить та із яким їм буде хотілося б працювати.

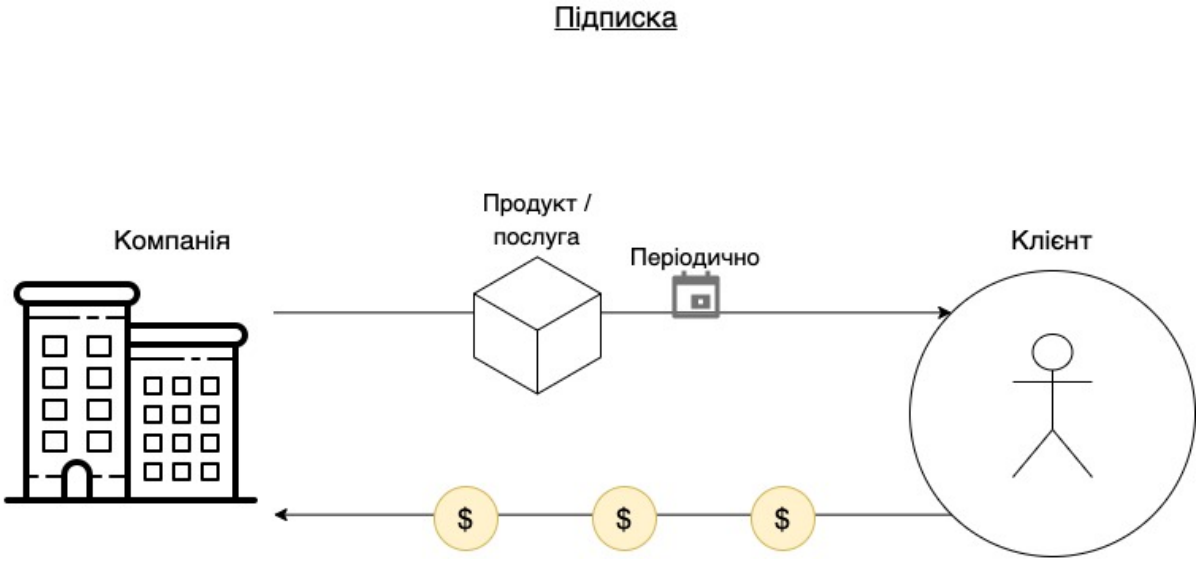


Рисунок 1.3 – Модель підписки

Для визначення ефективності електронного сервісу підбору гардеробу можна використовувати як загальні для всіх Web-сервісів метрики ефективності [1] так і унікальні для даного сервісу метрики. Серед загальних метрик можна визначити:

- час відгуку – це час за який електронний сервіс надсилає відповідь до клієнта на його запит. На різні запити час відгуку може відрізнятися, але в середньому цей показник не повинен перевищувати 1 секунди (завантаження

текстової інформації про користувача чи одяг), а на пікових навантаженнях цей показник не повинен перевищувати значення у 12 секунд (завантаження фотографії одягу у високій якості);

- кількість одночасно працюючих користувачів – цей показник відображає скільки користувачів можуть одночасно користуватися сервісом без негативних наслідків на сервіс. Дана метрика залежить повністю від реалізації системи, кількості працюючих серверів та визначається вона індивідуально для кожного сервісу в процесі тестування системи;
- кількість запитів в секунду – це кількість запитів, що здатний обробити електронний сервіс за одну секунду. Дана метрика визначається в процесі тестування сервісу;
- кількість транзакції в секунду – це метрика, що визначає скільки успішно завершених транзакцій (послідовність операцій об’єднаних в один логічний ланцюг) за секунду здатна виконати інформаційна система.

Метрики унікальні для даного електронного сервісу являються метриками, що відображають цінність данного сервісу як бізнесу. Серед них можна визначити:

- кількість відкритих контрактів – ця метрика данного сервісу відображає скільки протягом одного дня було відкрито нових контрактів (контрактів між клієнтом та стилістом);
- кількість успішно завершених контрактів за день – ця метрика відображає скільки за один день було закрито успішних контрактів;
- кількість неуспішно завершених контрактів – ця метрика відображає скільки протягом дня було закрито неуспішних контрактів;
- кількість створених образів – ця метрика відображає скільки протягом дня було створенно нових образів стилістами;
- кількість активних унікальних користувачів – ця метрика відображає популярність данного сервісу серед користувачів.

2 МОДЕЛЮВАННЯ СЕРВІСУ ПІДБОРУ ГАРДЕРОБУ

2.1 CASE-засоби моделювання інформаційних систем

Тенденції розвитку сучасних інформаційних систем приводять до постійного зростання їх складності і характеризуються вони, як правило, складністю опису, наявністю великої кількості взаємозв'язуючих компонентів та необхідністю інтеграції додатків, що вже існують. Для вирішення проблем проектування таких систем було розроблено та реалізовано структурний підхід до проектування інформаційних систем, особливо на етапі отримання первинної інформації про систему. Ці методології орієнтовані на системи, що функціонують у порівняно стаціонарному середовищі і не завжди пристосовані до динамічних середовищ; орієнтовані скоріше на повторне проектування інформаційних систем, що вимагає у випадку постійних змін значних додаткових часових та грошових ресурсів.

Поява таких підходів вимагає створення відповідного інструментарію, що сприяли появі програмно-технологічних засобів спеціального класу – CASE–засобів (англ. Computer-Aided Software Engineering), що реалізують CASE–технології створення і супроводу інформаційних систем. Термін CASE використовується в даний час у досить широкому сенсі. Первісне значення терміну CASE, обмежене питаннями автоматизації розробки тільки програмного забезпечення, сьогодні набуло нового сенсу, що охоплює процес розробки складних інформаційних систем у цілому. Тепер під терміном CASE–засобу розуміють програмні засоби, що підтримують процеси створення і супроводу інформаційних систем, включаючи аналіз і формулювання вимог, проектування прикладного програмного забезпечення і баз даних, генерацію коду, тестування, документування, забезпечення якості, конфігураційне керування і керування проектом, а також інші процеси. CASE–засоби разом із системним програмним забезпеченням і технічними засобами утворюють повне середовище розробки інформаційних систем.

CASE є методологією проектування ІС, а також набором інструментальних засобів, що дозволяють у наочній формі моделювати предметну область,

аналізувати цю модель на всіх етапах розроблення і супроводу ІС і розробляти застосування відповідно до інформаційних потреб користувачів. Більшість існуючих CASE-засобів ґрунтується на методологіях структурного (в основному) або об’єктно-орієнтованого аналізу і проектування, що використовують специфікації у вигляді діаграм або текстів для описання зовнішніх вимог, зв’язків між моделями системи, динаміки поведінки системи й архітектури програмних засобів.

На сьогоднішній день існує велика кількість методологій, що можна використовувати для опису інформаційних систем, відповідно і велика кількість різноманітних діаграм, що входять в ці методології. Проте найдоречнішими із них для опису інформаційних систем, до класу якої входить електронний сервіс підбору гардеробу, є методології IDEF (IDEF0, IDEF3), DFD, ARIS та UML.

2.2 Загальна структура сервісу

Для початку потрібно описати загальну структуру електронного сервісу підбору гардеробу. Структура відображена на рисунку 2.1.

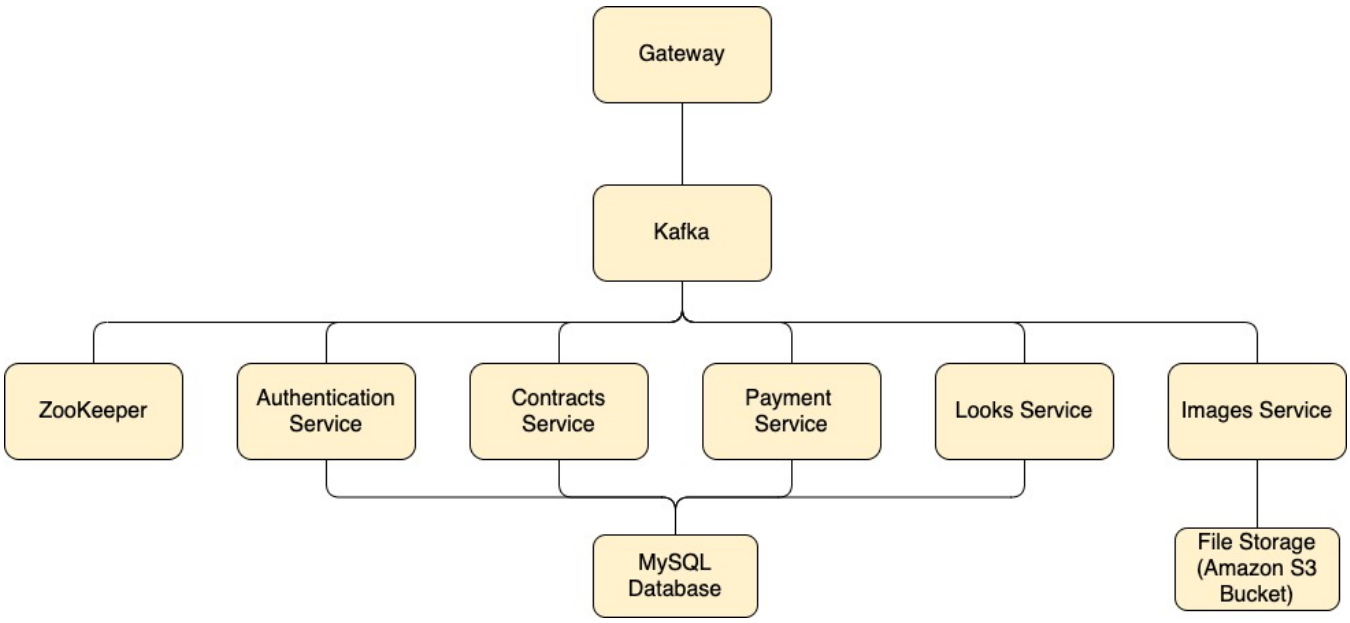


Рисунок 2.1 – Структура електронного сервісу підбору гардеробу

Компоненти електронного сервісу:

а) Gateway (шлюз) – програма, що отримує запити користувачів із глобальної мережі Інтернет та перенаправляє їх до мікросервісів через Kafka у вигляді повідомлень.

б) Kafka – це програма, що передає повідомлення між сервісами всередині системи та водночас є розподіленою чергою для цих повідомлень, що гарантує відсутність невиконаних запитів від користувачів.

в) ZooKeeper – програма-координатор, що використовується Kafka для координації та збереження метаданих розподіленої черги.

г) Authentication Service – мікросервіс, що працює із даними про користувачів в базі даних, а також відповідальна за авторизацію/автентифікацію користувачів у системі.

ґ) Contracts Service – мікросервіс, що використовується для обробки логіки взаємодії стилістів та користувачів через контракти всередині системи.

д) Looks Service – мікросервіс, що використовується для обробки інформації про образи створенні стилістами.

е) Payment Service – мікросервіс, що використовується користувачами для виконання платежів користувачами стилістам за їх роботу.

є) Images Service – мікросервіс, що працює із зображеннями та відповідальний за їх обробку та збереження.

ж) MySQL Database – база даних MySQL для централізованого зберігання даних про користувачів, контракти, образи та оплати.

з) File Storage (Amazon S3 Bucket) – файлове сховище для зберігання зображень, Amazon S3 Bucket – це чудовий приклад сховища для файлів.

2.3 Функціональна модель

Для загального уявлення про діяльність електронного сервісу підбору гардеробу потрібно описати його у вигляді контекстної діаграми, що покаже входи і виходи діяльності сервісу. Входи й виходи контекстної діаграми розподілені по чотирьох сторонах прямокутника. Призначення цих сторін такі:

а) Ліва сторона відповідає входам системи, величинам, які поступають у систему і переробляються нею у вихідні величини.

б) Верхня сторона відповідає входам по керуванню тобто різним керуючим діям, командам, стратегіям поведінки, процедурам, документами, що регламентують виконання роботи тощо.

в) Права сторона відповідає виходам системи, продуктам її діяльності, результатам перетворення вхідних величин.

г) Нижня сторона відповідає механізмам, а саме засобам, ресурсам, за допомогою яких виконуються вказані в прямокутнику функції.

Контекстна діаграма діяльності електронного сервісу підбору гардеробу зображена на рисунку 2.2.

Вхідною інформацією в даному випадку є:

- Реклама – те як підприємство позиціонує себе на ринку.
- Запити користувачів – клієнти сервісу просять про задоволення їх послуги.
- Фінансові ресурси – це можуть бути як і інвестиції так і плата за виконання послуг, контракт.

Вихідною інформацією є:

- Прибуток – фінансові результати діяльності сервісу.
- Організаційні витрати – пов’язані з витратами на команди, що забезпечують функціонування сервісу.
- Витрати на IT-інфраструктуру – витрати пов’язані із середовищем функціонування сервісу.
- Задоволення запитів користувачів – виконання обов’язків сервісу перед клієнтами.

