Київський національний торговельно-економічний університет

Кафедра цифрової економіки та системного аналізу

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

“Веб-система для оптимізації внутрішніх бізнес-процесів на поліграфічному підприємстві”

Студента 2 курсу, 1м групи,
Спеціальності
051 “Економіка”
спеціалізації
“Цифрова економіка”

_____
підпис студента

Осипчука Владислава
Олеговича

Науковий керівник
кандидат економічних наук,
доцент

_____
підпис керівника

Кулаженко
Володимир Валерійович

Гарант освітньої програми
доктор фізико-математичних
наук, професор

_____
підпис гаранта

Гамалій Володимир
Федорович

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра цифрової економіки та системного аналізу

Освітній рівень магістр

Спеціальність 051 "Економіка"

Спеціалізація "Цифрова економіка"

Затверджую

Зав. кафедри _____ Роскладка А.А.

"15" листопада 2021

Завдання

на випускну кваліфікаційну роботу (проект) студенту

Осипчуку Владиславу Олеговичу

(прізвище, ім'я, по-батькові)

1. Тема випускної кваліфікаційної роботи (проекту)

"Веб-система для оптимізації внутрішніх бізнес-процесів на поліграфічному підприємстві"

Затверджена наказом ректора від *"22" жовтня 2020р. №3066*

2. Строк здачі студентом закінченої роботи *"05" листопада 2021 року*

3. Цільова установка та вихідні дані до роботи

Мета роботи: створення веб-системи поліграфічного підприємства для автоматизації, обліку та оптимізації внутрішніх процесів.

Об'єкт дослідження: поліграфічне підприємство та бізнес-процеси підприємства.

Предмет дослідження: засоби для виконання процесів, пов'язаних з роботою менеджерів з продажу, дизайнерів та директора на поліграфічному підприємстві.

4. Консультанти по роботі (проекту) із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видано	Завдання прийнято
1	Кулаженко В. В.	15.11.2020 р.	15.11.2020 р.
2	Кулаженко В. В.	15.11.2020 р.	15.11.2020 р.
3	Кулаженко В. В.	15.11.2020 р.	15.11.2020 р.

5. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ВЕБ-СИСТЕМ ПІДПРИЄМСТВ ТА ПІДХОДІВ ДО ЇХ СТВОРЕННЯ

1.1. Поняття та сутність веб-системи

1.2. Архітектура та особливості веб-систем

1.3. Існуючі веб-системи підприємств

Висновки до розділу 1

РОЗДІЛ 2. ПОБУДОВА БІЗНЕС-ПРОЦЕСІВ ПОЛІГРАФІЧНОГО ПІДПРИЄМСТВА ПОВ'ЯЗАНИХ З РОБОТОЮ МЕНЕДЖЕРІВ, ДИЗАЙНЕРІВ ТА ДИРЕКТОРА

2.1. Дослідження основних бізнес-процесів поліграфічного підприємства

2.2. Побудова моделі бізнес-процесів поліграфічного підприємства

2.3. Співставлення бізнес-процесів з функціоналом веб-системи

Висновки до розділу 2

РОЗДІЛ 3. СТВОРЕННЯ ВЕБ-СИСТЕМИ ПОЛІГРАФІЧНОГО ПІДПРИЄМСТВА НА ОСНОВІ МІКРОСЕРВІСІВ JAVA

3.1. Створення архітектури веб-системи

3.2 Розробка мікросервісів веб-системи

3.3 Зв’язок з Front-end веб-системи

Висновки до розділу 3

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи (проекту)

№ з/п	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	Вибір теми випускної кваліфікаційної роботи	20.10.2020	20.10.2020
2	Розробка та затвердження завдання на випускну кваліфікаційну роботу	15.11.2020	15.11.2020
3	Вступ	01.03.2021	
4	Розділ 1. Аналіз існуючих веб-систем підприємств та підходів до їх створення	25.06.2021	
5	Розділ 2. Побудова бізнес-процесів поліграфічного підприємства пов'язаних з роботою менеджерів, дизайнерів та директора	01.09.2021	
6	Підготовка статті у збірник наукових статей магістрів	15.09.2021	
7	Розділ 3. Створення веб-системи поліграфічного підприємства на основі мікросервісів java	18.10.2021	
8	Висновки	01.11.2021	
9	Здача випускної кваліфікаційної роботи на кафедру науковому керівнику	05.11.2021	
10	Попередній захист випускної кваліфікаційної роботи	25.11.2021	
11	Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи	28.11.2021	
12	Представлення готової зшитої випускної кваліфікаційної роботи на кафедру	30.11.2021	
13	Публічний захист випускної кваліфікаційної роботи	За розкладом роботи ЕК	

7. Дата видачі завдання «15» листопада 2020 р.

8. Науковий керівник випускної кваліфікаційної роботи (проекту)

Кулаженко В.В

(підпис , прізвище, ініціали)

9. Гарант освітньої програми Гамалій В. Ф.

(підпис , прізвище, ініціали)

10. Завдання прийняв до виконання студент Осипчук В.О.

(підпис , прізвище, ініціали)

11. Відгук наукового керівника випускної кваліфікаційної роботи (проекту)

Науковий керівник випускної кваліфікаційної роботи (проекту)
_____ 05.11.2021 р.
(підпис, дата)
Відмітка про попередній захист Кулаженко В.В.
_____25.11.2021 р.
(ПІБ наукового керівника, підпис, дата)

12. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) студента Осипчука В.О.
(прізвище, ініціали)
може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Гамалій В.Ф.
(підпис, прізвище, ініціали)
Завідувач кафедри_____ Роскладка А.А.
(підпис, прізвище, ініціали)

«_____»_____2021 р.

Анотація

У цій роботі досліджено поняття веб-систем, веб-додатків та веб-сайтів, їх особливості та види, досліджено архітектури веб-систем, їх переваги, недоліки та принципи використання. Обґрунтовано застосування кожного виду архітектури для конкретного додатку. Досліджено існуючі додатки для управління поліграфічним підприємством. Досліджено організаційну структуру поліграфічного підприємства Starter та важливість побудови моделі бізнес-процесів підприємства.

У практичній частині роботи побудовано модель організаційної структури поліграфічного підприємства Starter, модель його бізнес-процесів, реалізовано веб-систему на основі мікросервісної архітектури за допомогою таких технологій, як Java 8, Spring, Spring Boot, MySQL.

Ключові слова: моделювання бізнес-процесів, веб-система, веб-додаток, мікросервіси.

Anotation

This research explores the concepts of web systems, web applications and websites, their features and types, explores the architectures of web systems, their advantages, disadvantages and principles of use. The research includes using of each type of architecture for a specific application. The existing applications for the management of a printing company are studied. The organizational structure of the company Starter and the importance of building a model of business processes of the enterprise are studied.

In the practical part of work the model of organizational structure of the polygraphy company Starter, a models of it`s business processes is created, implemented a web system based on microservice architecture using technologies such as Java 8, Spring, Spring Boot, MySQL.

Keywords: business process modeling, web system, web application, microservices.

ЗМІСТ

ВСТУП

В сучасному цифровому світі при активному розвитку мережі Інтернет створення сайтів для надання різного роду послуг та інформації стає дуже актуальним і необхідним явищем. Сайти та додатки стали масовим продуктом і кожна компанія має власну сторінку в мережі, більші ж за розмірами компанії мають власні сайти та додатки. Зі збільшенням конкуренції, розвитком технологій та збільшень масштабів підприємств необхідний постійний розвиток технологій та додатків. Збільшення можливості користувачів, автоматизація деяких процесів, обмін даними з користувачами та в середині підприємства - все це допомагає успішному управлінню бізнесом. Різноманітність сайтів та додатків також зростає з кожним днем, існують сайти-візитки з невеликою кількістю інформації та контенту, існують інтернет-каталоги для покупки товарів чи послуг з цінами, характеристиками, фото та описами цих товарів чи послуг, існують системи для внутрішнього використання підприємствами і ще багато інших видів сайтів.

Для створення веб-системи найважливішим є обрання правильної архітектури для подальшої розробки і майбутнього супроводу системи.

Аналіз останніх досліджень та публікацій. На сьогоднішній день проводиться доволі багато наукових досліджень проблеми створення веб-систем та веб-додатків, на різних мовах програмування та різних архітектурних рішеннях. Серед наукових праць вітчизняних науковців присвячених дослідженню та створенню веб-систем на Java можна виділити роботи Герасимова В. В. Пономарьова І. В., Ліщенко О. О. [1], О.С. Маліцького[2] та Бурлакова А.А., Балишина О.О[3].

Серед зарубіжних вчених значний вплив на розвиток теоретичних і практичних знань у розробленні веб-додатків на Java зробили Joshua Bloch, Cay S. Horstmann, Bruce Eckel.[4-6]

Актуальність цього дослідження полягає в необхідності оптимізувати роботу персоналу поліграфічного підприємства STARTER, а також у потребі дослідити принципи створення веб-системи.

Метою роботи є створення веб-системи для поліграфічного підприємства для оптимізації внутрішніх бізнес-процесів поліграфічного підприємства.

Об'єктом дослідження є поліграфічне підприємство.

Предметом дослідження є процеси, пов'язані з роботою менеджерів з продажу, дизайнерів та директора на поліграфічному підприємстві.

Для досягнення поставленої мети необхідно вирішити наступні **завдання:**

- Дослідити поняття, сутність, архітектуру та особливості веб-систем;
- Проаналізувати існуючі веб-системи поліграфічних веб-систем;
- Дослідити основні бізнес-процеси поліграфічного підприємства, пов'язаних роботою менеджерів, дизайнерів та директора;
- Створити модель основних бізнес-процесів описаного поліграфічного підприємства;
- Розробити архітектуру веб-системи поліграфічного підприємства на базі моделі його бізнес-процесів;
- Розробити базу даних та мікросервіси веб-системи поліграфічного підприємства;
- Створити контроллери для зв'язку з front-end веб-системи.

Теоретична значущість випускної кваліфікаційної роботи полягає в систематизації існуючих видів веб-систем та веб-додатків, теоретичних принципів розробки веб-додатків, що стали основою для визначення практичних розробок.

Практичне значення отриманих в рамках виконання випускної кваліфікаційної роботи результатів полягає в розробці моделі бізнес-процесів поліграфічного підприємства Starter та розробці веб-системи поліграфічного підприємства на основі розробленої моделі бізнес-процесів.

Апробація результатів досліджень. За результатами проведеного дослідження опубліковано наукову статтю на тему “Особливості мікросервісного підходу при розробці додатків”, яка увійшла до збірника наукових статей студентів КНТЕУ “Цифрова економіка”, м. Київ 2021 рік.