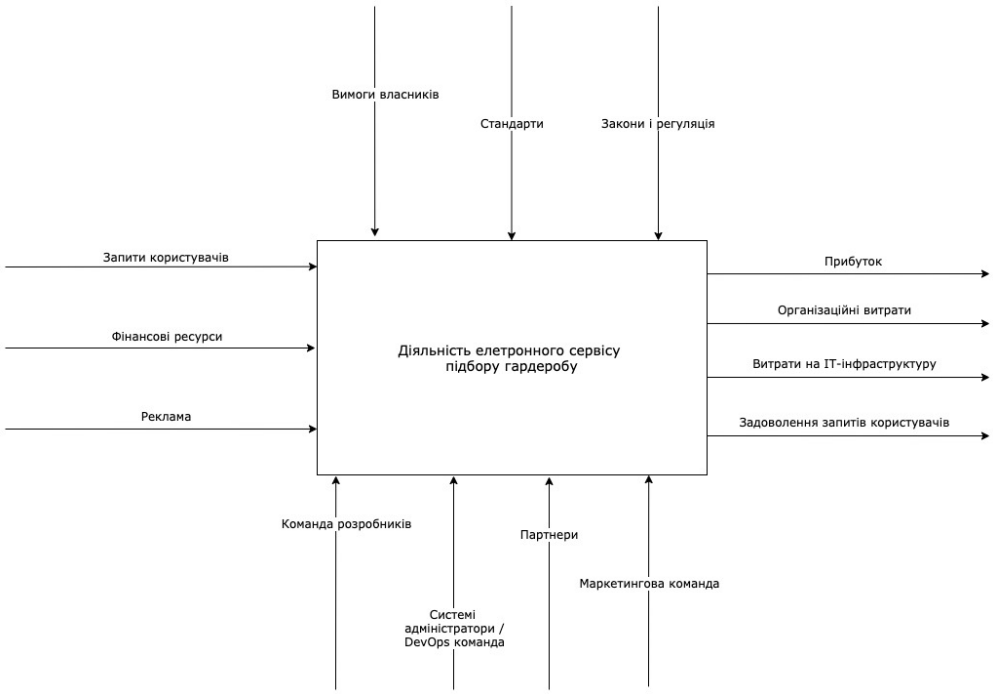


Рисунок 2.2 – Контекстна діаграма діяльності електронного сервісу підбору гардеробу

За управління діяльністю підприємства відповідають:

- вимоги співвласників сервісу;
- стандарти;
- закони та нормативні акти.

За механізм виконання відповідають:

- команда розробників – команда розробників, архітекторів сервісу;
- системні адміністратори / DevOps команда – відповідальні за середовище функціонування сервісу, підтримку його діяльності;
- партнери – стороні для електронного сервісу юридичні або фізичні особи, що мають для даного сервісу особливий статус;
- маркетингова команда – забезпечує розповсюдження інформації про сервіс, продажі та залучення клієнтів.

2.4 Розробка діаграми варіантів використання

Функції електронного сервісу підбору гардеробу, що здатен виконувати користувач, зображені у вигляді UML діаграми варіантів використання на рисунку 2.3.

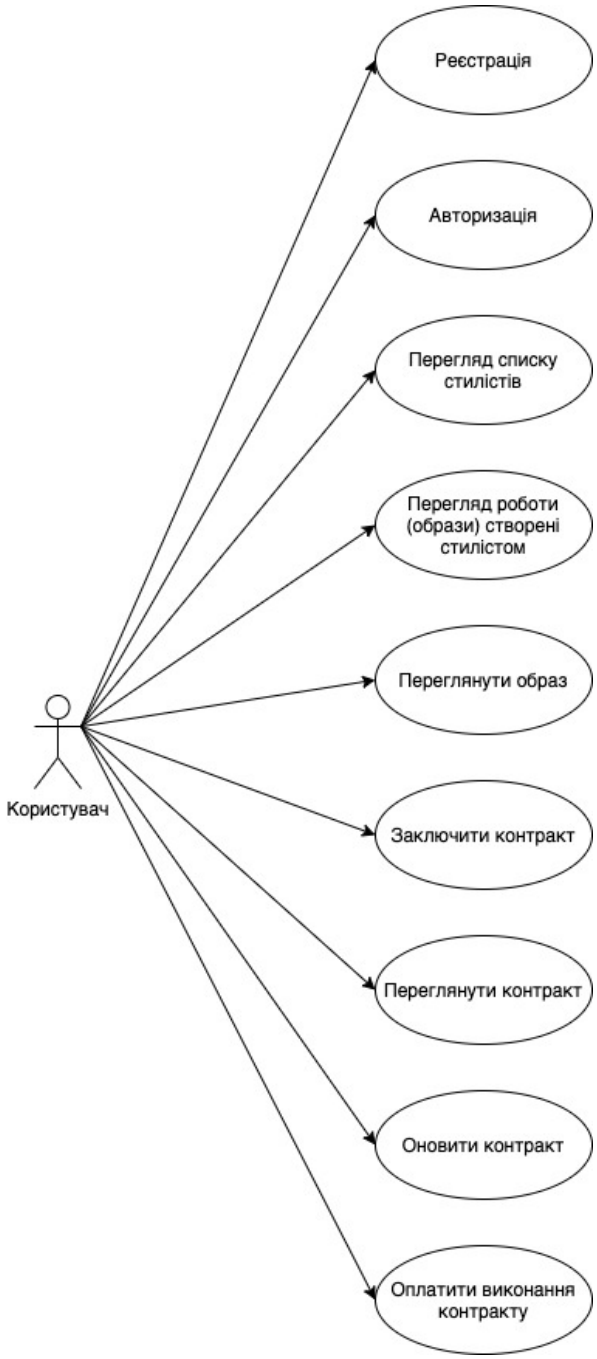


Рисунок 2.3 – Діаграма варіантів використання

2.5 Розробка UML-діаграм послідовності

Вище зображені функції системи потрібно зобразити у вигляді загально відомої нотації і краще за всі інші відомі нотації підійде UML діаграма послідовності, оскільки на ній можна відобразити не лише послідовність дій, а й взаємодію компонентів в процесі виконання.

Процес реєстрації користувача в системі описаний у вигляді UML діаграми послідовності і зображений на рисунку 2.4.

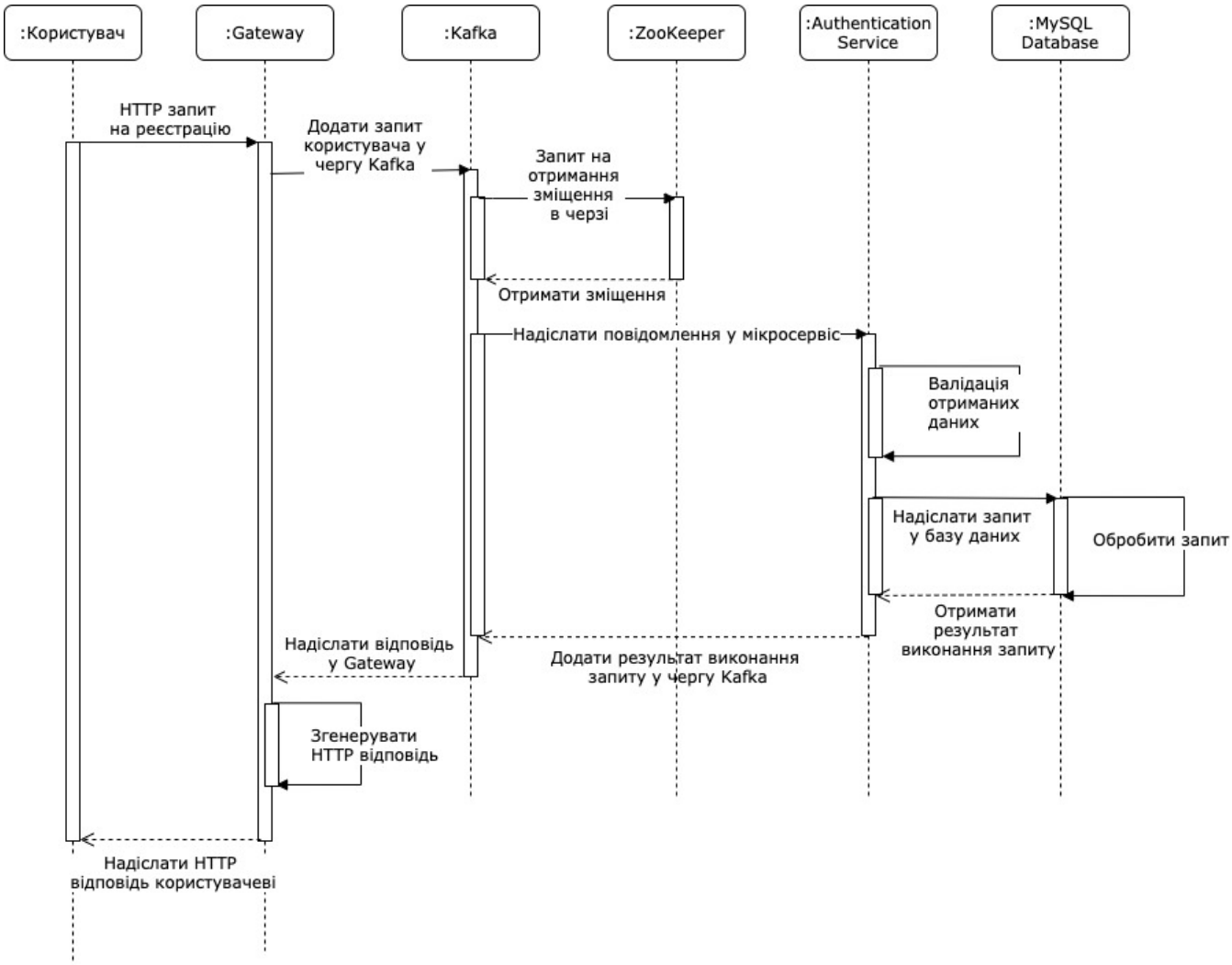


Рисунок 2.4 – Реєстрація користувача

- Користувач надсилає запит у сервіс;
- Gateway мікросервіс отримує HTTP-запит користувача, формує текстове повідомлення та додає це повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає мікросервісу Authentication Service повідомлення для обробки запиту користувача;
- Мікросервіс валідує дані із запиту користувача;
- Authentication Service виконує запит до бази даних для збереження даних;
- База даних обробляє запит та повертає результат обробки запиту у мікросервіс;

- Authentication Service формує текстове повідомлення-відповідь на запит користувача та додає його у розподілену чергу Kafka;
- Kafka передає повідомлення у Gateway;
- Gateway отримує текстове повідомлення, отримує із цього повідомлення дані та формує HTTP-відповідь;
- Gateway надсилає користувачеві HTTP-відповідь.

Процес входу користувача в сервіс використовуючи дані збереженні під час процесу реєстрації зображено на рисунку 2.5.

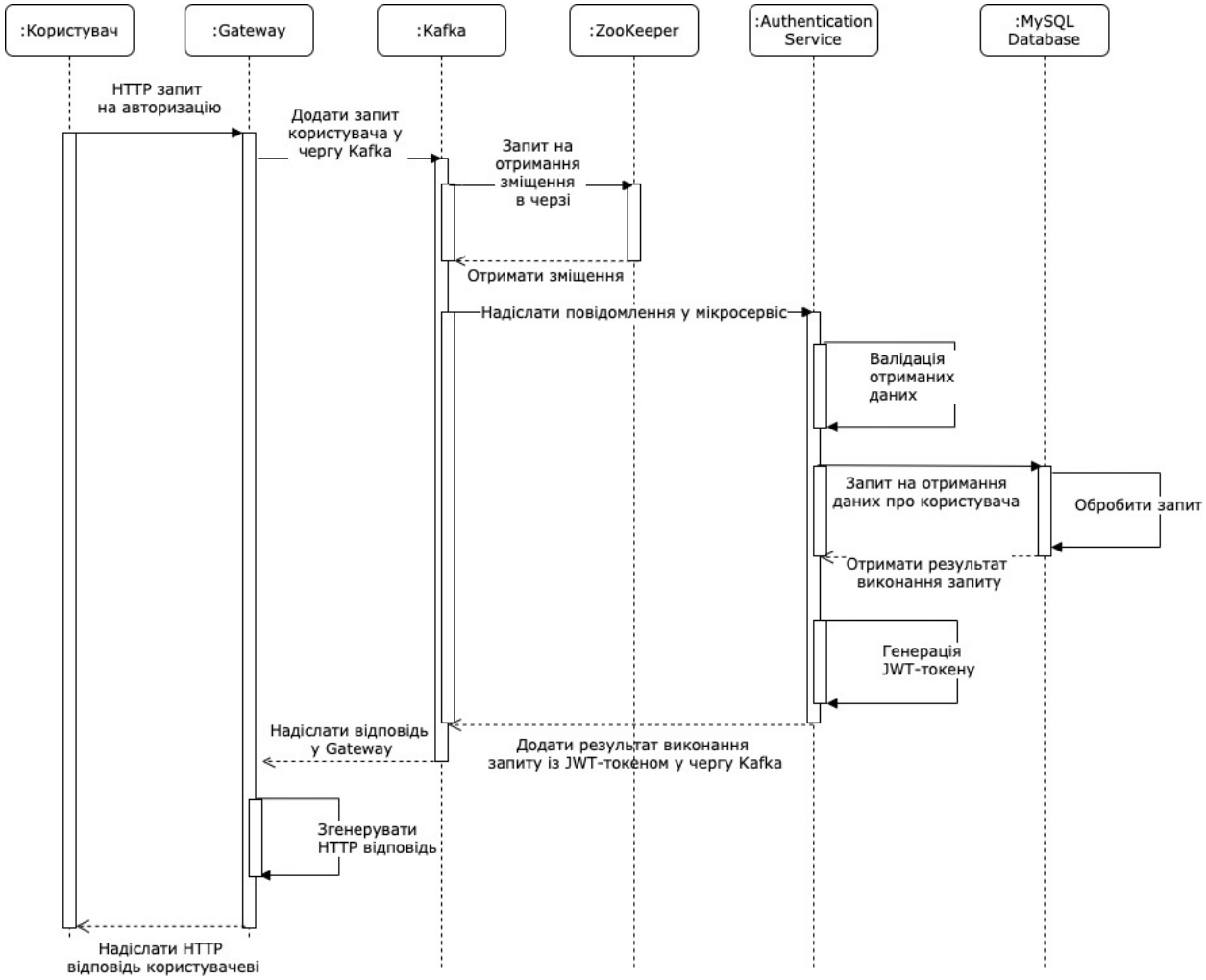


Рисунок 2.5 – Авторизація користувача

- користувач надсилає HTTP-запит у сервіс;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;

- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає повідомлення до Authentication Service для обробки запиту користувача;
- Authentication Service зчитує дані запиту із повідомлення та валідує їх;
- Authentication Service виконує запит у базу даних для зчитування всієї інформації про користувача;
- база даних обробляє запит та повертає результат обробки запиту у мікросервіс;
- Authentication Service отримує всі дані про користувача і генерує із них JWT-токен;
- Authentication Service створює текстове повідомлення із згенерованим JWT-токеном та додає його у розподілену чергу Kafka;
- Kafka надсилає повідомлення у Gateway;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення та формує HTTP-відповідь;
- Gateway надсилає HTTP-відповідь користувачеві.

Для вибору потрібного стиліста користувач повинен познайомити із інформацією про стилістів, а отже йому потрібно отримати список стилістів. Виконання цієї дії вимагає участі декількох мікросервісів. Перший мікросервіс отримує дані про стилістів із бази даних та надсилає їх користувачеві, після того як користувач отримав дані про стилістів потрібно зробити іще запит на отримання аватарів стилістів. Процес зображено на рисунку 2.6.

Процес перегляду образів вибраного стиліста також вимагає участі двох мікросервісів. Перший мікросервіс знаходить і надсилає користувачеві інформацію про образи разом із посиланням на зображення, а інший сервіс надсилає користувачеві зображення цих образів. Процес перегляду образів зображено на рисунку 2.7.

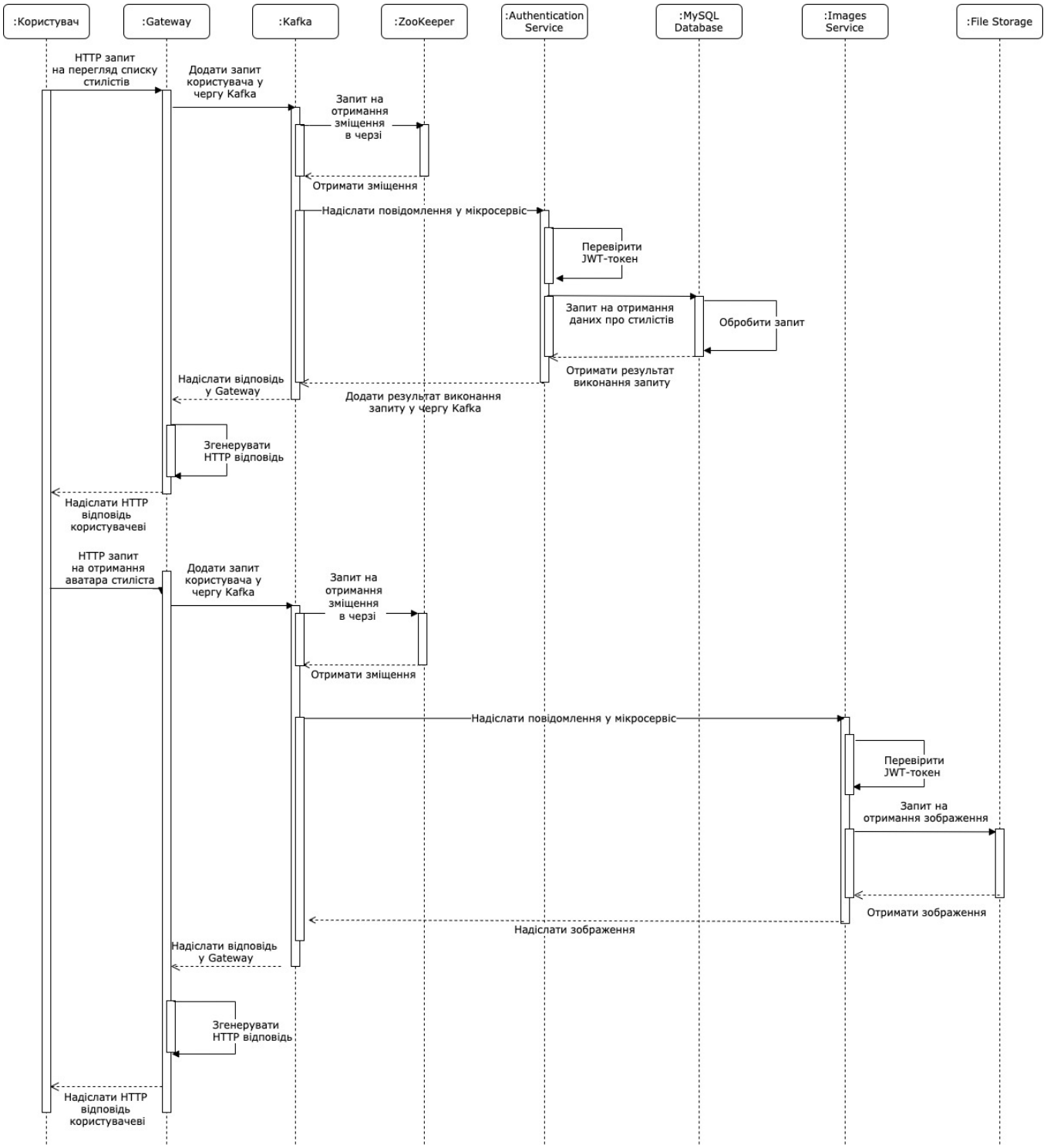


Рисунок 2.6 – перегляд списку стилістів

- користувач надсилає HTTP-запит у сервіс;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;

- Kafka надсилає повідомлення до Authentication Service для обробки запиту користувача;
- Authentication Service зчитує дані запиту із повідомлення та валідує їх;
- Authentication Service виконує запит у базу даних для зчитування інформації про стилістів;
- база даних обробляє запит та повертає результат обробки запиту у мікросервіс;
- Authentication Service отримує інформацію про стилістів та формує текстове повідомлення із цими даними;
- Authentication Service додає повідомлення у розподілену чергу Kafka;
- Kafka надсилає повідомлення у Gateway;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення, формує HTTP-відповідь та надсилає його користувачеві;
- Користувач надсилає HTTP-запит у сервіс для отримання зображень-аватарів стилістів;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає повідомлення до Images Service для обробки запиту користувача;
- Images Service зчитує дані запиту із повідомлення, валідує JWT-токен та дані запиту користувача;
- Images Service виконує запит у файлове сховище для отримання зображень стилістів;
- Images Service отримує зображення та формує текстове повідомлення із ними;
- Images Service додає повідомлення у розподілену чергу Kafka;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення, формує HTTP-відповідь та надсилає відповідь користувачеві.

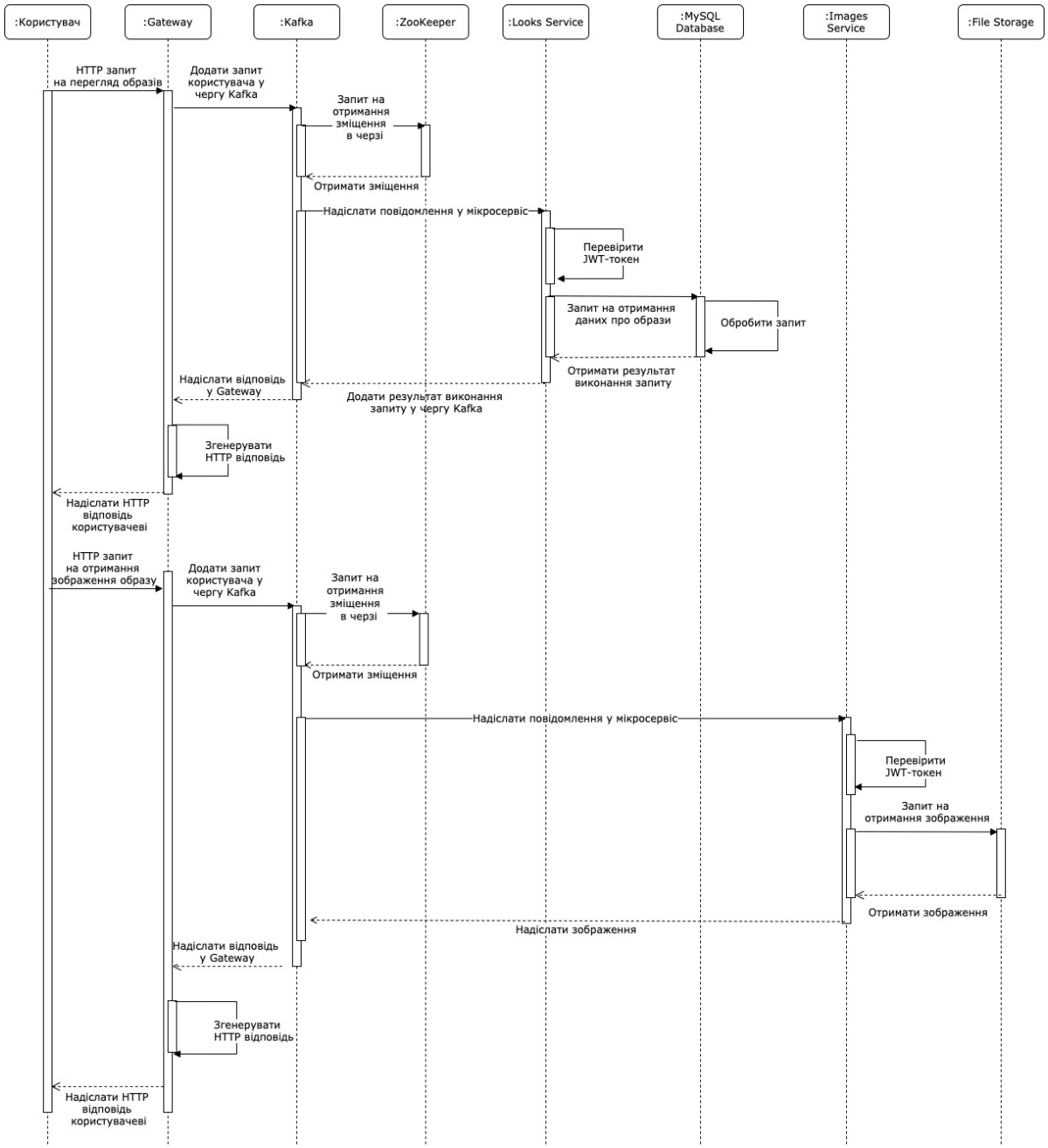


Рисунок 2.7 – перегляд образів стиліста

- користувач надсилає HTTP-запит у сервіс;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає повідомлення до Looks Service для обробки запиту користувача;

- Looks Service зчитує дані запиту із повідомлення, валідує JWT-токен та дані запиту;
- Looks Service виконує запит у базу даних для зчитування інформації про образи;
- база даних обробляє запит та повертає результат обробки запиту у мікросервіс;
- Looks Service отримує інформацію про образи та формує текстове повідомлення;
- Looks Service додає повідомлення у розподілену чергу Kafka;
- Kafka надсилає повідомлення у Gateway;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення, формує HTTP-відповідь та надсилає його користувачеві;
- Користувач надсилає HTTP-запит у сервіс для отримання зображень образів;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає повідомлення до Images Service для обробки запиту користувача;
- Images Service зчитує дані запиту із повідомлення, валідує JWT-токен та дані запиту користувача;
- Images Service виконує запит у файлове сховище для отримання зображень образів;
- Images Service отримує зображення та формує текстове повідомлення із ними;
- Images Service додає повідомлення у розподілену чергу Kafka;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення, формує HTTP-відповідь та надсилає відповідь користувачеві.

Переглянувши роботи декількох стилістів користувач повинен зробити вибір на користь одного з них для укладання контракту між ними. Вибравши стиліста користувач створює контракт всередині системи, коли контракт буде створено стилісту прийде сповіщення про новий контракт. Процес створення контракту всередині системи відображено на рисунку 2.8.