Структура роботи. Робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи - 54 сторінок. Вона містить 6 таблиць, 23 рисунки та 12 додатків. Кількість використаних джерел - 30, їх список наведений на 3 сторінках.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ВЕБ-СИСТЕМ ПІДПРИЄМСТВ ТА ПІДХОДІВ ДО ЇХ СТВОРЕННЯ

1.1. Поняття та сутність веб-системи

Уся інформація в Інтернеті зберігається у вигляді файлів, які розташовані на серверах, постійно підключених до ліній зв'язку, через які відбувається доступ до них. Усі файли в сукупності – це ресурси Інтернету. Самі ресурси (файли) можуть бути різних типів: текстові, гіпертекстові, звукові, графічні, електронні листи та ін. Усі файли організовані в певні структури, з яких їх можна дістати за допомогою протоколів доступу. Для кожного ресурсу Інтернету є свій протокол доступу. Наприклад, з файловими архівами працюють за протоколом ftp, з ресурсами Gopher – gopher, з ресурсами WWW – http, з групами новин – news або nntp. Під Інтернет-ресурсами розуміють розміщену на серверах Інтернету текстову, графічну й мультимедіа-інформацію, яка може бути доставлена на комп'ютер користувача за його запитом у формі файлів або наборів файлів (зокрема архівів) різних форматів (html, doc, pdf, txt і ін.). Використання (читання, перегляд) цієї інформації вимагає наявності відповідних програмних засобів (браузерів) на комп'ютері користувача. На сервері інформація може зберігатися статично (у файлах) або формуватися динамічно залежно від запиту користувача (з бази даних). До Інтернет-ресурсів також належать розміщені на серверах виконувані програми (дистрибутиви), які можуть за запитом користувача передаватися на його комп'ютер для подальшого використання[7]

Для доступу до веб ресурсів, їх зчитуванню, зміні, видаленню чи додаванню, користувачі використовують веб-сайти, веб-додатки чи веб-системи.

Веб-сайт – це сукупність загальнодоступних, взаємопов'язаних веб-сторінок, які мають єдине доменне ім'я. Веб-сайти можуть створюватися та підтримуватися окремою особою, групою, бізнесом чи організацією для виконання різноманітних цілей. Разом усі загальнодоступні веб-сайти утворюють

Всесвітню мережу. Хоча це іноді називають «веб-сторінкою», це визначення є хибним, оскільки веб-сайт складається з кількох веб-сторінок. [8]

Веб-сайти представлені майже нескінченною різноманітністю, включаючи освітні сайти, сайти новин, форуми, сайти соціальних медіа, сайти електронної комерції тощо. Сторінки на веб-сайті зазвичай є поєднанням тексту та інших медіа. При цьому немає правил, які диктують форму веб-сайту. Людина може створити веб-сайт з будь-яким контентом та за будь яким шаблоном, просто текст чи додаючи певні схеми чи картинки. І хоча шаблонів веб-сайтів безліч багато сайтів дотримуються стандартного шаблону головної сторінки, яка переходить до інших категорій і вмісту веб-сайту. Домашня сторінка являє собою головну сторінку самого сайту. Кожна сторінка є єдиним HTML-документом, і всі вони з'єднані через гіперпосилання (або просто «посилання»), які можна об'єднати в навігаційну панель для зручності використання. Кожен HTML-документ доповнюється CSS-файлом, який додає стилі документу та робить його приємним на вигляд, іноді для додавання інтерактивності чи певного функціоналу такого як валідація форм під час заповнення на стороні клієнту використовують також JS-файли.

Веб-додаток (Web app) — це прикладна програма, яка зберігається на віддаленому сервері та доставляється через інтерфейс Інтернет браузера. Веб-сервіси за визначенням є веб-додатками, і багато веб-сайтів, хоча й не всі, містять веб-програми. За словами редактора «Web.AppStorm» Джарела Реміка, будь-який компонент веб-сайту, який виконує певну функцію для користувача, кваліфікується як веб-додаток. Веб-додатки можуть бути розроблені для найрізноманітніших цілей і можуть бути використані будь -ким; від організації до окремої особи з багатьох причин. Зазвичай використовувані веб-додатки можуть включати веб-пошту, онлайн-калькулятори або магазини електронної комерції. До деяких веб-програм можна отримати доступ лише за допомогою

певного браузера; проте більшість з них доступні незалежно від веб-переглядача.[9]

Веб-додатки доступні через веб-браузер і вони адаптуються до того пристрою, на якому їх переглядають. Вони не прив’язані для певної системи, і їх не потрібно завантажувати або встановлювати.[10]

Веб-додатки отримують і зберігають інформацію за допомогою скрипту на стороні сервера (на таких мовах, як PHP, ASP, Java тощо), тоді як скрипти на стороні клієнта (в JavaScript і HTML5) представляють відповідну інформацію в інтерфейсі користувача. Ця інформація може мати будь-яку кількість форм. Поширені типи веб-додатків включають кошики для покупок, системи керування вмістом та онлайн-форми. Завдяки своїй універсальності, веб-програми дозволяють людям виконувати різноманітні функції. Для споживачів це включає розміщення інформації, створення списків і запитів про продукти чи послуги через веб-сторінки. Додатки також дозволяють співробітникам обмінюватися документами, спілкуватися один з одним, редагувати файли та спільно працювати над спільними проектами. У нову епоху дистанційної роботи це надзвичайно важливо.[11]

Веб-додатки використовуються коли в використанні веб-сайту недостатньо простого зчитування інформації, як тільки потрібно додати логіку роботи чи доступ до бази даних доцільно розробляти веб-додатки. Вибір мови для написання веб-додатку та складність його написання залежить від логіки, яку потрібно реалізувати при розробці, та того як вона буде використовуватись, супроводжуватись та можливо розширюватись в майбутньому. Якщо проект обіцяє бути масштабним та складним і використовуватись великою кількістю користувачів то доцільно буде писати веб-систему.

Веб-система - це більш потужний веб-додаток призначений для використання великою кількістю користувачів. Веб-система зазвичай має

розмежування прав доступу до різних веб-сторінок та веб-ресурсів, а також зазвичай складається з декількох менших веб-додатків.

Більшість сайтів в мережі Інтернет є веб-додатками, більша частина логіки роботи яких зосереджена на веб-сервері. Користувач при його використанні бачить лише веб-сторінку і прямого доступу до додатку чи бази даних в якій зберігається вся інформація користувач не має. Спілкування між сервером на якому розташований веб-додаток та користувачем відбувається за допомогою запитів, після отримання сервером запиту від користувача на відкриття певної сторінки сервер виконує обчислення, якщо потрібно дістає чи модифікує інформацію в базі даних та відправляє відповідь з усією необхідною інформацію до браузера користувача через мережу з використанням протоколу HTTP після чого браузер формує для користувача веб-сторінку з інформації отриманої від сервера. Такий підхід до роботи дозволяє не тільки приховати всю роботу від користувача, а також дозволяє одному додатку бути клієнтом іншого веб додатку, так за допомогою запитів проходить спілкування різних веб-додатків один з одним.

1.2. Архітектура та особливості веб-систем

Розглядаючи різні веб-системи та веб-додатки їх можна розділити на певні категорії залежно від їх призначення, особливостей використання та технологій та архітектури написання. Кожен додаток має свої особливості, переваги та недоліки тож розподіл додатків на види є дуже суб’єктивним і гнучким.

Якщо розглядати веб-додатки та веб-системи за призначенням, то можна виділити наступні види:

- Рахівники;
- Рейтинги;
- Форми відправки повідомлень;
- Форми реєстрації;
- Форми завантаження ;

- Системи голосування;
- Пошукові системи;
- Content Managment System (CMS);
- Банерні движки та системи ;
- Веб-пошта;
- Персоналізовані веб-системи різного спрямування, які надають комплекс послуг своїм користувачам;
- CRM-системи;
- Блоги та форуми;
- Додатки для зберігання даних;
- Системи онлайн-бронювання;
- eCommerce;
- Інтернет-банкінг;
- Системи електронних платежів;
- Системи управління взаємодією з клієнтами;
- Навчальні веб-додатки;
- Різні корпоративні сайти, що мають розширену та нестандартну функціональність.

Це далеко не всі види веб-додатків, але основні з них, список додатків за призначенням можна доповнити використовуючи більш вузькі призначення, тому можливих варіацій існує безліч.

Якщо ж розділяти веб-системи та веб-додатки за особливостями використання, то потрібно звертати увагу на такі критерії:

- Платформна незалежність чи залежність використання від певної платформи;
- Мова реалізації та допоміжні бібліотеки чи фреймворки;
- Продуктивність, масштабованість;

- Можливості розширення і інтеграції;
- Простота використання, наявність засобів розробки;
- Наявність необхідних програмних бібліотек.

За архітектурним стилем написання веб-додатки можна розділити на 2 основні категорії:

- Монолітні веб-додатки
- Мікросервісні веб-додатки

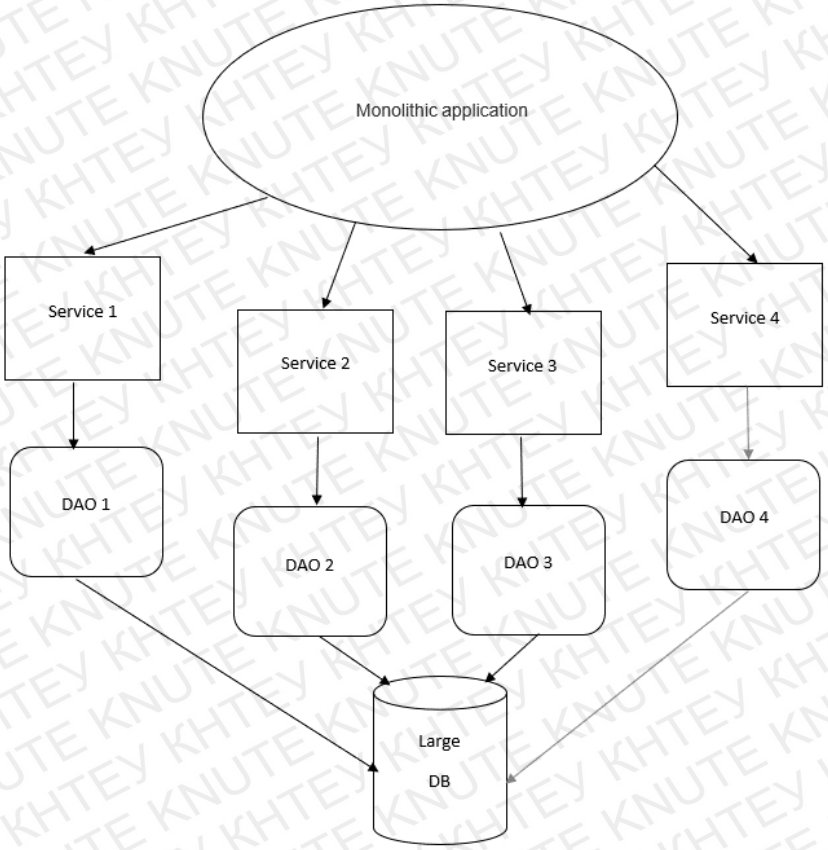

Рис.1.1. Приклад монолітної архітектури веб додатку

Джерело: Авторська розробка

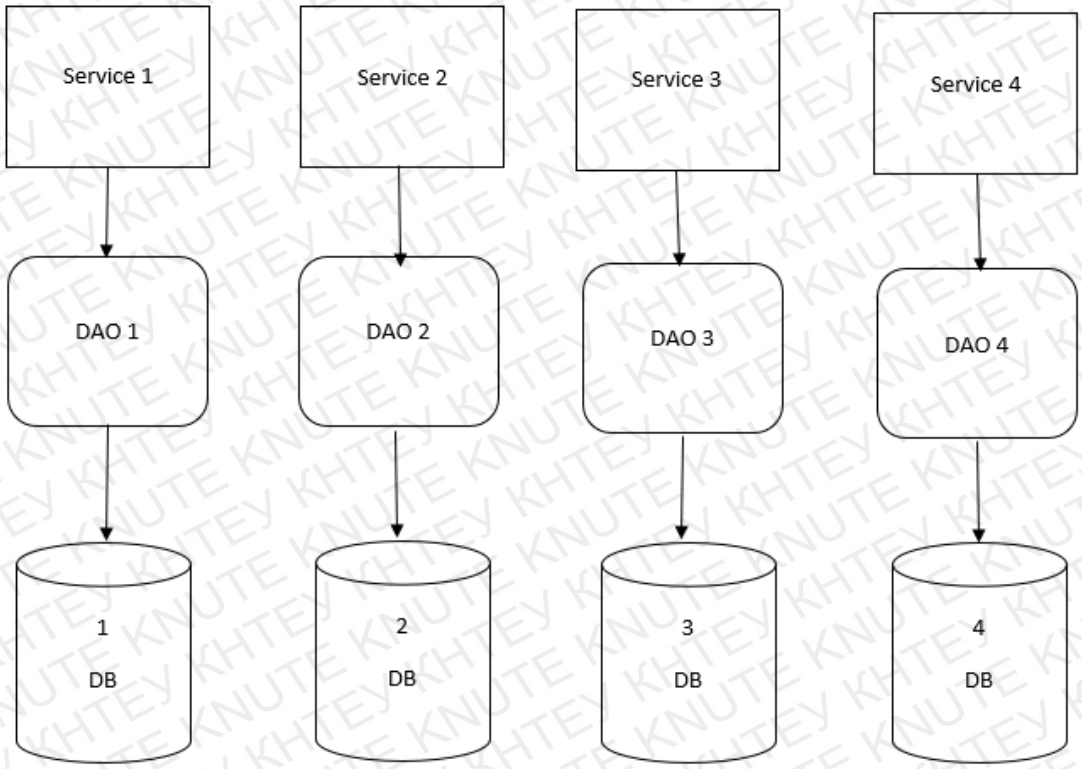

Рис.1.2. Приклад мікросервісної архітектури веб додатку

Джерело: Авторська розробка

Монолітний підхід є найстаршим і досить популярним в написанні програм як на Java, так і з використанням інших мов програмування. Для написання монолітного додатку на 1 сторінку і пари дій із сторінкою (зв'язок з базою даних, відправка даних кінцевому користувачеві, валідації та перевірки) цілком підходить монолітний підхід, де весь вихідний код упаковується в jar або war пакет для розгортання і використання його в одній JVM, одним процесом, його початковий розмір буде в діапазоні 1-100 МБ і досить мала кількість коду.

Проблеми монолітного коду починаються з розростанням проекту, наростання додаткових сторінок, можливостей і функцій проекту, зміну команди, рефакторингу старого коду або затяжному проекті на пару років. Звичайний розмір монолітної програми 100 000 - 1 000 000 рядків сильно пов'язаного між собою коду, новим розробникам для роботи з монолітним додатком спочатку

доведеться ознайомитися з роботою старого коду, при зміні команди розробникам новим розробникам потрібно буде пояснювати як все працює, а при зміні будь-якого підходу потрібно буде проводити рефакторинг більшої частини старого коду.

З цього випливає висновок, що головна проблема монолітних додатків в розмірі цих додатків, в монолітному підході jar або war пакет з додатком працює в одній JVM, один процес на одному сервері.

Знаючи проблему розробники почали шукати рішення і прийшли до висновку, що якщо якась частина коду не є сильно пов'язаної із загальною структурою то її можна винести в окремий проект і налаштувати між цією і основною частиною зв'язок для спілкування. Наприклад, відділ закупівель використовує одну частину програми, а відділ маркетингу зовсім іншу і між собою вони не взаємодіють, а просто знаходяться на одному сайті.

У практичній площині це означає, що замість виклику методу/компонента відповідаючого за зв'язок з сервісом відділу закупівель всередині нашого контролера, цей метод/bean-компонент з усіма його допоміжними класами можна перенести в свій власний проект Maven/Gradle і розгорнути його незалежно від решти монолітного коду. Завдяки такому розподілу за розробку або підтримку даного міні-проекту може взятися зовсім інша команда і розробляти його повністю незалежно від іншої частини програми. Кожен мікросервіс є сервісно-орієнтованою архітектурою, але не навпаки[12]

Отже, додаток розділений на 2 частини, основна частина програми та маленька частина, яка тепер діє незалежно від основної, але так як вони спочатку були одним додатком, а зараз це 2 абсолютно різних додатки то між ними потрібно налагодити зв'язок для спілкування. Є два варіанти зв’язку: синхронна взаємодія і асинхронна взаємодія.

Синхронна взаємодія використовується за допомогою HTTP/REST сервісів, які повертають XML або JSON - хоча це не є обов'язковим. REST/HTTP - комунікація використовується коли потрібна негайна відповідь.

Асинхронний мікросервісний зв'язок зазвичай здійснюється за допомогою обміну повідомленнями за допомогою реалізації JMS і/або за допомогою протоколу, такого як AMQP. Асинхронна взаємодія використовується коли не потрібна негайна відповідь, наприклад, користувачі натискають кнопку «купити зараз», і ви хочете згенерувати рахунок-фактуру, ця операція не повинна відбуватися в рамках циклу запиту-відповіді користувача на покупку.

Отже, мікросервіс - це архітектурний стиль, який структурує додаток як набір сервісів, які є незалежними, слабо пов'язаними, організованими навколо бізнес-можливостей і належать невеликій команді[13]

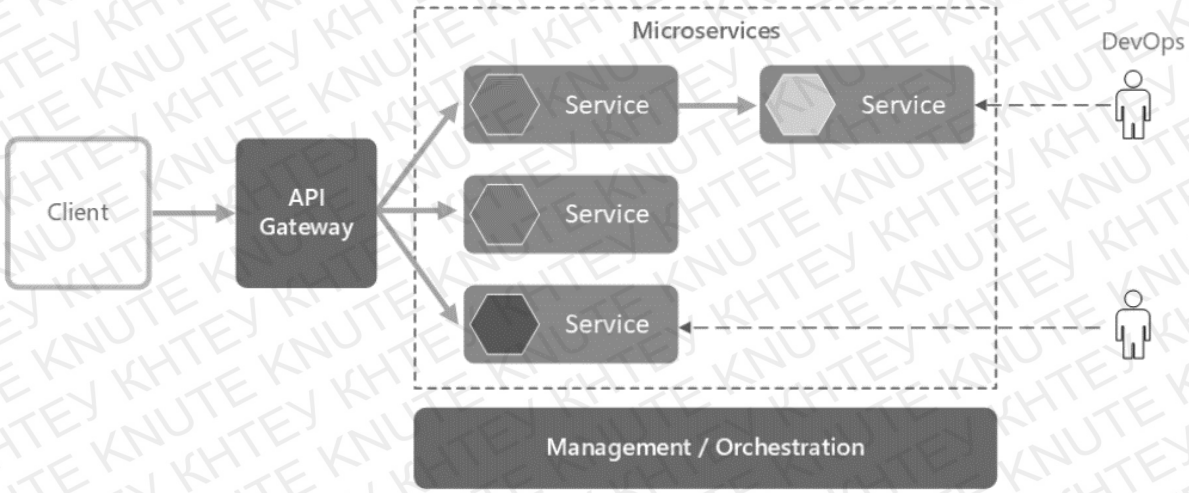

Рис.1.3. Доступ користувача до мікросервісів

Джерело: [14]

Моноліти характеризуються наступними особливостями:

- Великий розмір програми;
- Тривалі цикли випуску;
- Великі команди;

Типові проблеми монолітів включають:

- Проблеми масштабованості;
- Прийняття нових технологій;
- Додавання нових процесів;
- Складність у автоматизації тестів;
- Важко адаптуватися до сучасних практик розробки;
- Важко адаптуватися до швидкого зростання проекту.

Кожен мікросервіс є[15]:

- Легким для обслуговування та тестування - забезпечує швидкий та частий розвиток та впровадження;
- Слабо поєднаний з іншими службами - дає змогу групі працювати незалежно більшість часу над своїми послугами без впливу на них інших послуг та без впливу на інші послуги;
- Незалежним в розгортанні - дозволяє групі розгортати свою службу без узгодження з іншими командами;
- Зручним в розробці невеликою командою - важливо для високої продуктивності, уникаючи високого рівня спілкування великих команд.

Переваги мікросервісів:

- Використання нових технологій і процесів адаптація стає простішим. Можна спробувати нові технології з новими мікросервісами, які створюються. Більшість розробників цілком здатні зрозуміти новий синтаксис і, як правило, можуть зрозуміти новий синтаксис за відносно короткий проміжок часу. Що потребує набагато більше часу - це розуміння викривленої та викривленої логіки домену, розкиданої по 6 пакетах та бібліотеках, об’єднаних у одну монолітну програму. [16]
- Більш швидкі цикли випуску;
- Масштабування за допомогою хмари;

- Можливість написання різних мікросервісів на різних мовах програмування;
- Менша команда. Накладні витрати на спілкування та координацію різко зростають із збільшенням команди. У найгіршому з можливих випадків, коли всім учасникам проекту необхідно спілкуватися та координувати свої дії з усіма іншими, вартість цих зусиль зростає як квадрат кількості людей у команді. Насправді це настільки потужний ефект, що велика команда не могла сподіватися досягти мети, щоб кожен координував свої зусилля. Але маленька команда могла б[17].
- Легка підтримка коду;
- Можливість швидкого та простого додавання нових модулів до додатку в якості нових мікросерісів;

Хоча розробка кількох невеликих компонентів може здатися легкою, існує ряд властивих їй складнощів, пов'язаних з архітектурою мікросервісів:

- Потрібно швидке налаштування: не можна витрачати місяць чи більше на налаштування кожного мікросервіса. Потрібно бути в змозі швидко створювати мікросервіси;
- Автоматизація: оскільки замість моноліту є ряд дрібніших компонентів, необхідно автоматизувати всі збірки, розгортання, моніторинг.
- Видимість: тепер є кілька невеликих компонентів для розгортання та обслуговування. Може бути, 100 або 1000 компонентів. Потрібно відслідковувати і визначати проблеми автоматично, а також мати видимість навколо всіх компонентів.
- Обмежений контекст: дуже складно визначити де закінчується один мікросервіс і має починатись наступний;
- Управління конфігураціями: необхідно підтримувати конфігурації для сотень компонентів в різних середовищах.

- Стек мікросервісів: Якщо мікросервіс в нижній частині ланцюжка викликів виходить з ладу, він може вплинути на всі інші мікросервіси. Мікросервіси повинні бути відмовостійкими.
- Узгодженість: не може бути широкого спектра інструментів, які вирішують одну і ту ж проблему. Хоча важливо стимулювати інновації, важливо також мати децентралізоване управління мовами, платформами, технологіями та інструментами, використовуваними для реалізації / розгортання / моніторингу мікросервісів.

1.3. Існуючі веб-системи підприємств

Існує дуже багато готових веб-додатків для використання підприємствами, які надають готові рішення для автоматизації роботи. Розглядаючи ринок поліграфічних підприємств серед найпоширеніших та найвідоміших є лише настільні додатки такі як: ASystem (рис. 1.4), Sevit Print Shop Manager (рис. 1.5), PrintEffect (рис. 1.6), 1С Поліграфія 8 (рис. 1.7).

Настільні додатки мають свої переваги, але веб-додатки більш зручні для використання користувачем. Серед переваг веб-додатків можна виділити:

- Не потрібно встановлювати на комп'ютері.
- Не потрібно застосовувати перевірку версії на клієнтській машині.
- Оновлення простіше та проходить без участі користувачів.
- Полегшує виправлення помилок, усунення багів та внесення коректив .
- Можливість отримання доступу з будь-якого браузера .
- Незалежність від платформи та характеристик.
- Підтримка та обслуговування простіші.

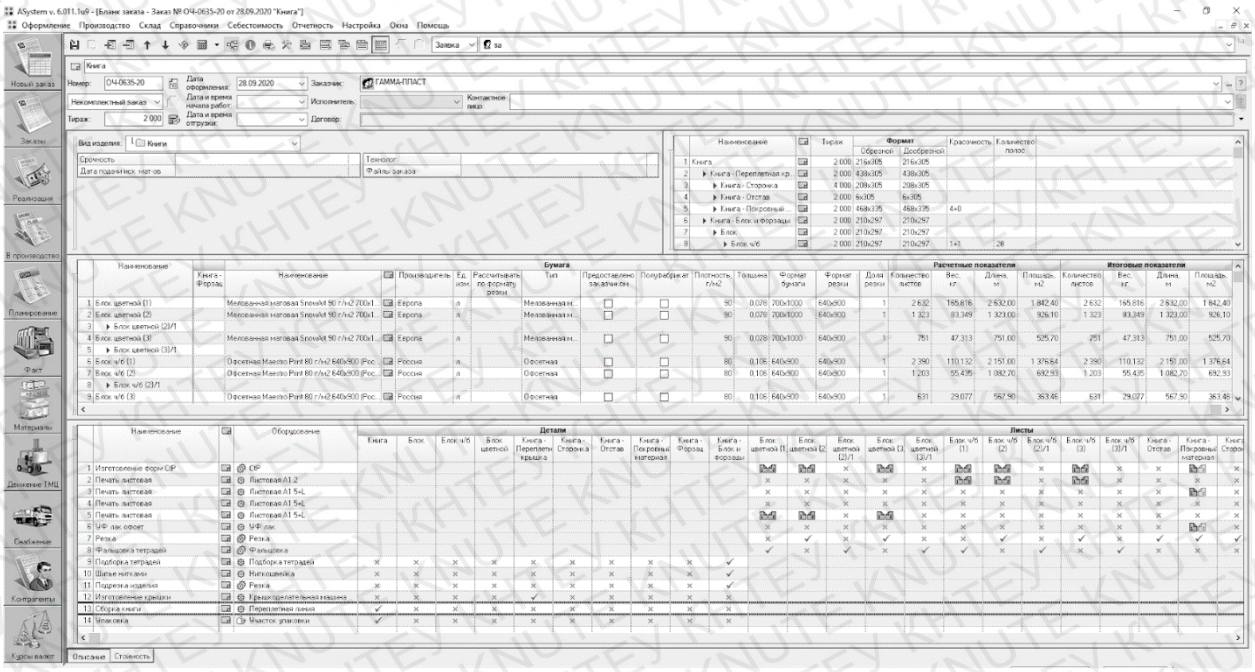

Рисунок 1.4. Додаток ASystem

Джерело: Авторська розробка

ASystem – це система управління поліграфічним та пакувальним виробництвом класу MIS (Management Information System). Забезпечує оперативне управління основним виробництвом та бізнес-процесами друкарні, а також управлінський облік та аналіз результатів виробничої діяльності.

ASystem дозволяє:

- проводити розрахунок вартості та собівартості замовлення, часу виконання виробничих операцій, а також потреби в основних матеріалах та папері;
- контролювати всі стадії виконання замовлення - від отримання макета від клієнта до відвантаження готової продукції;
- реєструвати операції з реалізації готової продукції та відстежувати стан взаєморозрахунків із клієнтами;
- планувати завантаження обладнання та оцінювати час виконання кожної стадії технологічного процесу;

- планувати витрату матеріалу та паперу, відстежувати їх своєчасну підготовку;
- управляти процесом виробництва замовлення, оцінювати, як зміна технології впливає собівартість;
- реєструвати фактичну витрату ресурсів;
- аналізувати результати виробничої та комерційної діяльності підприємства;
- обмінюватися даними з різними зовнішніми системами (наприклад, 1С, Bitrix24 та ін.)

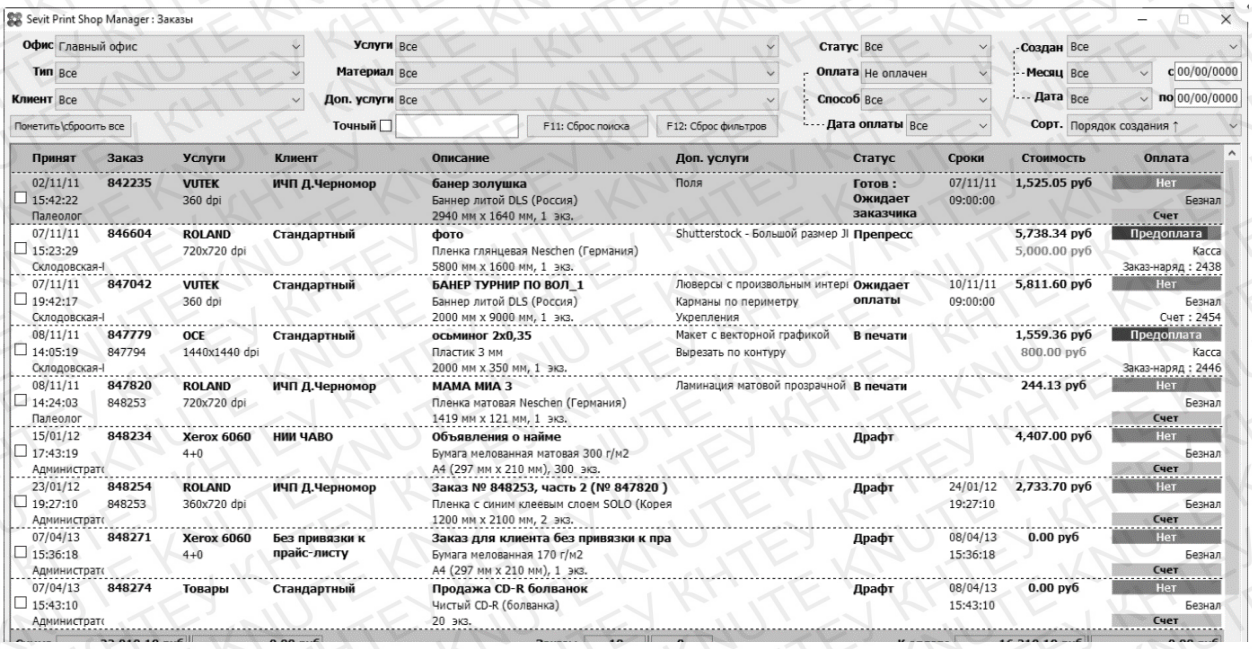

Рисунок 1.5. Додаток Sevit Print Shop Manager

Джерело:Авторська розробка

У Sevit Print Shop Manager можна приймати замовлення (з різних джерел), проводити розрахунок вартості (цифровий, широкоформатний друк та інші типи), контролювати хід виконання замовлення, а також формувати автоматизовану базу клієнтів. При цьому активні замовлення зручно відображаються - у таблиці по кожному можна переглянути коротку інформацію:

номер замовлення, тип послуги, опис до замовлення, перелік додаткових послуг, терміни на виконання, статус, загальну вартість проекту, чи було зроблено передплату або замовлення сплатили повністю.

Рисунок 1.6. Додаток PrintEffect

Джерело: Авторська розробка

PrintEffect - це система управління класу MIS (Management information system) для малого поліграфічного бізнесу. Реалізує єдиний інформаційний простір основних служб друкарні. Забезпечує повний контроль за проходженням замовлень у виробництві.[18]

Основні функції даного додатку є:

- Професійний розрахунок поліграфічних замовлень;
- Управління продажами;
- Управління виробництвом;
- Управління складськими запасами;
- Собівартість та управлінський аналіз.

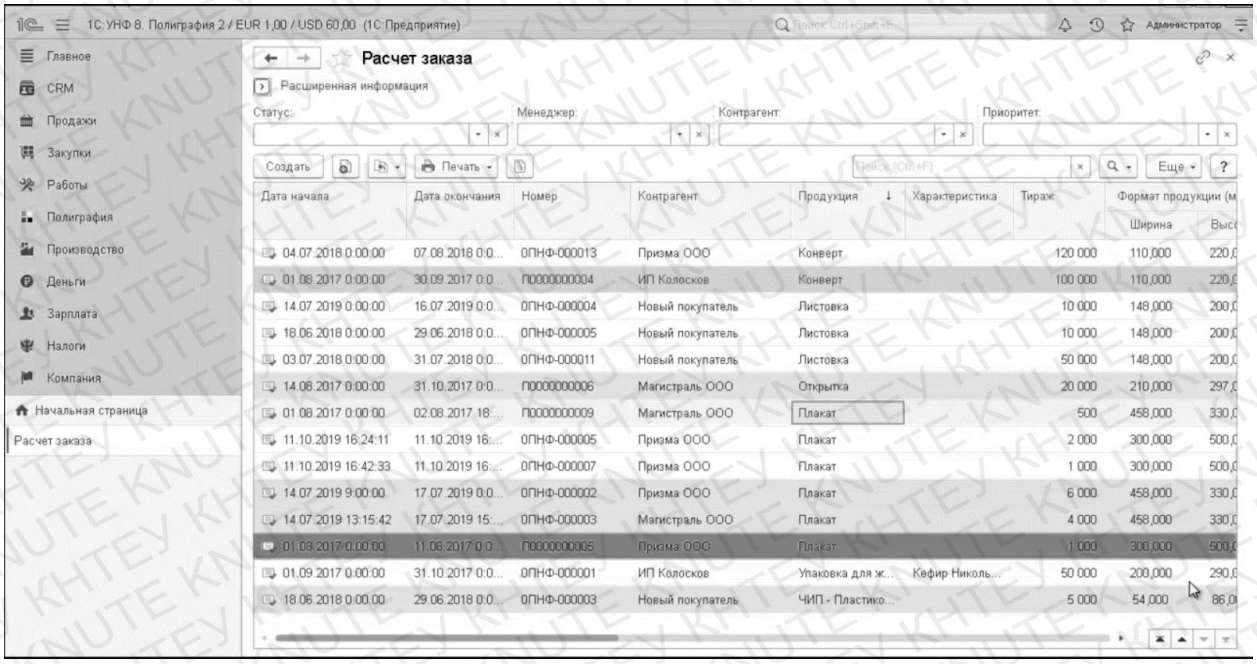

Рисунок 1.7. Додаток 1С Поліграфія 8

Джерело: Авторська розробка

Програма для друкарень дозволять автоматизувати різні ділянки поліграфічного підприємства — від обліку матеріалів до планової вартості поліграфічних замовлень.