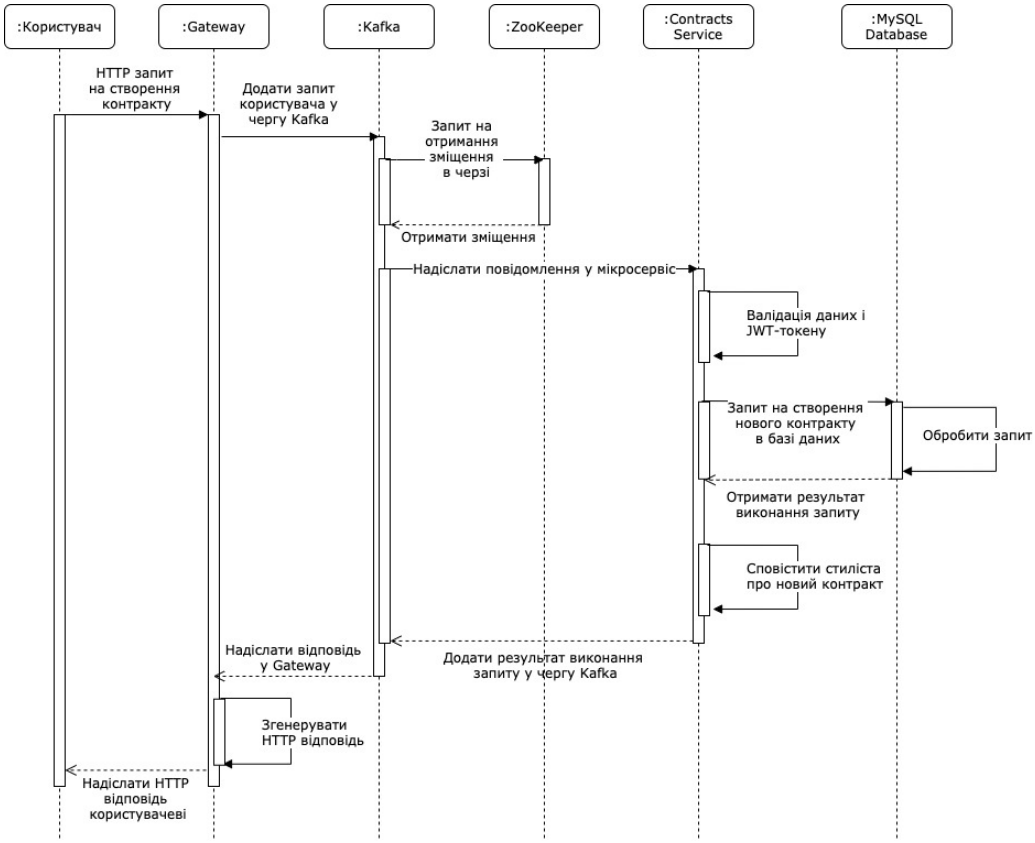


Рисунок 2.8 – Створення контракту

- користувач надсилає HTTP-запит у сервіс;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає повідомлення до Contracts Service для обробки запиту користувача;
- Contracts Service зчитує дані запиту із повідомлення, валідує JWT-токен та дані запиту;

- Contracts Service виконує запит у базу даних для створення нового контракту;
- база даних обробляє запит та повертає результат обробки запиту у мікросервіс;
- Contracts Service сповіщає стиліста про новий контракт;
- Contracts Service формує текстове повідомлення із результатом виконання запиту;
- Contracts Service додає повідомлення у розподілену чергу Kafka;
- Kafka надсилає повідомлення у Gateway;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення, формує HTTP-відповідь та надсилає його користувачеві.

Коли контракт буде виконано користувач має змогу змінити статус контракту на виконаний, іншими словами закрити його. Цей процес зображено на рисунку 2.9.

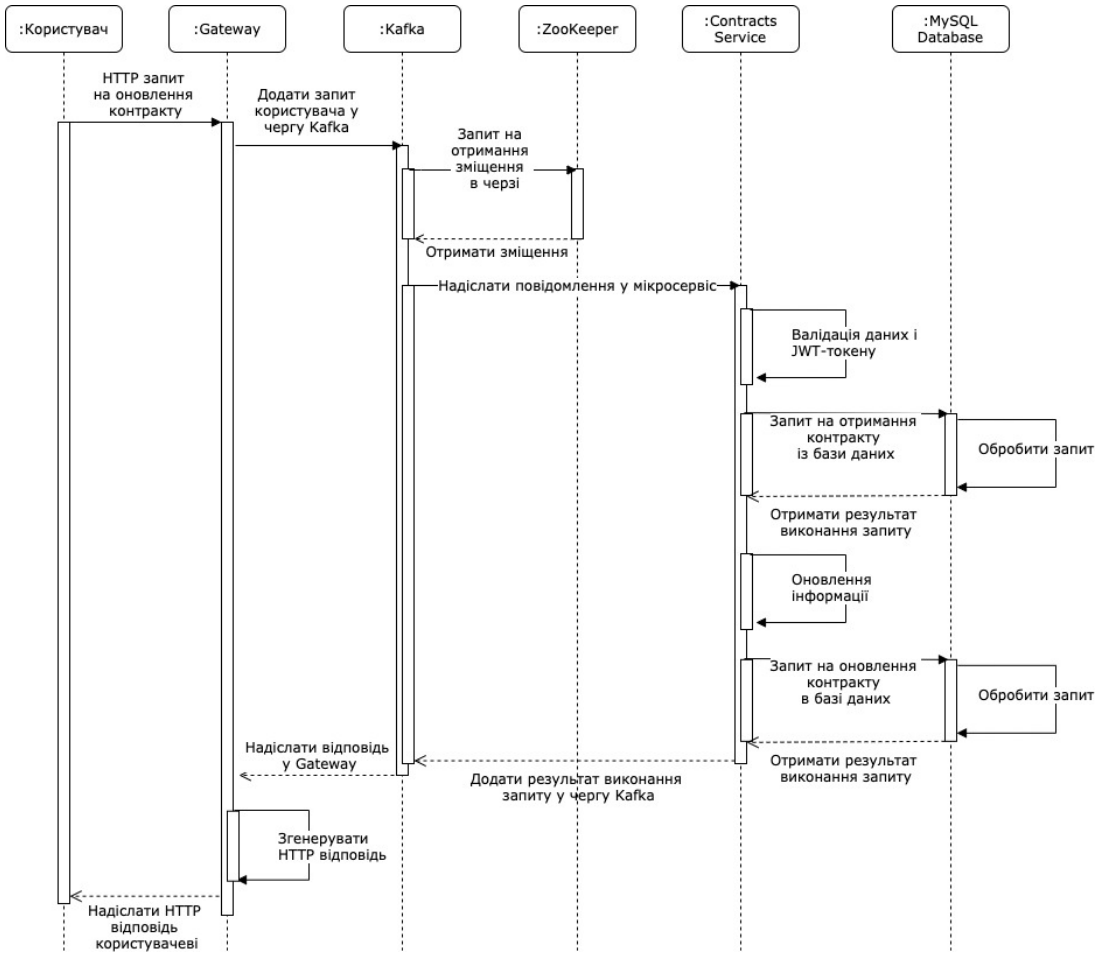


Рисунок 2.9 – Оновлення контракту

- користувач надсилає HTTP-запит у сервіс;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає повідомлення до Contracts Service для обробки запиту користувача;
- Contracts Service зчитує дані запиту із повідомлення, валідує JWT-токен та дані запиту;
- Contracts Service виконує запит у базу даних на отримання даних про контракт;
- база даних обробляє запит та повертає результат обробки запиту у мікросервіс;
- Contracts Service оновлює дані про контракт;
- Contracts Service виконує ще один запит у базу даних, для збереження змін;
- база даних обробляє запит та повертає результат обробки запиту у мікросервіс;
- Contracts Service формує текстове повідомлення із результатом виконання запиту користувача;
- Contracts Service додає повідомлення у розподілену чергу Kafka;
- Kafka надсилає повідомлення у Gateway;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення, формує HTTP-відповідь та надсилає його користувачеві.

Завершальний етап в роботі над образом – оплата послуг стиліста. Користувач повинен заплатити за виконання контракту завчасно обумовлену суму. Процес оплати відображений на рисунку 2.10.

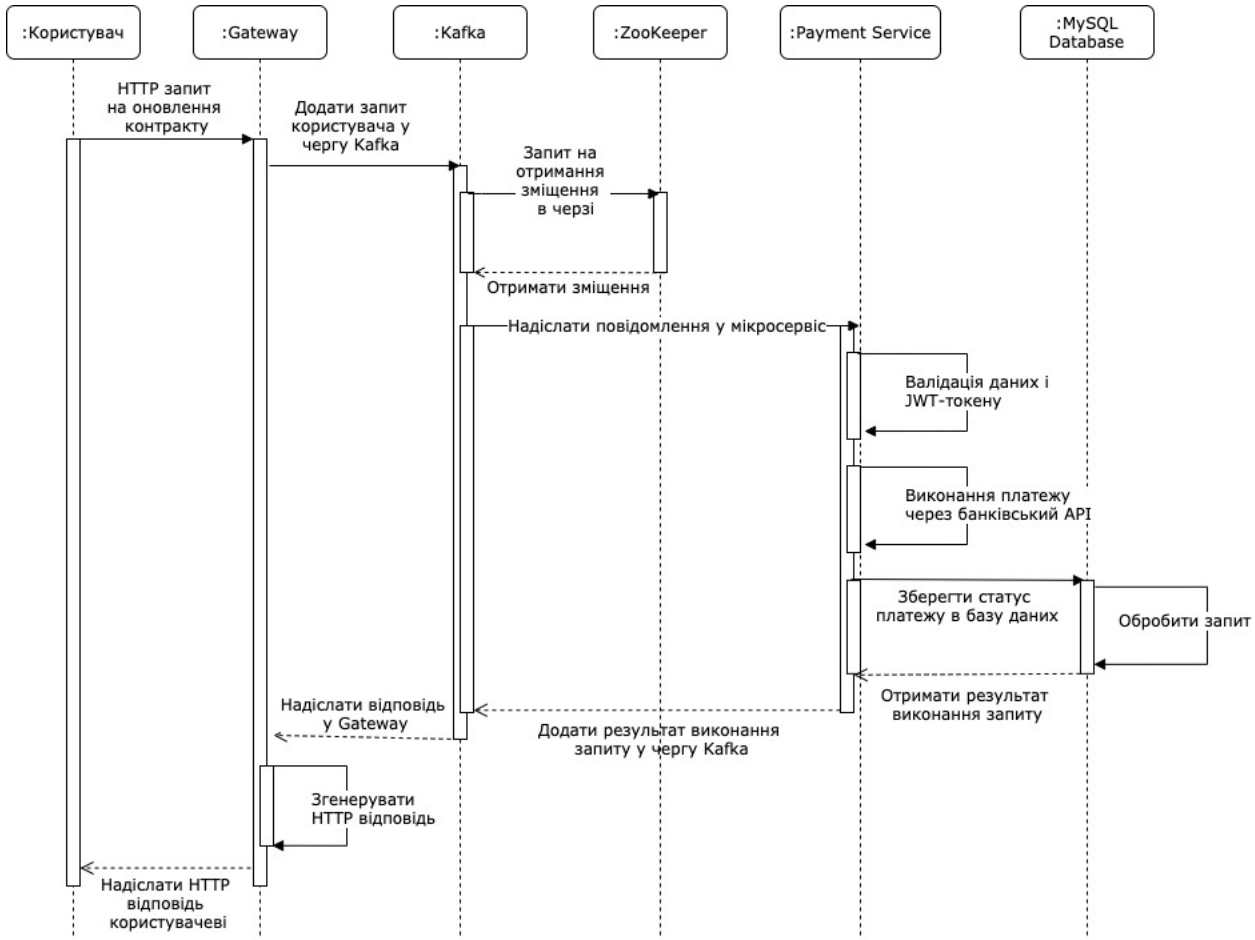


Рисунок 2.10 – Оплата контракту

- користувач надсилає HTTP-запит у сервіс;
- Gateway отримує HTTP-запит користувача, формує текстове повідомлення та додає сформоване повідомлення у розподілену чергу Kafka;
- Kafka надсилає запит на ZooKeeper для отримання зміщення черги та інші метадані черги;
- Kafka надсилає повідомлення у Payment Service для обробки запиту користувача;
- Payment Service зчитує дані запиту із повідомлення, валідує JWT-токен та дані запиту;
- Payment Service виконує запит банківський API для виконання платежу;
- Payment Service виконує запит у базу даних для збереження даних та статусу платежу;
- база даних обробляє запит та повертає результат обробки запиту у мікросервіс;

- Payment Service формує текстове повідомлення із результатом виконання запиту користувача;
- Payment Service додає повідомлення у розподілену чергу Kafka;
- Kafka надсилає повідомлення у Gateway;
- Gateway отримує текстове повідомлення, зчитує дані із повідомлення, формує HTTP-відповідь та надсилає його користувачеві.