1С:Поліграфія забезпечує вирішення всіх основних завдань, які потрібні друкарні[19]:

- Категорії клієнтів, аналіз каналів залучення, аналіз роботи менеджерів;
- Інтеграція IP-телефонії, пошти SMS, робота з лідами, всі контакти та події в одному вікні;
- Оповіщення по SMS та e-mail про статуси замовлень, нагадування про завдання співробітникам, система взаємодій;
- Калькуляція будь-яких типів замовлень: офсет, цифра, флексодрук, широкоформатні, складені, збірні спуски і т.д.;
- Оперативне та довгострокове планування, план робіт для кожного робочого центру та виконавця;

- MES-система, реєстрація факту виконавцями, підключення тачскринів та ШК;
- Контроль та планування складських залишків, пористі склади, штрихкодування;
- Транспортні витрати, рольовий облік матеріалів, облік у кількох одиницях виміру, ордерні склади;
- Маршрутні листи, мобільні програми для кур'єрів, інтеграція з Яндекс.Доставка;
- Розрахунок бонусів менеджерам, облік відрядної зарплати, нарахування та утримання, кадровий облік;
- Аналіз собівартості, рентабельності, прибутку, всіх статей витрат у розрізі план-факт;
- Доходи та витрати за статтями, монітори керівника, підсумковий фінансовий результат компанії.

Висновки до розділу 1

Проаналізувавши інформацію у першому розділі, можна побачити що хоча на ринку існують готові програмні рішення для використання поліграфічними підприємствами всі вони декілька застарілі та мають різні набори переваг та недоліків, а головне всі вони являються настільними додатками.

При створенні власного додатку потрібно обрати правильну архітектуру яка полегшить розробку та майбутній супровід системи. Для спрощення створення архітектури потрібно обрати який підхід буде використовуватись при розробці: архітектурний чи монолітний, переваги та недоліки яких було досліджено в розділі 1. Зважаючи на майбутнє розширення бізнесу підприємства та можливої необхідності в оновлення та додаванню нового функціоналу найкращим виходом буде використовувати мікросервісну архітектуру при розробці веб-системи.

Використання веб-додатку створеного власноруч допоможе спростити використання користувачам та також дасть змогу обрати оптимальний набір функцій щоб оптимізувати переваги та недоліки і створити веб-систему які підходить для конкретного підприємства найбільше.

РОЗДІЛ 2. ПОБУДОВА БІЗНЕС-ПРОЦЕСІВ ПОЛІГРАФІЧНОГО ПІДПРИЄМСТВА ПОВ'ЯЗАНИХ З РОБОТОЮ МЕНЕДЖЕРІВ, ДИЗАЙНЕРІВ ТА ДИРЕКТОРА

2.1. Дослідження основних бізнес-процесів поліграфічного підприємства

Під бізнес-процесом розуміють ланцюг логічно пов’язаних, повторюваних дій, у результаті яких використовуються ресурси підприємства для переробки об’єкта (фізично і віртуально) з метою досягнення певних результатів або продукції для задоволення внутрішніх і зовнішніх споживачів. Тому діяльність будь-якого підприємства необхідно розглядати як мережу процесів, у якій протікають взаємозалежні, взаємозв’язані бізнес-процеси та реалізують функції підприємства. [20]

№ п/п	Класифікаційна ознака, автор	Види БП
1	За функціональною ознакою або рівнем створення цінності	- основні бізнес-процеси; - підтримуючі (обслуговуючі); - забезпечуючи; - бізнес-процеси розвитку; - бізнес-процеси управління діяльністю підприємства;
2	За характером продукту	- адміністративні бізнес-процеси; - виробничі бізнес-процеси
3	По відношенню до клієнта	-зовнішні; - внутрішні
4	За рівнем ієрархії	- процеси верхнього рівня; - підпроцеси; - операції.
5	За ознакою часу	- циклічні (постійно повторюються); - періодичні; - неперервні; - дискретні
6	За структурою взаємодії	- горизонтальні; - вертикальні; - індивідуальні
7	За напрямком діяльності	- типові; - специфічні
8	За ступенем складності	- прості; - складні
9	За рівнями значимості	- суперпроцеси; - гіперпроцеси; - метапроцеси; - субпроцеси; - макропроцеси; - мікропроцеси.
10	За способом зв’язку процесів	- локальні; - інтегральні
11	За ступенем впливу на результативність	- ключові (вирішальні); - ризикові
12	По відношенню до діяльності підприємства	- прямі; - зворотні
13	За значимістю	- первинні; - вторинні
14	За виходом процесу	- торговельні; - сервісні; - інформаційні
15	За ступенем виробництва	- автоматизовані; - автоматичні
16	Ефективність бізнес-процесів	-економічна; - соціальна; - управлінська
17	В залежності від потреб споживачів	- існуючі потреби; - приховані потреби

Рисунок 2.1. Класифікація бізнес-процесів підприємства за відмінними ознаками (систематизовано)

Джерело: [20]

Одним з перших основних етапів побудови процесно-орієнтованої організації і управління діяльністю підприємства є виділення й класифікація бізнес-процесів. Основу для класифікації бізнес-процесів становлять чотири базові категорії[21]:

- основні бізнес-процеси;
- забезпечуючі бізнес-процеси;
- управлінські бізнес-процеси;
- бізнес-процеси розвитку.

Для створення правильної архітектури веб-системи поліграфічного підприємства потрібно виділити операційні та управлінські процеси які пов’язані з роботою менеджерів та дизайнерів.

Операційні бізнес-процеси передбачають використання і розвиток усіх виробничих потужностей з метою досягнення конкурентоспроможності підприємства. Операційні бізнес-процеси взаємоузгоджені з прийняттям рішень, пов’язаних з розробленням виробничого процесу й інфраструктури, необхідної для його функціонування (системи планування, забезпечення системи управління якістю та мотивації праці тощо). Розроблення виробничого процесу полягає у виборі оптимальної технології, складанні графіків процесу, визначенні товарно-матеріальних потоків кожного структурного підрозділу[22]

Управлінські бізнес-процеси – це бізнес-процеси, які охоплюють весь комплекс функцій менеджменту на рівні кожного бізнес-процесу й бізнес-системи в цілому.

Модель бізнес-процесів підприємства створюється в 4 етапи :

- побудова схеми організаційної структури підприємства;
- побудува матриці відповідальності;
- розпис паспортів бізнес-процесів;
- побудува моделі бізнес-процесів підприємства.

Starter - це маленьке поліграфічне підприємство. Його особливістю є те, що процес безпосереднього виробництва виконується друківнями-підрядниками. Тобто, менеджери компанії приймають замовлення, якщо потрібно, то дизайнери компанії розробляють макет майбутньої продукції, потім менеджери передають

макет підрядникам. Друківню здійснюють доставку готової продукції на адресу компанії Starter і після перевірки якості та кількості продукції, менеджер передає замовлення клієнту. Організаційна структура веб-системи поліграфічного підприємства Starter має 3 складові (рис 2.2.) директор, менеджери та дизайнери

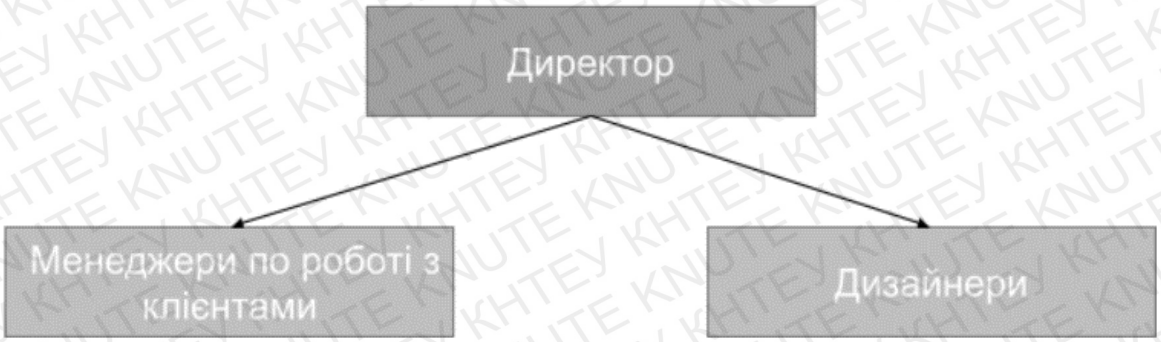

Рисунок 2.2. Схема організаційної структури веб-системи поліграфічного підприємства

Джерело: Авторська розробка

Менеджери приймають замовлення від клієнтів, додають опис необхідного дизайну для дизайнерів, передають замовлення на виробництво, приймають замовлення з виробництва, видають замовлення клієнту, збирають відгуки від клієнтів.

Дизайнери створюють дизайни за описом від менеджерів і передають готові роботи менеджерам.

Директор слідкує за показниками ефективності роботи компанії, оплачує роботу аутсорсингових компаній, виплачує заробітну плату працівникам, наймає нових працівників, веде ділові переговори з постачальниками.

Спочатку визначимо матрицю відповідальності поліграфічного підприємства(таблиця 2.1) і за допомогою матриці визначимо описання паспортів бізнес-процесів.

Таблиця 2.1

Матриця відповідальності поліграфічного підприємства

Назва бізнес-процесу	Менеджери	Дизайнери
Процес прийому замовлення	Вик	-
Процес створення макету	Відп	У
Процес виробництва	Відп	-
Процес передачі замовлення клієнту	Вик	-
Процес збору відгуків від клієнтів	Вик	-

Джерело: Авторська розробка

Умовні позначення:

Відп - відповідальний за бізнес-процес (несе відповідальність за результат бізнес-процесу, але може його не виконувати)

Вик - виконавець (і виконує бізнес-процес, і несе відповідальність за його результат)

У - учасник (бере участь у бізнес-процесі)

Перший паспорт бізнес-процесів - процес прийому замовлення (таблиця 2.2). Клієнт замовляє продукцію в поліграфічному підприємстві, передає менеджеру вимоги до дизайну продукції чи готовий дизайн, якщо розробка від дизайнера не є необхідною, сплачує повністю чи надає передплату.