3 ІНФОРМАЦІЙНІ, ІНФОКОМУНІКАЦІЙНІ ТА ОРГАНІЗАЦІЙНІ РЕСУРСИ ЕЛЕКТРОННОГО СЕРВІСУ ПІДБОРУ ГАРДЕРОБУ

3.1 Вибір архітектури

Для послідовного вирішення того які ресурси будуть задіяні в діяльності електронного сервісу підбору гардеробу потрібно вирішити її архітектуру як інформаційної системи. Оскільки електронний сервіс потрібно вирішити як цей сервіс спілкуватиметься із клієнтськими додатками, тобто інтерфейс взаємодії. Web-сервіс – це парадигма розробки сервісів, що надають послуги через глобальну мережу Інтернет. Такий підхід до надання послуг через глобальну мережу став доступним в процесі розвитку інформаційних технологій, вдосконалення протоколів передачі даних, створення нових стандартів обміну повідомленнями та розробці нових архітектурних шаблонів інформаційних систем.

Web-сервіси – програмні системи, призначені підтримувати взаємодію між інтероперабельними пристроями через мережу. По суті це реалізації абсолютно чітких інтерфейсів обміну даними між різними додатками. Навколо Web-сервісів виросла ціла екосистема технологій для забезпечення гнучкості у створенні різноманітних систем, від простих CRUD Web-сервісів (сервіси, що забезпечують наступні операції над об’єктами: Create – створити, Read – прочитати, Update – оновити, Delete – видалити), до високонавантажених систем, що здатні обробляти сотні запитів користувачів за секунду. Відповідно і інструментарій для сервісів різної складності відрізняється, якщо наприклад для простого CRUD сервісу достатньо простої бази даних то для високонаватнажених сервісів вже потрібно підключати Load Balancer (розподілювач навантаження), сервіси-координатори, такі як ZooKeeper [7] – сервіси для синхронізації роботи асинхроних додатків, розподілені черги як Apache Kafka [8], бази даних із механізмом реплікації даних і так далі.

Для ефективного використання всіх компонентів системи існують правила оформленні у стандарти розробки інформаційних систем. На сьогоднішній день

найбільш відомими стандартами розробки Web-сервісів є REST, SOAP та різноманітні реалізації RPC [9]. Для більшості випадків достатньо використовувати REST оскільки цей стандарт повністю покриває вимоги CRUD теорії, є простим та прозорим. Для додатків яким потрібно більше функціональних можливостей, тобто виконання функцій, які виходять за рамки CRUD теорії потрібно використовувати SOAP чи реалізацію RPC (наприклад gRPC). Для реалізації електронного сервісу підбору гардеробу достатньо використовувати стандарт REST, оскільки практично всі операції, що будуть виконуватися сервісом пов’язані із даними що зберігаються у базі даних (виконання CRUD операцій) та робота із зображеннями.

Для реалізації сервісу використовується REST стандарт. REST використовує JSON – текстовий формат обміну даними, для опису ресурсів та даних. Для позначення операцій, що потрібно виконати над цими ресурсами використовуються HTTP-методи: POST – для створення ресурсу, GET – для отримання інформації про ресурс, PUT – для оновлення інформації про ресурс, DELETE – для видалення інформації про ресурс.

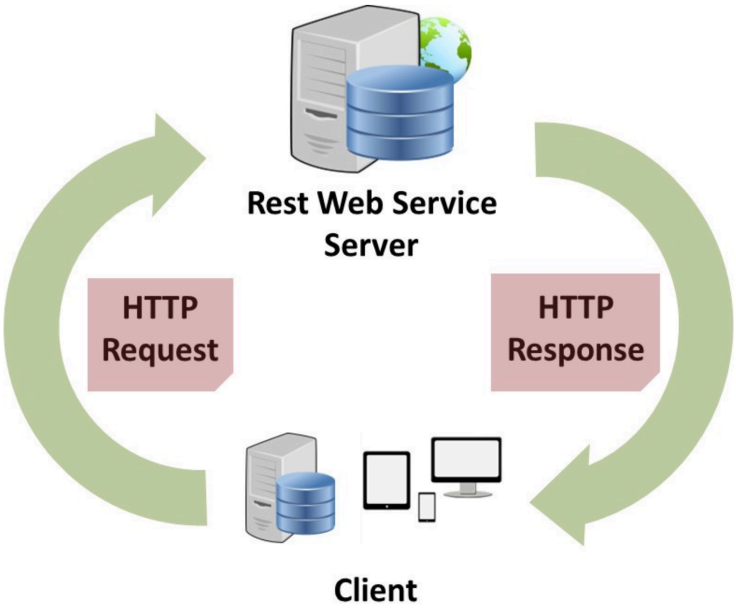


Рисунок 3.1 – REST архітектура [4]

Для реалізації електронного-сервісу підбору гардеробу найкращим рішенням взаємодії між програмними модулями є REST, тому що функціональних можливостей стандарту REST достатньо для виконання поставлених Web-сервісом

завдань, а завдяки його відносній простоті його легше використовувати ніж SOAP чи RPC.

Для організації роботи всередині інформаційних систем існують два фундаментальні підходи: монолітна та мікросервісна архітектури. Кожна архітектура має як свої переваги так і недоліки, у таблиці 3.1 наведено порівняльну характеристику цих архітектур.

Таблиця 3.1 – Порівняльна характеристика монолітної та мікросервісної архітектур

Характеристика	Монолітна архітектура	Мікросервісна архітектура
Швидкість розробки	Монолітні програми легше почати розробляти бізнес-логіку оскільки не потрібно роздумувати про міжпроцесне спілкування.	Мікросервісні додатки розробляються довше тому що кожний мікросервіс реалізовує свою частину бізнес логіки цілої системи і потрібно продумувати взаємодію між цими мікросервісами.
Модульність	Монолітні програми створюються як одне ціле, тому з часом вони виростають до гігантських розмірів які важко супроводжувати, розробляти нові функції та тестувати.	Мікросервіси легше тримати модульними. Це забезпечується жорсткими межами між окремими сервісами.

Продовження таблиці 3.1

Технологічна різноманітність	Монолітні програми повинні розроблятися із використанням того самого стеку технологій із яким починали створювати.	Кожний мікросервіс може розроблятися із використанням різних наборів технології.
Процес розгортання	За рахунок того, що моноліт являє собою повністю функціональну одиницю роботи системи його легше розгорнути. Проте час на компіляцію і часу запуску програми значно більший ніж у мікросервісах.	Для розгортання та управління мікросервісами потрібно вміти використовувати спеціальні інструменти оркестрації, проте час запуску менший ніж у монолітної програми.
Тестування	Функції монолітної програми стає дедалі важче в процесі розробки системи, проте інтеграційні та end-to-end тести проводити легше ніж у мікросервісних системах.	У системах побудованих на мікросервісній архітектурі важче проводити інтеграційне та E2E тестування, оскільки потрібно організувати взаємодію між цими сервісами. Проте функціональні тестування проводити легше ніж у монолітнах, оскільки мікросервісни - це відносно невеликі додатки.

Продовження таблиці 3.1

Обробка помилок	Помилка в роботі монолітної системи може призвести до катастрофічних наслідків, а час на запуск ще одного такого додатку може займати дуже багато часу.	Якщо станеться робота в помилці мікросервісу, це не призведе до повної зупинки роботи всієї системи, лише тієї частини за яку відповідає мікросервіс. А якщо один із мікросервісів перестане працювати зовсім – його майже відразу замінить такий самий мікросервіс.
Ресурси на пікових навантаженнях	Потрібно запускати новий додаток із копією всієї системи.	Кожний мікросервіс відповідальний за свою частину бізнес логіки системи, тому можна запускати лише ті сервіси які найбільше використовуються.

Мікросервісна архітектура краще підходить ніж монолітна тому, що одна частина функцій сервісу буде використовуватися частіше ніж інші функції, мікросервісна архітектура допоможе оптимально розподілювати навантаження із меншими затратами ресурсів та зробити систему гнучкішою і простішою.

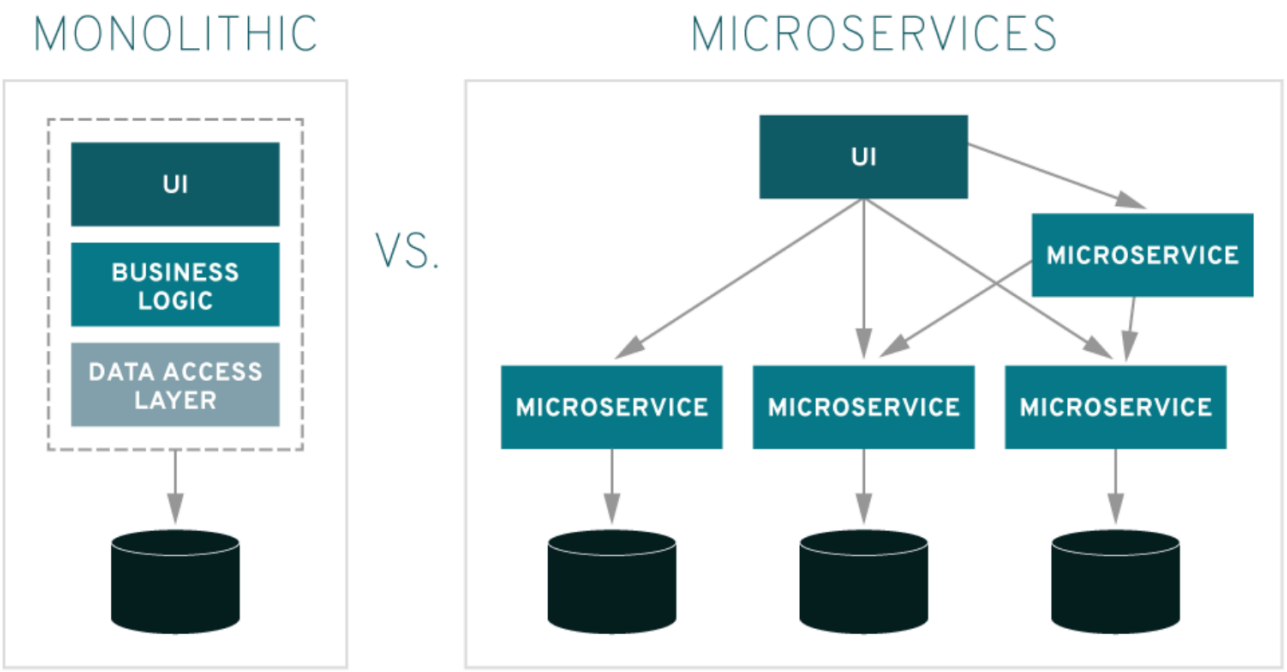


Рисунок 3.2 – Монолітна та мікросервісна архітектури [6]

Щодо того якими характеристиками повинна володіти дана система як Web-сервіс, то серед них можна виділити наступні:

- контрактна взаємодія із сервісом – таким контрактом визначається програмний інтерфейс, комунікаційні вимоги, обмеження, властивості та політика використання сервісу. Програмний модуль, що буде використовувати даний веб-сервіс повинний виконувати ці умови і не очікувати, що веб-сервіс буде виконувати будь які побажання;

- ізоляція внутрішньої логіки від зовнішнього світу – вся інформація про сервіси повинна бути прихована, окрім контрактної інформації – інтерфейсу взаємодії із сервісом. Перевага такого підходу заключається в мінімізації впливу на клієнтські програми, що використовують сервіс, у випадку яких небуть дій по оновленню сервісів;

- сервіс не повинен мати властного стану – сервіс повинен володіти властивістю. Якщо сервіс в тій чи іншій формі зберігає інформацію про свій стан, то виявляється прив'язаним до виконуваного ним завдання. Для переходу до наступного завдання йому потрібно звільнитися від накопиченого стану, а це вимагає додаткових ресурсів і взагалі не завжди можливо. Відсутність властного

стану забезпечує нейтральність по відношенню до компонентів, що звертаються до сервісу.

- мікросервіси повині бути самоврядними – для того щоб мікросервіси могли існувати незалежно один від одного и від навколишнього середовища, вони повині бути автономними тобто володіти властивістю самоуправління;
- мікросервіси повині використовуватися багаторазово – один і той самий мікросервіс може використовуватися компонентами, що його викликають, тобто може використовуватися багаторазово, як будь яка функція чи підпрограм.

Результуюча архітектура сервісу зображена на рисунку 3.3.