Таблиця 2.2

Паспорт бізнес-процесу Процес прийому замовлення

Параметр		Значення
Найменування процесу		Процес прийому замовлення
Результати		Прийняте замовлення
Виходи БП	матеріальні	-
	інформаційні	Технічне завдання для дизайнера (якщо це потрібно)
Входи БП	матеріальні	Гроші (передплата)
	інформаційні	Запит на виконання робіт, вимоги до готового продукту
Власник БП		Менеджер
Управлінські (нормативні) документи		Правила спілкування з клієнтами, Правила оформлення технічного завдання
Учасники	персонал	Менеджер
	засоби реалізації	веб-сайти, інтернет, телефон, соціальні мережі, електронна пошта, комп’ютер
Споживач		Дизайнер (якщо потрібно відмальовувати макет)
Попередній процес		-
Наступний процес		Процес створення макету або Процес виробництва

Джерело: Авторська розробка

Менеджер обробляє замовлення клієнта і описує дизайн та його складність для дизайнера за правилами написання технічного завдання або передає замовлення на виробництво, якщо клієнт має готовий дизайн. Клієнт може зробити замовлення через декілька каналів зв’язку: зателефонувати, залишити заявку на сайті компанії, в соціальних мережах (Instagram, Facebook, LinkedIn), надіслати повідомлення на електронну пошту чи особисто переговорити з менеджером в офісі компанії. Відповідальність за виконання цього бізнес-процесу несе менеджер.

Наступний процес відбувається, якщо клієнт не має власного дизайну або до його макету треба вносити правки (таблиця 2.3).

Таблиця 2.3

Паспорт бізнес-процесу Процес створення макету

Параметр		Значення
Найменування процесу		Процес створення макету
Результати		Готовий макет у графічному редакторі
Виходи БП	матеріальні	-
	інформаційні	Файл з макетом
Входи БП	матеріальні	-
	інформаційні	Технічне завдання
Власник БП		Менеджер
Управлінські (нормативні) документи		-
Учасники	персонал	Учасники

Продовження таблиці 2.3

Параметр		Значення
	засоби реалізації	Графічні редактори (Adobe Photoshop, Adobe Illustrator, Figma тощо), Інтернет
Споживач		Менеджер
Попередній процес		Процес прийому замовлення
Наступний процес		Процес виробництва

Джерело: Авторська розробка

Менеджер передає створене технічне завдання дизайнеру. Дизайнер виконує завдання, використовуючи графічні редактори Adobe Photoshop, Adobe Illustrator, Figma та інші. Після закінчення виконання роботи дизайнер передає файл з макетом менеджеру та відмічає затрачений час.

Наступним кроком є передача макету на виробництво (таблиця 2.4). Оскільки підприємство не має власної друкарні, то ці послуги надаються аутсорсинговою компанією.

Таблиця 2.4

Паспорт бізнес-процесу Процес виробництва

Параметр		Значення
Найменування процесу		Процес виробництва
Результати		Готовий результат
Виходи БП	матеріальні	Акт виконаних робіт, готовий продукт
	інформаційні	-

Продовження таблиці 2.4

Параметр		Значення
Входи БП	матеріальні	Гроші
	інформаційні	Файл з макетом
Власник БП		Менеджер
Управлінські (нормативні) документи		-
Учасники	персонал	Учасники
	засоби реалізації	
Споживач		Менеджер
Попередній процес		Процес прийому замовлення або Процес створення макету
Наступний процес		Процес передачі замовлення клієнту

Джерело: Авторська розробка

Для цього менеджер залишає заявку на сайті компанії-друкарні, прикріплюючи файли макету та після того, як менеджери компанії-друкарні розрахують вартість надання послуги, повідомляє директора, що потрібно оплатити замовлення.

Після того, як друкарня виконає свою роботу, менеджер забирає замовлення і перевіряє його на якість і кількість.

Після перевірки продукції менеджер може переходити до наступного кроку - Процес передачі замовлення клієнту (таблиця 2.5).

Таблиця 2.5

Паспорт бізнес-процесу Процес передачі замовлення клієнту

Параметр		Значення
Найменування процесу		Процес передачі замовлення клієнту
Результати		Виконане замовлення
Виходи БП	матеріальні	Акт виконаних робіт
	інформаційні	-
Входи БП	матеріальні	Готовий продукт
	інформаційні	-
Власник БП		Менеджер
Управлінські (нормативні) документи		-
Учасники	персонал	Менеджер
	засоби реалізації	телефон, соціальні мережі, месенджери, Інтернет
Споживач		-
Попередній процес		Процес виробництва
Наступний процес		Процес збору відгуків від клієнта

Джерело: Авторська розробка

Менеджер зв’язується з клієнтом, зателефонувавши, написавши в соціальних мережах чи месенджерах йому/їй, і повідомляє про готовність замовлення. Клієнт обирає спосіб отримання замовлення (таксі, пошта чи особисте отримання в офісі). У разі вибору клієнтом способу доставки таксі чи пошта, менеджер організовує доставку продукції.

Після отримання клієнтом замовлення менеджер робить запит клієнту на написання відгуку (таблиця 2.6). Цей процес є дуже важливим для розвитку компанії, оскільки саме клієнти помічають прогалини у сервісі чи продукції.

Таблиця 2.6

Паспорт бізнес-процесу Процес збору відгуків від клієнта

Параметр		Значення
Найменування процесу		Процес збору відгуків від клієнта
Результати		Заповнена форма відгуку
Виходи БП	матеріальні	-
	інформаційні	Заповнена форма відгуку
Входи БП	матеріальні	-
	інформаційні	Форма відгуку
Власник БП		Менеджер
Управлінські (нормативні) документи		-
Учасники	персонал	-
	засоби реалізації	телефон, соціальні мережі, месенджери, Інтернет, електронна пошта
Споживач		Директор

Продовження таблиці 2.6

Параметр	Значення
Попередній процес	Процес передачі замовлення клієнту
Наступний процес	-

Джерело: Авторська розробка

Менеджер надсилає клієнту лист з проханням заповнити форму відгуку. Усі відгуки переглядає директор і в разі виникнення негативного досвіду взаємодії з компанією, ініціює дії для покращення сервісу чи виготовлення продукції.

2.2. Побудова моделі бізнес-процесів поліграфічного підприємства

За допомогою паспортів бізнес-процесів, методологію BPMN та інструмента bpmn.io[23] можна побудувати моделі бізнес-процесів.

Нотація моделювання бізнес-процесів (BPMN) — це мова візуального моделювання для додатків бізнес-аналізу та визначення робочих процесів підприємства, яка є відкритим стандартним позначенням для графічних блок-схем, що використовується для визначення моделей бізнес-процесів. Це популярна та інтуїтивно зрозуміла графіка, яку можуть легко зрозуміти всі зацікавлені сторони бізнесу, включаючи бізнес-користувачів, бізнес-аналітиків, розробників програмного забезпечення та архітекторів даних.[24]

Створимо першу модель бізнес-процесу - Прийом замовлення у клієнта(рис 2.3). В моделі є 2 варіанта розвитку подій, створення дизайну чи наявність у замовника уже готового варіанту, тому було додано розгалуження.

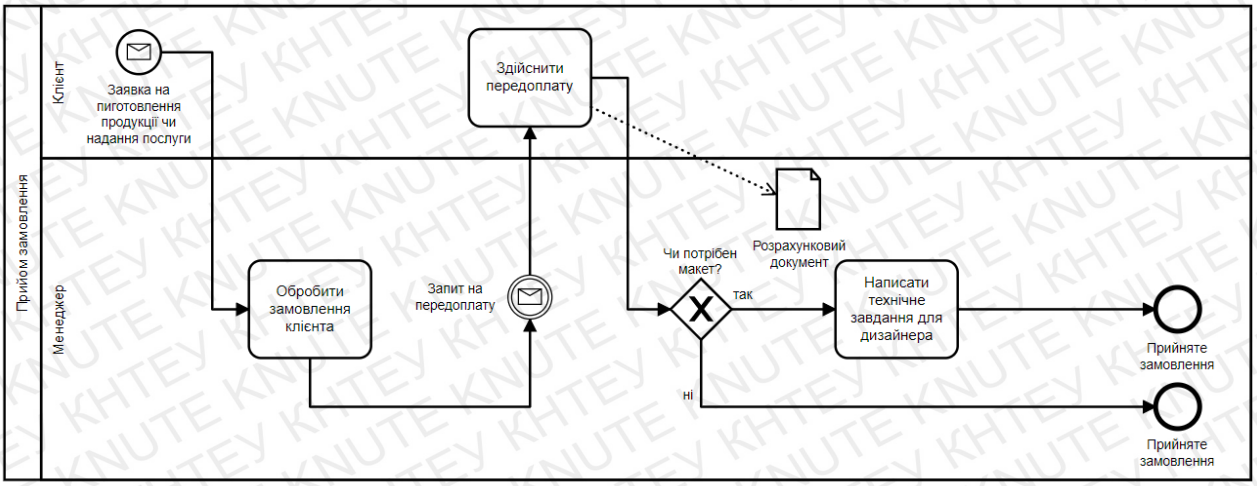

Рисунок 2.3. Модель бізнес-процесу Прийом замовлення у клієнта

Джерело: Авторська розробка

На другій моделі, моделі створення макету(рис. 2.4), також наявне розгалуження, адже дизайн підготовлений для клієнта може його не задовільнити і потрібно буде внести правки чи переробити дизайн.

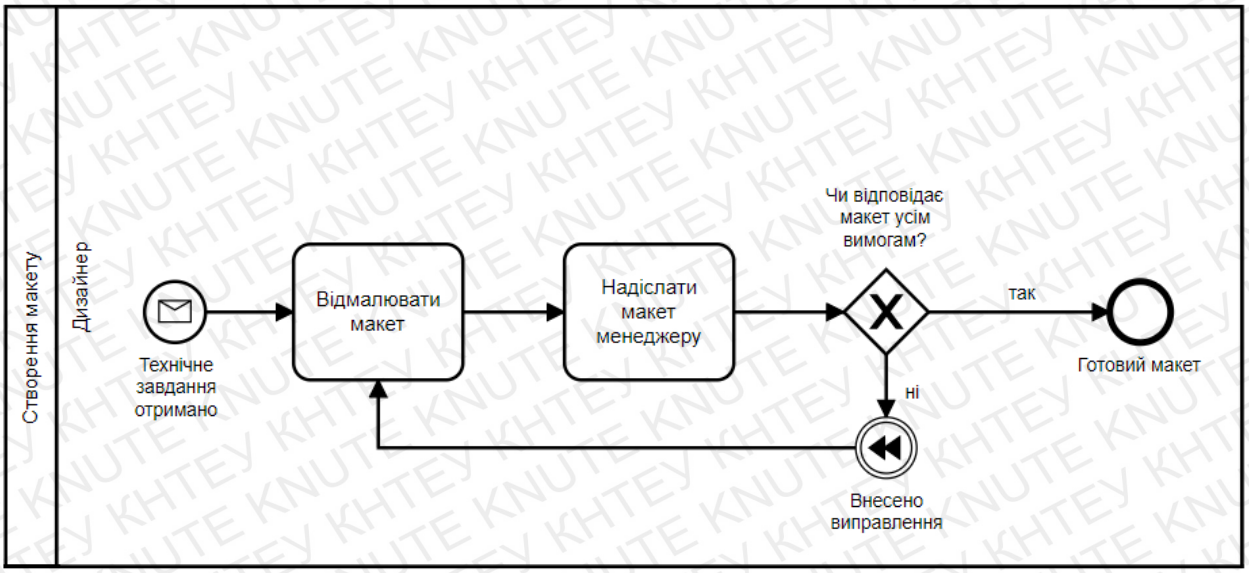

Рисунок 2.4. Модель бізнес-процесу Створення макету

Джерело: Авторська розробка

Модель бізнес-процесу Передача замовлення на друк підрядникам (рис 2.5) не має розгалуження, адже після заповнення заявки менеджером поліграфії на

сайті друкарні починаються бізнес-процеси друкарні, на які поліграфія не має жодного впливу.

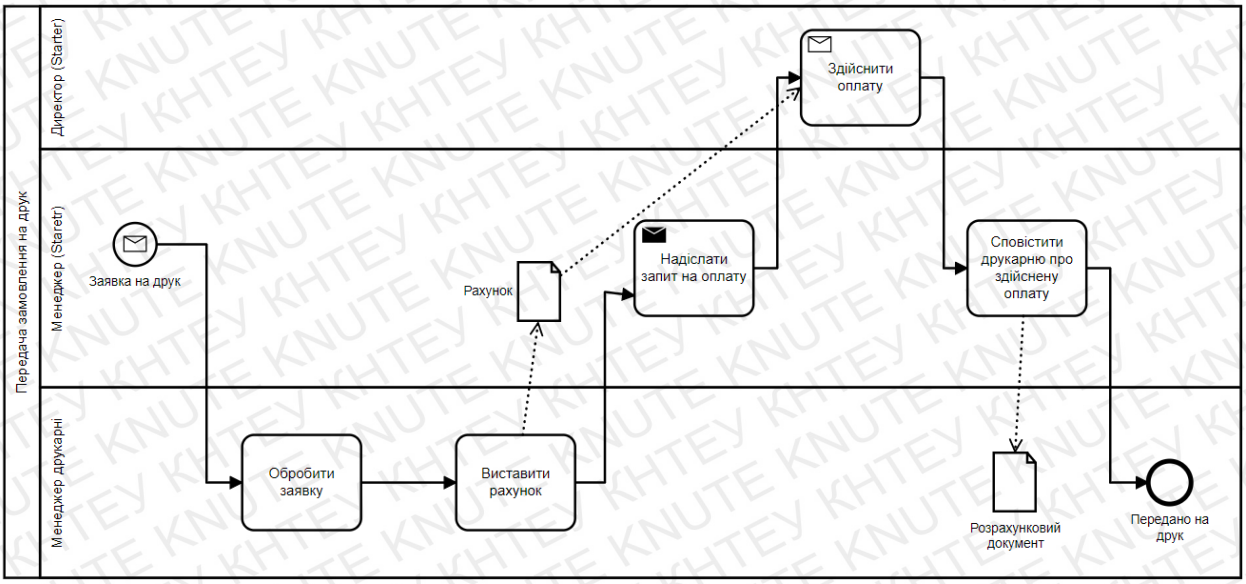

Рисунок 2.5. Модель бізнес-процесу Передача замовлення на друк підрядникам

Джерело: Авторська розробка

Модель бізнес-процесу передачі замовлення клієнту(рис 2.6) має розгалуження на етапі перевірки готової продукції менеджером поліграфії на предмет відповідності то характеристик та кількості в замовленні.

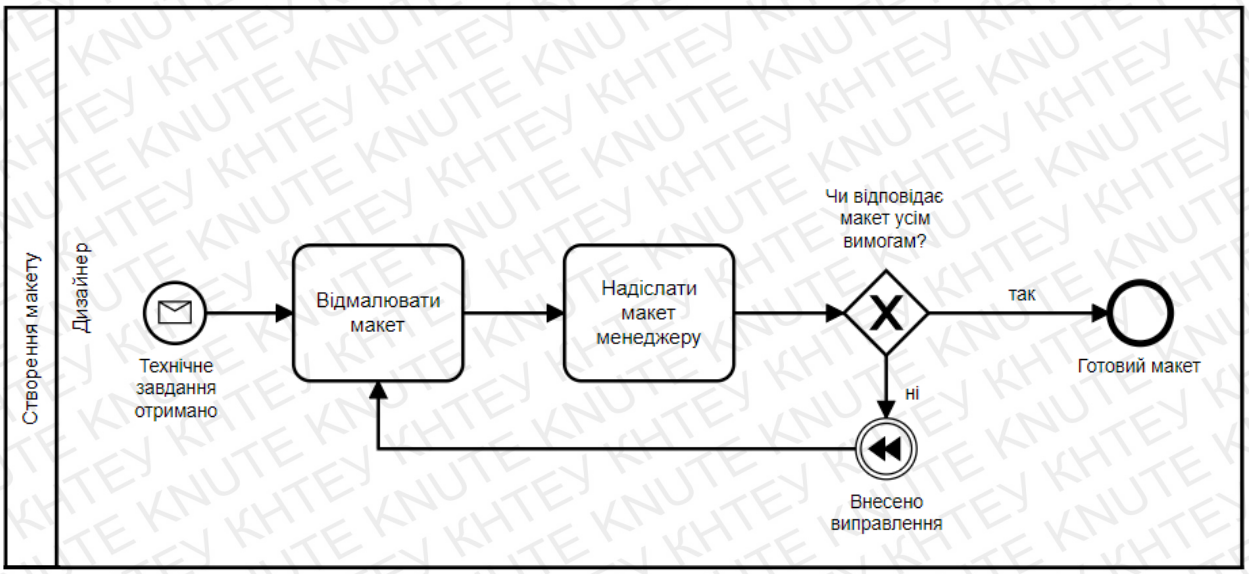

Рисунок 2.6. Модель бізнес-процесу Передача замовлення клієнту

Джерело: Авторська розробка

Також було виділено процес збору відгуків з отриманням форми відгуку від клієнта і внесення цих даних у веб-систему.

2.3. Співставлення бізнес-процесів з функціоналом веб-системи

Веб-система складається з чотирьох мікросервісів які забезпечують: безпеку, функціонал менеджерів, функціонал директора та функціонал дизайнерів. Розподіл функціоналу на окремі мікросервіси відбувався за функціоналом доступним для кожного відділу підприємства і окремим додатком який забезпечує безпеку інших. Кожен користувач має свою роль на підприємстві і може використовувати лише доступний для своєї ролі мікросервіс.

Відповідно до моделі бізнес-процесів мікросервіс для відділу менеджерів повинен реалізовувати такий функціонал:

- отримати замовлення за його id
- зберегти нове замовлення
- отримати всі замовлення
- додати новий продукт до замовлення за id замовлення
- оновити продукт за його id
- видалити продукт за його id
- отримати всі матеріали
- дозволяє додати новий матеріл до БД
- отримати всі фідбеки
- додати фідбек до замовлення за id замовлення
- отримати фідбек за його id
- видалити фідбек за його id
- оновити фідбек за його id

Відповідно до моделі бізнес-процесів мікросервіс для директора має такий функціонал:

- отримати всі замовлення

- отримати всі фідбеки
- отримати всі дизайни
- отримати фідбек за його id

Відповідно до моделі бізнес-процесів мікросервіс для відділу дизайнерів має такий функціонал:

- отримати всі продукти зі статусом «створення дизайну» та статусом дизайну «новий»
- отримати продукт за id
- оновити дизайн
- отримати всі продукти зі статусом «створення дизайну», статусом дизайну «створення» та id користувача
- отримати всі продукти зі статусом дизайну «готово» та id користувача

Висновки до розділу 2

Отже, на поліграфічному підприємстві присутні такі основні бізнес-процеси:

- Процес прийому замовлення;
- Процес створення макету;
- Процес передачі замовлення на друк підрядникам;
- Процес передачі замовлення клієнту;
- Процес збору відгуків від клієнтів.

Усі бізнес-процеси проходять через додатки у веб-системі, та відображаються в базі даних, за допомогою записів в базу даних та можливості їх перегляду через ендпойнти веб-системи ми можемо відслідковувати статуси товарів, замовлення та дизайнів. Кожен працівник бачить лише необхідну йому інформацію, повністю систематизовану та відображену в відсортованому вигляді та з застосуванням фільтрів.

Веб-система була поділена на 4 додатки, 3 з них використовуються для розподілу бізнес-процесів в організаційній структурі веб-системи і останній який забезпечує захист веб-системи, та дозволяє користувачам використовувати лише ті функції які передбачає їх роль на поліграфічному підприємстві.

РОЗДІЛ 3. СТВОРЕННЯ ВЕБ-СИСТЕМИ ПОЛІГРАФІЧНОГО ПІДПРИЄМСТВА НА ОСНОВІ МІКРОСЕРВІСІВ JAVA

3.1. Створення архітектури веб-системи

Для створення веб-системи поліграфічного підприємства було обрано мову програмування Java[25] та фреймворк Spring Boot[26]. Всі данні підприємства будуть зберігатись в MySQL[27] базі даних (рис 3.1), а зв'язок з базою буде здійснюватися за допомогою JPA[28] — Java Persistence API та Hibernate[29]. Також для зменшення обсягу коду було використано Lombok[30] який генерує більшу частину шаблонного коду.