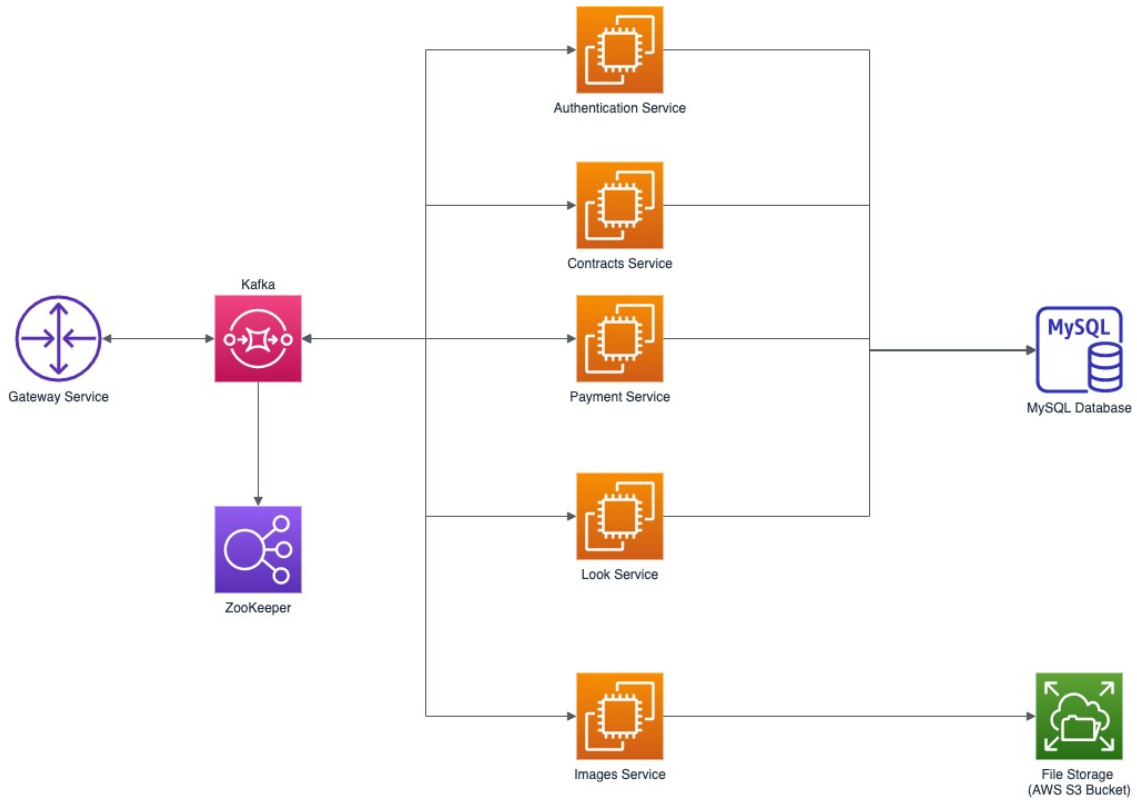


Рисунок 3.3 – Архітектура електронного сервісу підбору гардеробу

Ролі компонентів:

а) Gateway Service – це компонент системи, що відповідальний за спілкування із зовнішнім, по відношенню до системи, середовищем. Усі запити користувачів проходять через цей сервіс.

б) Kafka. Цей компонент використовується для спілкування всередині системи. Запити від Gateway до мікросервісів проходять у вигляді повідомлень, що

накопичуються у вигляді черги в Kafka. Підхід використання черги у мікросервісі дозволяє оминути проблеми із відмовою обробки запитів користувачів у часи пікового навантаження на систему, оскільки запити потрапляють у чергу.

в) ZooKeeper. Цей компонент використовується Kafka для збереження метаданих черг, в тому числі зміщення в черзі.

г) Authentication Service – це мікросервіс, що використовується для роботи із даними користувачів в тому числі і авторизація користувачів.

ґ) Contracts Service – це мікросервіс відповідальний за виконання логіки взаємодії клієнтів та стилістів через внутрішні контракти.

д) Payment Service – це мікросервіс відповідальний за платіжні функції сервісу.

е) Look Service – це мікросервіс відповідальний за роботу із образами користувачів, що створюються стилістами.

є) Images Service – це мікросервіс, що відповідає за роботу із зображеннями в сервісі.

ж) MySQL Database – база даних MySQL, використовуються для збереження усієї текстової інформації сервісу і даних користувачів.

з) File Storage – компонент сервісу, що використовується для збереження графічних даних сервісу – зображень. У випадку електронного сервісу підбору гардеробу це AWS S3 Bucket – файлове сховище компанії Amazon.

Web-сервіс – це система, що надає послуги користувачам через глобальну мережу Інтернет і як правило працює із даними тому для їх накопичення та подальшої роботи із ними потрібне місце для їх централізованої обробки. Для таких цілей використовують бази даних.

3.2 Розробка бази даних

Питання вибору бази даних залежить від багатьох факторів, оскільки на сьогоднішній день існує величезна кількість класів баз даних та їх реалізацій. Перелічимо основні критерії класифікації баз даних:

а) По моделі даних:

1) Реляційна.
2) Мережева.
3) Ієрархічна.
4) Об’єктно-реляційна.

б) За середовищем зберігання:

1) В оперативній пам’яті.
2) В постійній (вторинній пам’яті).
3) В третинній пам’яті.

в) По степені розподіленості:

1) Централізована.
2) Розподілена.
3) Децентралізована.
4) Фрагментована.

Для зручності користування базами даних використовують СКБД (система керування базами даних) які мають свої реалізації доступу до даних в базах даних, свою мову опису даних (DDL – англ. Data Description Language), мову маніпулювання даними (DML – англ. Data Manipulation Language) та мову запитів (англ. Query Language).

На основі цієї класифікації можна вибрати СКБД, що підходить для даного електронного сервісу підбору гардеробу – це MySQL. MySQL – це СУБД із відкритим вихідним кодом, що тим не менш виконує свої функції на рівні найкращих комерційних СКБД як Oracle чи Amazon Redshift. MySQL – реляційна, цетралізована база даних із механізмом реплікації даних Причина вибору MySQL полягає в тому, що сама СКБД надає всі необхідні можливості для роботи даного проекту, не потребує грошових витрат на ліцензію за комерційне використання та є простою в освоюванні.

Оскільки робота електронного сервісу підбору гардеробу будується навколо даних – потрібно правильно організувати базу даних. База даних електронного сервісу підбору гардеробу представлена на рисунку 3.4.

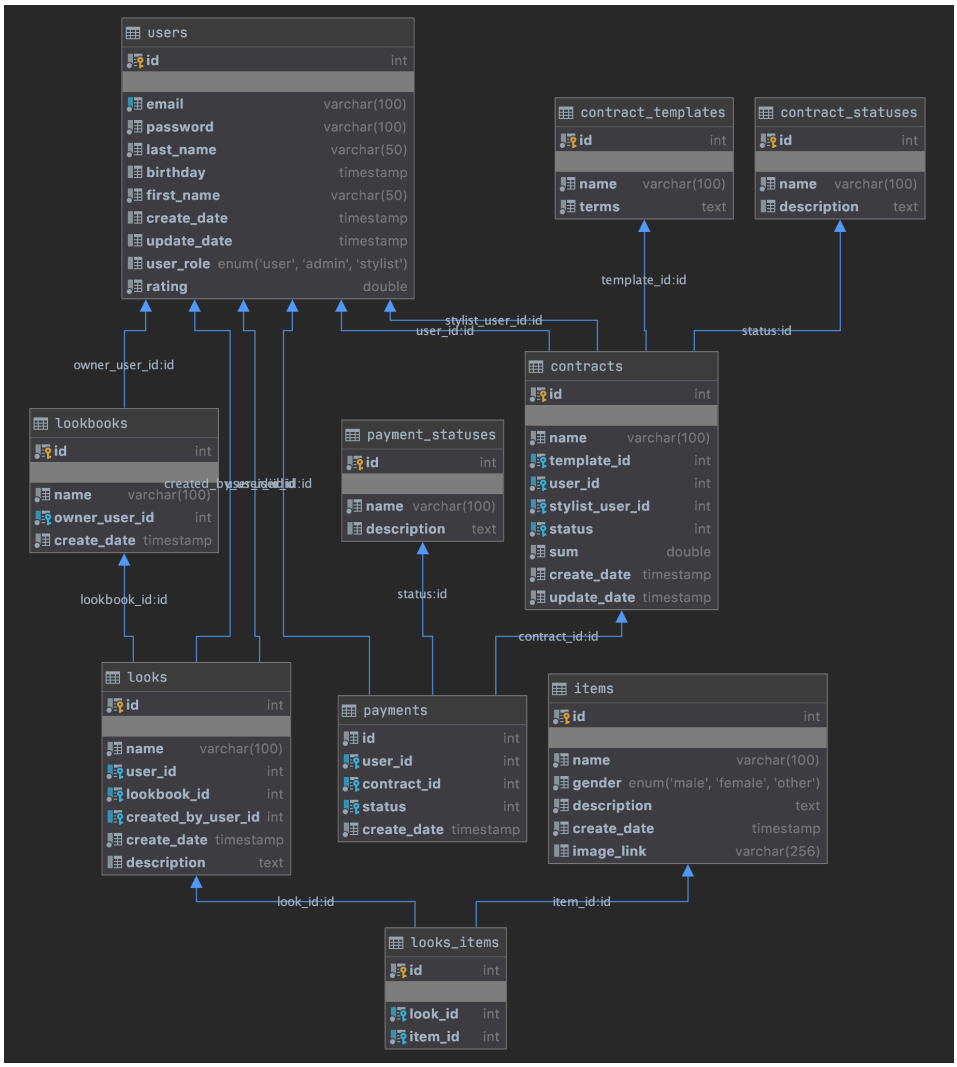


Рисунок 3.4 – Схема бази даних електронного сервісу підбору гардеробу

Таблиця “users” (таблиця із інформацією про користувачів):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “email” – унікальний, зрозумілий людині, ідентифікатор користувача;
- колонка “password” – пароль користувача для авторизації в сервісі;
- колонка “birthday” – вказує на дату народження користувача;
- колонка “first_name” – ім’я користувача;
- колонка “last_name” – прізвище користувача;
- колонка “create_date” – дата і час коли запис було створено в базі даних;
- колонка “update_date” – дата і час коли запис було оновлено в базі даних;
- колонка “user_role” – вказує тип аккаунта користувача, його роль в сервісі;

- колонка “rating” – відображає рейтинг користувача системи (або стиліста якщо це аккаунт стиліста).

Таблиця “lookbooks” (таблиця із інформацією про альбоми образів стилістів):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “name” – назва альбому;
- колонка “owner_user_id” – числовий ідентифікатор користувача, що є власником даного альбому образів;
- колонка “create_date” – дата і час коли запис було створено запис в базі даних.

Таблиця “looks” (таблиця із інформацією про образи):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “name” – назва образу;
- колонка “user_id” – числовий ідентифікатор користувача для якого було створено даний образ;
- колонка “lookbook_id” – числовий ідентифікатор альбому образів;
- колонка “created_by_user_id” – числовий ідентифікатор користувача-стиліста, що створив даний образ.
- колонка “create_date” – дата і час коли образ було створено.

Таблиця “items” (речі до створеного образу):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “name” – назва речі;
- колонка “gender” – стать для якої підходить дана річ;
- колонка “description” – опис речі;
- колонка “image_link” – посилання на зображення речі;
- колонка “create_date” – дата і час коли запис було зроблено в базу даних.

Таблиця “looks_items” (зв’язуюча таблиця для зв’язку багато-до-багатьох):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “look_id” – унікальний числовий ідентифікатор образу;
- колонка “item_id” – унікальний числовий ідентифікатор речі для образу.

Таблиця “contract_statuses” (статуси контракту):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “name ” – назва статусу контракту;
- колонка “description” – опис статусу контракту, його трактування.

Таблиця “contract_template” (шаблони контрактів):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “name ” – назва шаблону контракту;
- колонка “terms” – вихідні умови контракту.

Таблиця “contracts” (таблиця контрактів):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “name” – назва контракту;
- колонка “template_id” – унікальний ідентифікатор шаблону, на основі якого створюється контракт;
- колонка “user_id” – унікальний числовий ідентифікатор клієнта (звичайного користувача);
- колонка “stylist_user_id” – унікальний числовий ідентифікатор користувача-стиліста;
- колонка “status” – унікальний числовий ідентифікатор статусу в таблиці “contract_statuses”;
- колонка “sum” – сума оплати стилістові за виконання контракту;
- колонка “create_date” – дата і час коли контракт було створено;
- колонка “update_date” – дата і час коли контракт було оновлено.

Таблиця “payment_statuses” (статуси оплати):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “name ” – назва статусу;
- колонка “description” – опис статусу.

Таблиця “payments” (таблиця проведених оплат):

- колонка “id” – унікальний числовий ідентифікатор запису в таблиці;
- колонка “user_id ” – унікальний ідентифікатор користувача, що виконував оплату;
- колонка “contract_id” – унікальний числовий ідентифікатор контракту;

- колонка “status” – унікальний числовий ідентифікатор статусу в таблиці “payment_statuses”;
- колонка “create_date” – дата і час коли було проведено оплату.

Код створення бази даних із таблицями написаний у додатку А.

ВИСНОВКИ

Під час виконання кваліфікаційної було виконано всі поставлені задачі:

1. Проведено аналіз аналогічних рішень для підбору гардеробу, визначено, що більшість існуючих електронних сервісів надають послуги використовуючи бізнес модель підписки, що має певні недоліки, а саме непрозору роботу із стилістом, постійні фінансові витрати та відсутність гнучкості.

2. Проведено моделювання предметної галузі із використанням нотації IDEF0 для відображення функцій системи та побудовано UML-діаграми для відображення процесів взаємодії користувача із сервісом.

3. Проаналізовано існуючі архітектури електронних сервісів та розроблено архітектуру електронного сервісу підбору гардеробу, що ґрунтується на мікросервісному шаблоні інформаційної системи та REST-архітектурі для реалізації комунікації між сервісом та клієнтом.

4. Реалізовано базу даних для сервісу із використанням СКБД MySQL, що дозволяє працювати із базою даних комерційного рівня без фінансових витрат на ліцензію.

5. В якості удосконалення запропоновано нову модель взаємодії користувачів із стилістами. В порівнянні із існуючими рішеннями, електронний сервіс підбору гардеробу націлений на поодиночну взаємодію із стилістами для створення образів та підбору гардеробу, а це означає, що клієнт використовуватиме сервіс тоді коли йому необхідно із меншими фінансовими затратами, адже не потрібно щомісяця поновлювати підписку на сервіс. Дана модель – модель контрактної взаємодії – також чудово підійде у випадку якщо потрібно в короткий період часу створити декілька образів, адже клієнт може створити контракт одразу декільком стилістам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Недашківський О.Л. Планування та проектування інформаційних систем / О.Л. Недашківський. – К.: ДУТ, 2015. – 218с.

2. Як виміряти ефективність вашого веб-сервісу? [Електронний ресурс]. - Режим доступу: https://www.a1qa.ru/blog/kak-izmerit-effektivnost-vashego-veb-servisa/

3. П’ять простих кроків для розуміння JSON Web Token (JWT) [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/340146/

4. RFC (Request For Comments) 7519 [Електронний ресурс]. – Режим доступу: https://tools.ietf.org/html/rfc7519

5. Оволодіння REST архітектурою – Деталі REST Архітектури [Електронний ресурс]. – Режим доступу: https://medium.com/@ahmetozlu93/mastering-rest-architecture-rest-architecture-details-e47ec659f6bc

6. Що таке мікросервіси? [Електронний ресурс]. – Режим доступу: https://www.redhat.com/en/topics/microservices/what-are-microservices

7. Документація Apache ZooKeeper [Електронний ресурс]. – Режим доступу: https://zookeeper.apache.org

8. Документація Apache Kafka [Електронний ресурс]. – Режим доступу: https://kafka.apache.org

9. Знайте свої протоколи API [Електронний ресурс]. – Режим доступу: https://www.mertech.com/blog/know-your-api-protocols

10. StitchFix [Електронний ресурс] : [Інтернет портал]. – Режим доступу: https://www.stitchfix.com/

11.Modabox [Електронний ресурс] : [Інтернет портал]. – Режим доступу: https://www.moda-box.com/

ДОДАТКИ

Додаток А – код створення бази даних і таблиць для електронного сервісу підбору гардеробу.

```
CREATE DATABASE wardrobe_service_db;
USE wardrobe_service_db;

-- 'users' table
CREATE TABLE users
(
  id INT AUTO_INCREMENT,
  email VARCHAR(100) NOT NULL,
  password VARCHAR(100) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  birthday TIMESTAMP NULL,
  first_name VARCHAR(50) NOT NULL,
  create_date TIMESTAMP NULL,
  update_date TIMESTAMP NULL,
  user_role ENUM('USER', 'ADMIN', 'STYLIST') NULL,
  rating DOUBLE DEFAULT 0 NOT NULL,
  CONSTRAINT users_pk
    PRIMARY KEY (id)
);

CREATE UNIQUE INDEX users_email_uindex ON users (email);

-- 'items' table
CREATE TABLE items
(
  id INT AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  gender ENUM('MALE', 'FEMALE', 'OTHER') NOT NULL,
  description TEXT NOT NULL,
  image_link VARCHAR(256) NULL,
  create_date TIMESTAMP NOT NULL,
  CONSTRAINT items_pk
    PRIMARY KEY (id)
);

-- 'lookbooks' table
CREATE TABLE lookbooks
(
  id INT AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  owner_user_id INT NOT NULL,
  create_date TIMESTAMP NOT NULL,
  CONSTRAINT lookbooks_pk
```

```
    PRIMARY KEY (id),
  CONSTRAINT lookbooks_users_id_fk
    FOREIGN KEY (owner_user_id) REFERENCES users (id)
);

-- 'looks' table
CREATE TABLE looks
(
  id INT AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  description TEXT NULL,
  user_id INT NOT NULL,
  lookbook_id INT NOT NULL,
  created_by_user_id INT NULL,
  create_date TIMESTAMP NOT NULL,
  CONSTRAINT looks_pk
    PRIMARY KEY (id),
  CONSTRAINT looks_lookbooks_id_fk
    FOREIGN KEY (lookbook_id) REFERENCES lookbooks (id),
  CONSTRAINT looks_users_id_fk
    FOREIGN KEY (user_id) REFERENCES users (id),
  CONSTRAINT looks_users_id_fk_2
    FOREIGN KEY (created_by_user_id) REFERENCES users (id)
);

-- 'looks_items' table
CREATE TABLE looks_items
(
  id INT AUTO_INCREMENT,
  look_id INT NOT NULL,
  item_id INT NOT NULL,
  CONSTRAINT looks_items_pk
    PRIMARY KEY (id),
  CONSTRAINT looks_items_items_id_fk
    FOREIGN KEY (item_id) REFERENCES items (id),
  CONSTRAINT looks_items_looks_id_fk
    FOREIGN KEY (look_id) REFERENCES looks (id)
);

-- 'contract_templates' table
CREATE TABLE contract_templates
(
  id INT AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  terms TEXT NOT NULL,
  CONSTRAINT contract_templates_pk
    PRIMARY KEY (id)
);
```

```
-- 'contract_statuses' table
CREATE TABLE contract_statuses
(
  id INT AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  description TEXT NULL,
  CONSTRAINT contract_statuses_pk
    PRIMARY KEY (id)
);

-- 'contracts' table
CREATE TABLE contracts
(
  id INT AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  template_id INT NOT NULL,
  user_id INT NOT NULL,
  stylist_user_id INT NOT NULL,
  status INT NOT NULL,
  sum DOUBLE NOT NULL,
  create_date TIMESTAMP NOT NULL,
  update_date TIMESTAMP NOT NULL,
  CONSTRAINT contracts_pk
    PRIMARY KEY (id),
  CONSTRAINT contracts_contract_statuses_id_fk
    FOREIGN KEY (status) REFERENCES contract_statuses (id),
  CONSTRAINT contracts_contract_templates_id_fk
    FOREIGN KEY (template_id) REFERENCES contract_templates (id),
  CONSTRAINT contracts_users_id_fk
    FOREIGN KEY (user_id) REFERENCES users (id),
  CONSTRAINT contracts_users_id_fk_2
    FOREIGN KEY (stylist_user_id) REFERENCES users (id)
);

-- 'payment_statuses' table
CREATE TABLE payment_statuses
(
  id INT AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  description TEXT NULL,
  CONSTRAINT payment_statuses_pk
    PRIMARY KEY (id)
);

-- 'payments' table
CREATE TABLE payments
(
  id INT NOT NULL,
  user_id INT NOT NULL,
```

```
  contract_id INT NOT NULL,
  status INT NOT NULL,
  create_date TIMESTAMP NOT NULL,
  CONSTRAINT payments_contracts_id_fk
    FOREIGN KEY (contract_id) REFERENCES contracts (id),
  CONSTRAINT payments_payment_statuses_id_fk
    FOREIGN KEY (status) REFERENCES payment_statuses (id),
  CONSTRAINT payments_users_id_fk
    FOREIGN KEY (user_id) REFERENCES users (id)
);
```

Додаток Б – презентація

Слайд 1:

Міністерство освіти і науки України
Державний університет телекомунікацій
Навчально-науковий інститут інформаційних технологій
Кафедра системного аналізу

БАКАЛАВРСЬКА РОБОТА
за темою «Системний аналіз електронного сервісу підбору гардеробу»

Виконав: студент групи САД-41
Когут Дмитро Русланович
Науковий керівник:
к.т.н., зав. кафедри Системного аналізу, Золотухіна О.А.

Київ-2020

Слайд 2:

Об’єкт, предмет, мета бакалаврської роботи

- **Об’єкт дослідження** – процес підбору гардеробу із використанням електронних сервісів.
- **Предмет дослідження** – моделі та технології електронного сервісу підбору гардеробу.
- **Мета роботи** – розробка пропозицій щодо вдосконалення електронного сервісу підбору гардеробу за рахунок впровадження нової моделі взаємодії клієнта та стиліста.

1

Слайд 3:

Актуальність бакалаврської роботи

- Мала поширеність електронних сервісів підбору гардеробу в Україні.
- Існуюча модель взаємодії реалізована у вигляді підписки, що підходить не усім користувачам.
- Клієнти взаємодіють із компаніями, а не із стилістами.

2

Слайд 4:

Завдання бакалаврської роботи

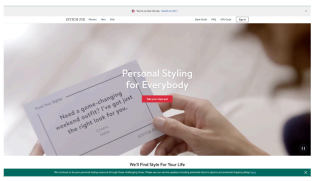
- Провести аналітичний огляд аналогів та їх бізнес-процесів підбору гардеробу на основі інформації із загальнодоступних джерел, в т.ч. із мережі Інтернет.
- Визначити недоліки існуючих рішень та проблемні питання.
- Розробити нову бізнес-модель роботи із стилістом.
- Описати нову модель бізнес-процесу з підбору гардеробу використовуючи графічні нотації.
- Розробити архітектуру електронного сервісу підбору гардеробу.
- Реалізувати базу даних для електронного сервісу підбору гардеробу.

3

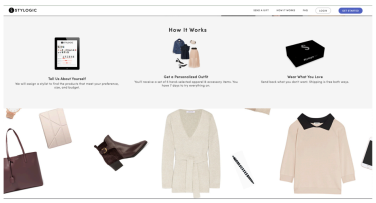
Слайд 5:

АНАЛОГИ СЕРВІСУ ДЛЯ ПІДБОРУ ГАРДЕРОБУ:

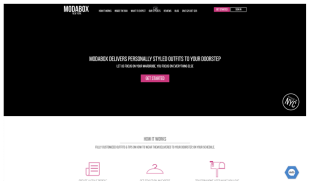
Stitch Fix



Stylogic



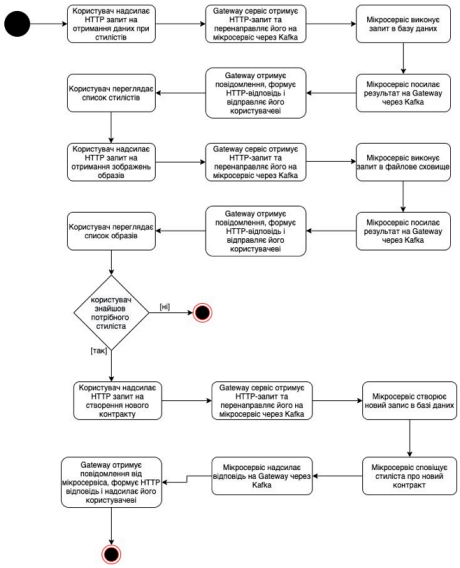
Modabox



4

Слайд 6:

КОНТРАКТНА МОДЕЛЬ ВЗАЄМОДІЇ



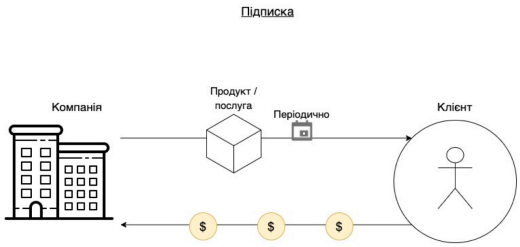
Статуси:

- Ініційований контракт
- Відхилений контракт
- Контракт в процесі виконання
- Невизначений стан
- Успішний контракт
- Неуспішний контракт
- Конфліктний контракт

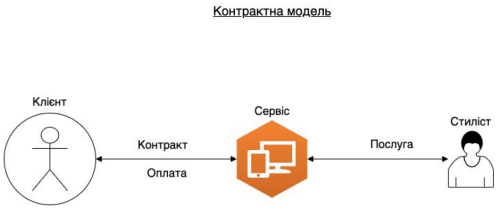
5

Слайд 7:

МОДЕЛЬ ПІДПИСКИ І КОНТРАКТНА МОДЕЛЬ



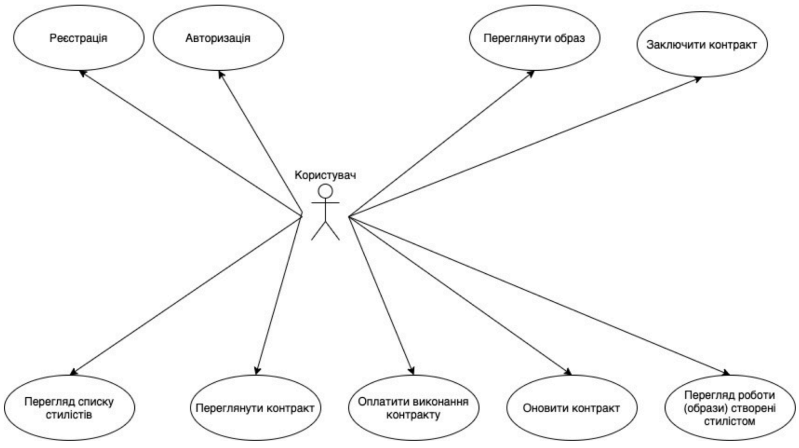
VS



6

Слайд 8:

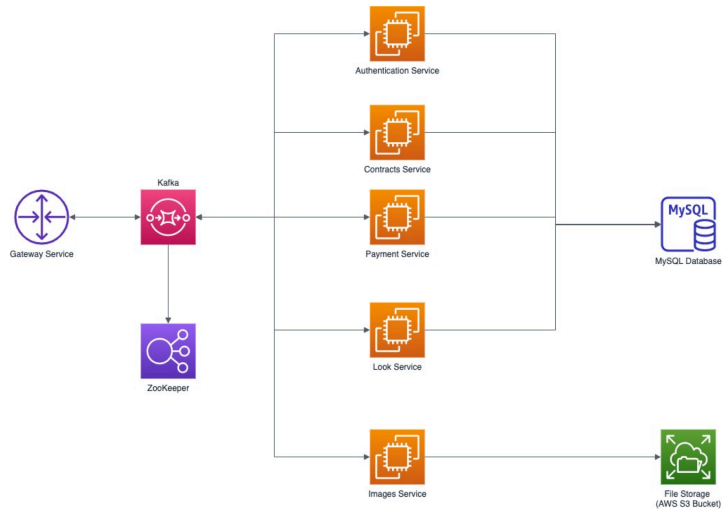
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



7

Слайд 9:

АРХІТЕКТУРА СЕРВІСУ

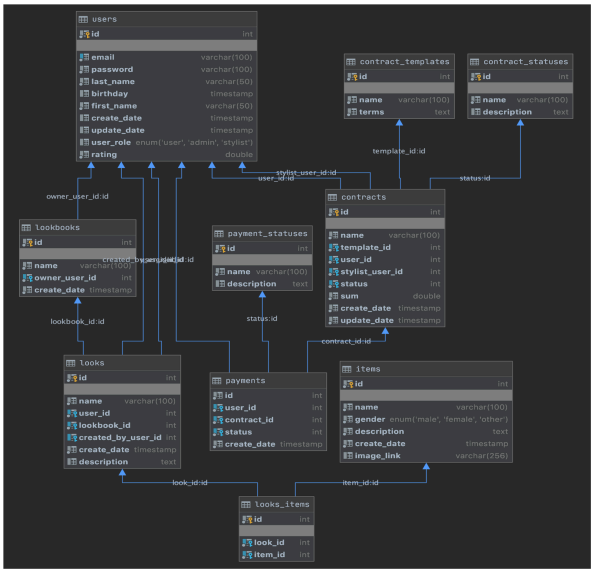


8

Слайд 10:

СХЕМА БАЗИ ДАНИХ ЕЛЕКТРОННОГО СЕРВІСУ

До даного сервісу було створено схему бази даних:

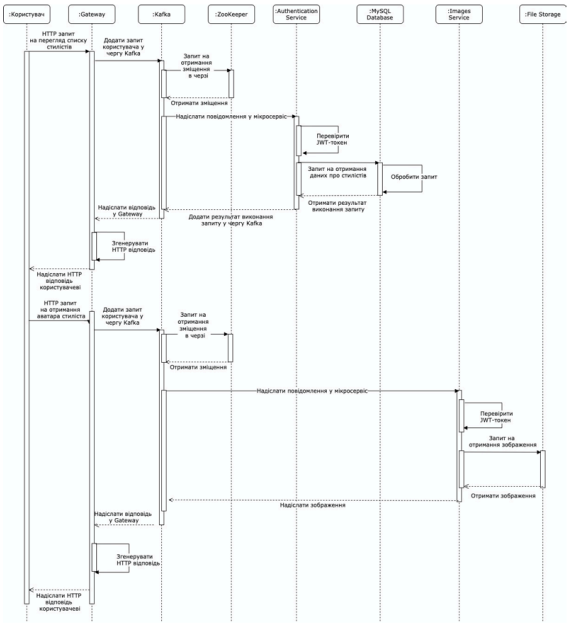


9

Слайд 11:

UML ДІАГРАМА ПОСЛІДОВНОСТІ ПЕРЕГЛЯДУ СПИСКУ СТИЛІСТІВ

1) Користувач надсилає запит у сервіс для отримання даних про стилістів
2) Сервіс додає запит у чергу для подальшої обробки
3) Сервіс перевіряє дані
4) Сервіс отримує дані із бази даних
5) Сервіс генерує і надсилає відповідь із даними про стилістів м
6) Користувач надсилає запит для отримання аватарів стилістів
7) Сервіс додає запит у чергу для подальшої обробки
8) Сервіс перевіряє дані
9) Сервіс отримує зображення із файлового сховища
10) Сервіс генерує і надсилає відповідь користувачеві

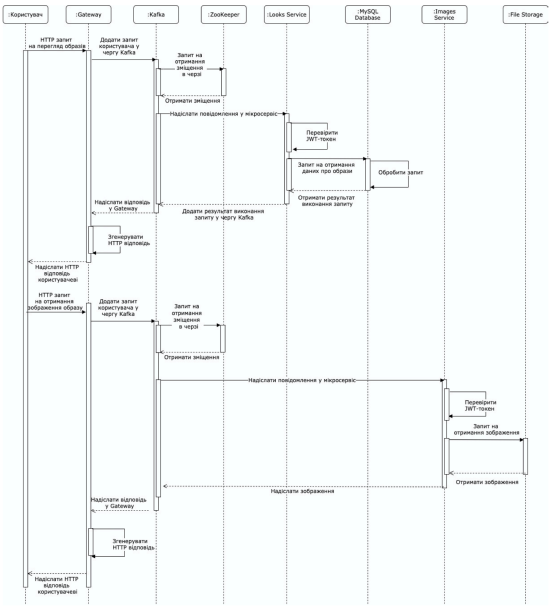


10

Слайд 12:

UML ДІАГРАМА ПОСЛІДОВНОСТІ ПЕРЕГЛЯДУ ОБРАЗУ

1) Користувач надсилає запит у сервіс для отримання текстових даних про образ
2) Сервіс додає запит у чергу для подальшої обробки
3) Сервіс перевіряє дані
4) Сервіс зчитує дані із бази даних
5) Сервіс генерує і надсилає відповідь із даними про образ користувачеві
6) Користувач надсилає запит для отримання зображень образу
7) Сервіс додає запит у чергу для подальшої обробки
8) Сервіс перевіряє дані
9) Сервіс отримує зображення із файлового сховища
10) Сервіс генерує і надсилає відповідь користувачеві

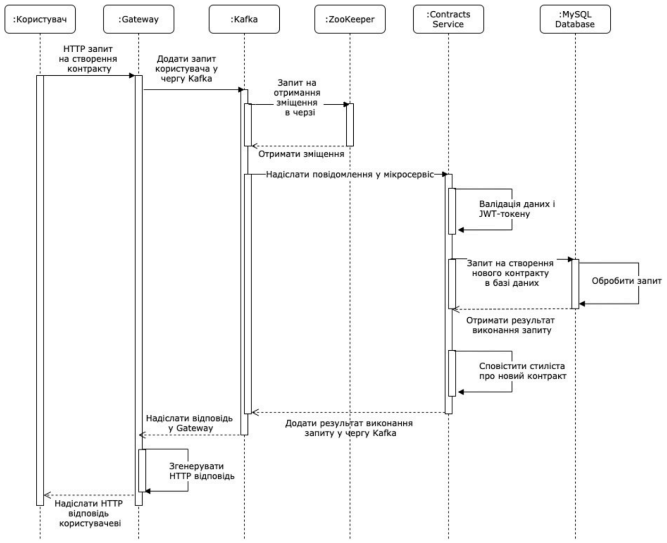


11

Слайд 13:

UML ДІАГРАМА ПОСЛІДОВНОСТІ СТВОРЕННЯ КОНТРАКТУ

1) Користувач надсилає запит у сервіс для створення нового контракту
2) Сервіс додає запит у чергу для подальшої обробки
3) Сервіс перевіряє дані
4) Сервіс створює запис у базі даних
5) Сервіс повідомляє стиліста про новий контракт
6) Сервіс генерує і надсилає відповідь із результатом обробки запиту користувачеві

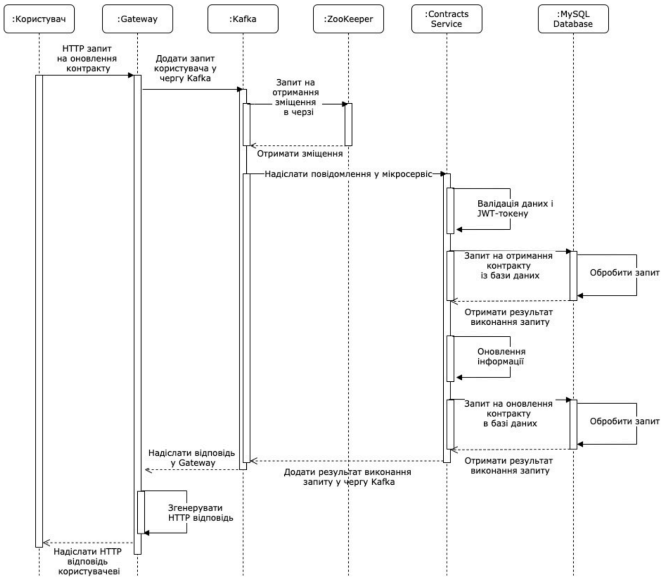


12

Слайд 14:

UML ДІАГРАМА ПОСЛІДОВНОСТІ ОНОВЛЕННЯ КОНТРАКТУ

1) Користувач надсилає запит у сервіс для оновлення контракту
2) Сервіс додає запит у чергу для подальшої обробки
3) Сервіс перевіряє дані
4) Сервіс отримує дані із бази даних
5) Сервіс оновлює дані
6) Сервіс записує оновлені дані у базу даних
7) Сервіс генерує і надсилає відповідь із результатом обробки запиту користувачеві

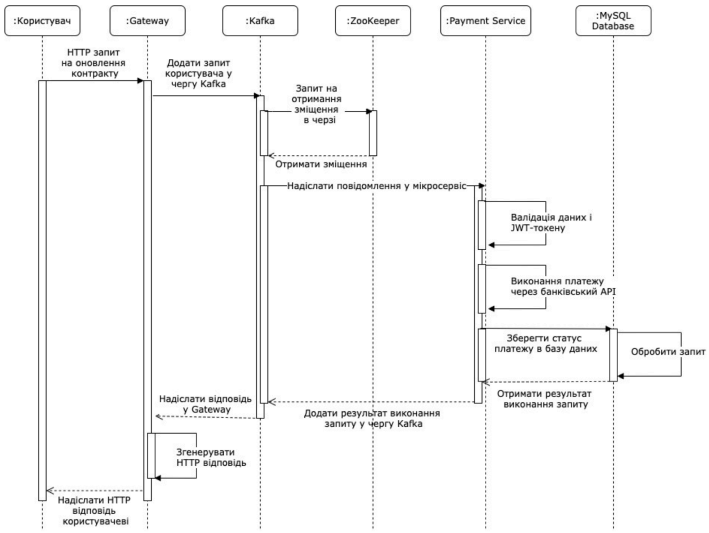


13

Слайд 15:

UML ДІАГРАМА ПОСЛІДОВНОСТІ ОПЛАТИ КОНТРАКТУ

1) Користувач надсилає запит у сервіс для оплати контракту
2) Сервіс додає запит у чергу для подальшої обробки
3) Сервіс перевіряє дані
4) Сервіс виконує платіж використовуючи дані користувача
5) Сервіс зберігає дані про оплату у базу даних
6) Сервіс генерує і надсилає відповідь із результатом обробки запиту користувачеві



14

Слайд 16:

АПРОБАЦІЯ РОБОТИ

- V Міжнародна науково-практична конференція «Концептуальні шляхи розвитку науки»
  Автор: Когут Дмитро Русланович
  Назва: Системний аналіз електронного сервісу підбору гардеробу
  Публікація: http://www.mcnd.ltd.ua/material/2020/травень1.pdf
- Міжнародна конференція «Стратегічні напрямки розвитку науки: фактори впливу та взаємодії»
  Автор: Когут Дмитро Русланович
  Назва: Системний аналіз електронного сервісу підбору гардеробу
  Публікація: https://ojs.ukrlogos.in.ua/index.php/mcnd/issue/view/22.05.2020/305

15

Слайд 17:

ВИСНОВКИ

1) Проведено аналіз аналогічних рішень для підбору гардеробу, визначено, що більшість існуючих електронних сервісів надають послуги використовуючи бізнес модель підписки, що має певні недоліки, а саме непрозору роботу із стилістом, постійні фінансові витрати та відсутність гнучкості.

2) Проведено моделювання предметної галузі із використанням нотації IDEF0 для відображення функцій системи та побудовано UML-діаграми для відображення процесів взаємодії користувача із сервісом.

3) Проаналізовано існуючі архітектури електронних сервісів та розроблено архітектуру електронного сервісу підбору гардеробу, що ґрунтується на мікросервісному шаблоні проектування інформаційних систем та REST-архітектурі для реалізації комунікації між сервісом та клієнтом.

4) Реалізовано базу даних для сервісу із використанням СКБД MySQL.

5) В якості удосконалення запропоновано нову модель взаємодії користувачів із стилістами.

16

Додаток В – Перелік зауважень нормоконтролера

Позначення	Сторінка	Зміст зауваження