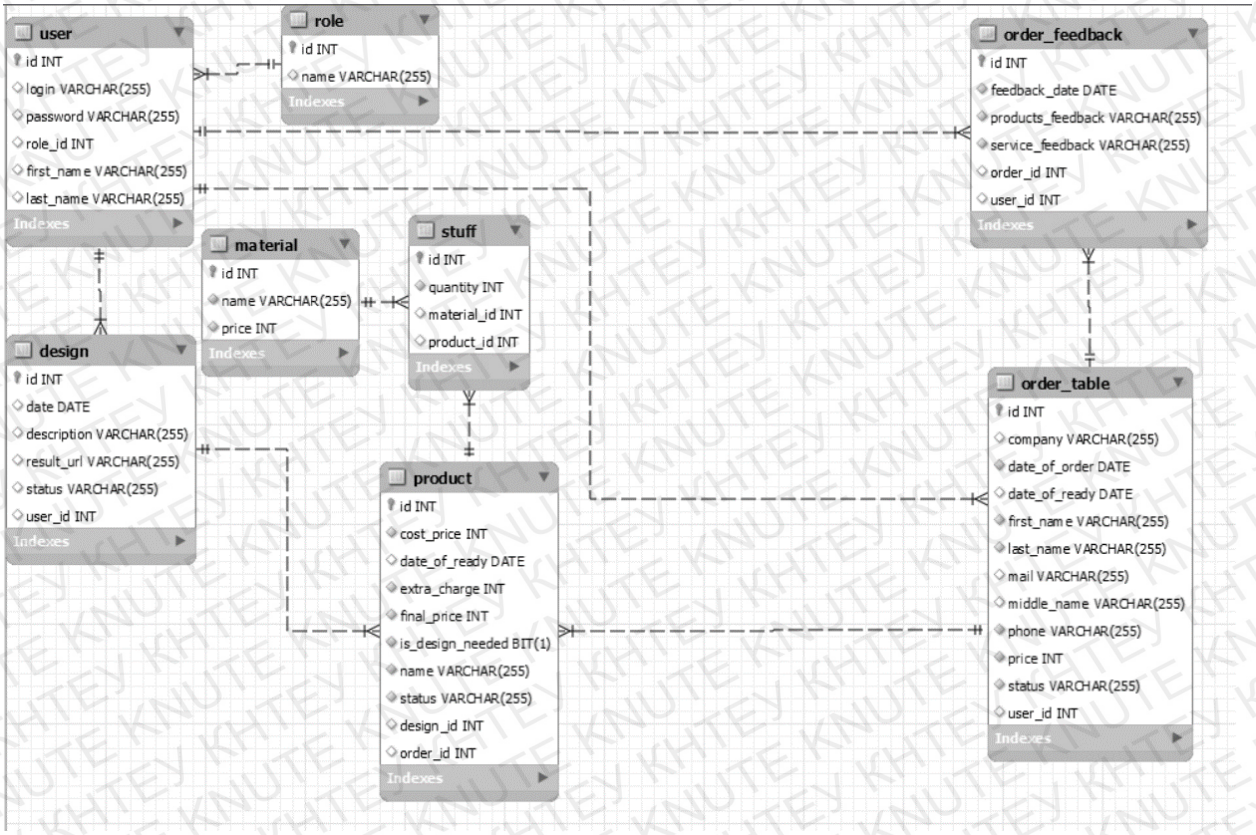

Рисунок 3.1 База даних поліграфічного підприємства

Джерело: Авторська розробка

Було здійснено розподіл веб-системи на 4 окремих RESTfull додатка:

- Polygraphy-manager (додаток доступний лише для працівників відділу менеджменту);

- Polygraphy-director (додаток доступний лише для директора підприємства);
- Polygraphy-designer (додаток доступний лише для працівників відділу дизайну);
- Polygraphy-jwt (додаток який забезпечує безпеку інших додатків, а саме генерує JSON Web Token для працівників різних відділів, що забезпечує їх доступ до інших додатків).

Поки використовується лише 4 веб-додатки і спільна база даних, але зі зростанням проекту можливе зручне додавання нових відділів, додатків та можливостей не змінюючи нічого з вже написаного коду. Також в майбутньому можливий розподіл БД на окремі частини, але при нинішніх вимогах до додатку і його можливостям спільна база є найкращим і найефективнішим рішенням.

Кожен мікросервіс в веб-системі побудований за допомогою MVC(Model–view–controller) Design Pattern. Також для кожного мікросервісу було створено однакові Entity (рис. 3.2) та їх реалізації(Додаток А-Ж) для зв’язку з кожною таблицею БД.

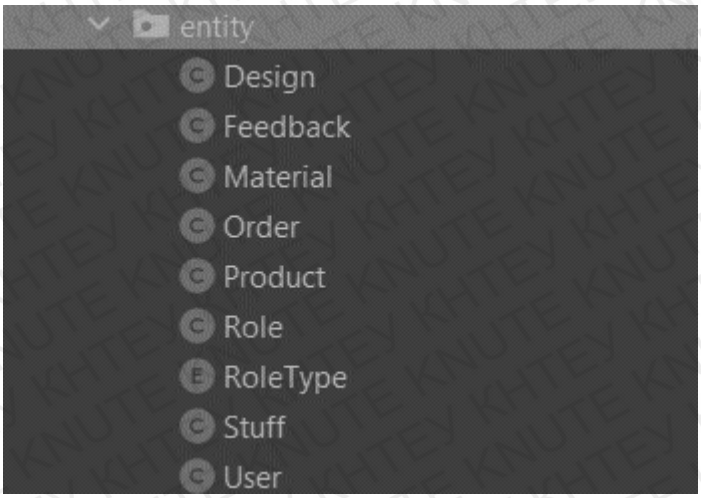

Рисунок 3.2 Entity для зв’язку з таблицями БД

Джерело: Авторська розробка

В кожному Entity вказуємо таблицю для звязку з бд за допомогою @Entity @Table(name=" "), і кожне поле звязуємо з полем в БД за допомогою @Column,

для дотримання чистого коду і правильного найменування колонок в БД використовуємо атрибут @Column(name = " "), а для створення звязку мфж таблицями будемо використовувати @ManyToOne, @OneToMany, @ManyToMany і @OneToOne аннотації. Особливої уваги заслуговує звязка Bidirectional @OneToMany + @ManyToOne, вона дозволяє бачити дочірній елемент в контексті батьківського і батьківський елемент в контексті дочірнього елементу, хоча такий підхід може зациклити запити до БД, тому доповнимо таку взаємодію додатковими анотаціями @JsonManagedReference та @JsonBackReference, які допоможуть уникнути зациклювання та оптимізувати запити.

Також в кожному мікросервісі будуть присутні класи необхідні для створення запитів до БД за допомогою JPA(рис 3.3)

```
@Repository
public interface DesignRepository extends JpaRepository<Design, Integer> {
}
```

Рисунок 3.3 Приклад JPA Repository

Джерело: Авторська розробка

Spring Boot разом з JPA створюють прості CRUD методи(рис 3.4) самостійно, а для більш вузьких методів необхідно прописати назву методу за спеціальним патерном і передати правильну комбінацію параметрів(рис 3.5).

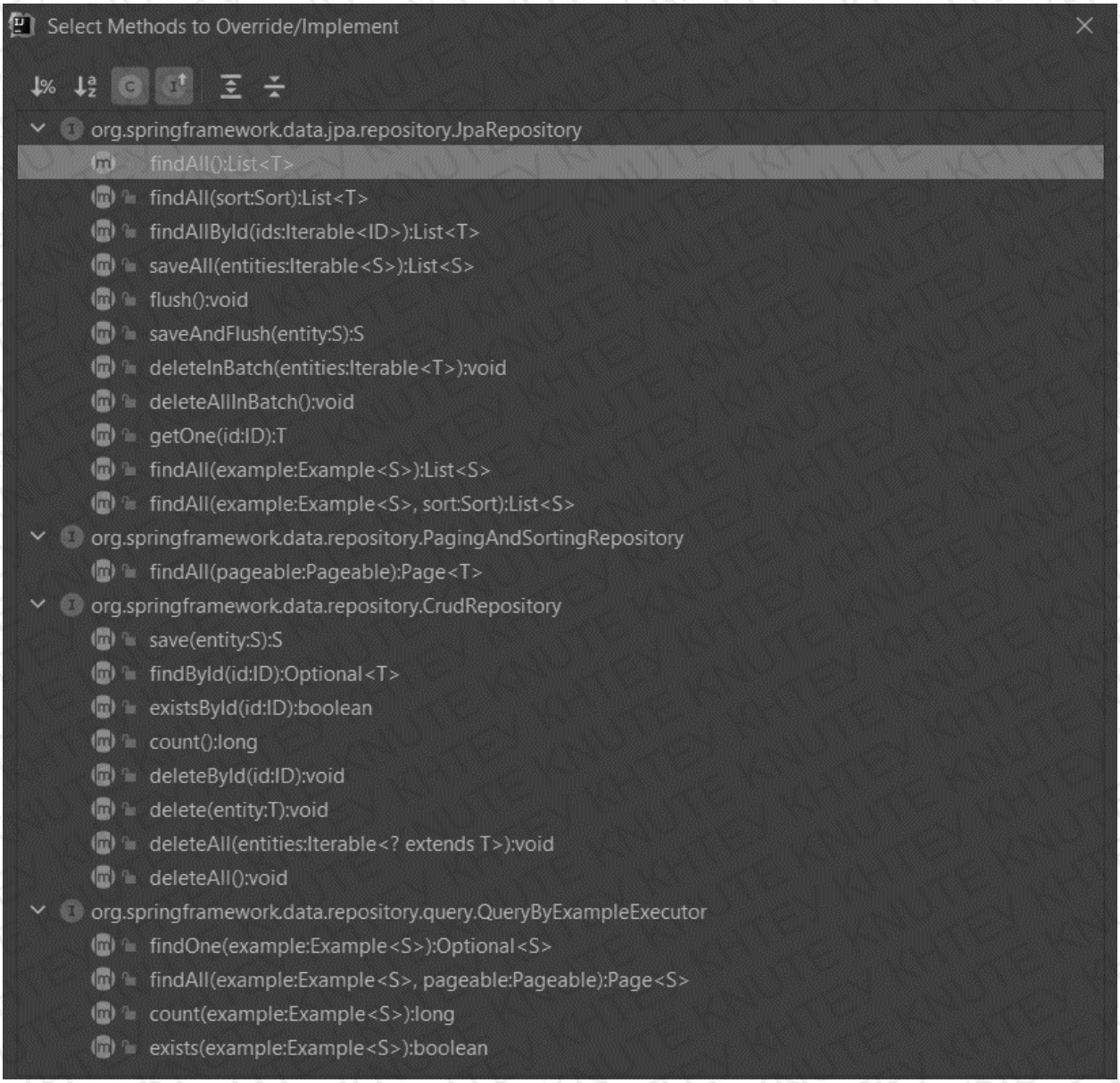

Рисунок 3.4 Згенеровані методи JPA Repository

Джерело: Авторська розробка

```
@Repository
public interface FeedbackRepository extends JpaRepository<Feedback, Integer> {
    boolean deleteById(int id);
    Page<Feedback> findAllByFeedbackDate(PageRequest page, String filterValue);
}
```

Рисунок 3.5 Власний метод для взаємодії з базою даних через інтерфейс JPA Repository

Джерело: Авторська розробка

Також в кожному мікросервісі використовуються конфігураційні файли (рис. 3.6) для захисту доступу, котрі контролюють права доступу користувачів для кожного ендпойнту та кожного мікросервісу. Доступ контролюється за допомогою jwt котрі передаються з фронт енду з кожним запитом до серверу, а отримати даний токен для кожного користувача можна при авторизації на сайті. Кожен токен має інформацію про логін користувачу, для якого даний токен було згенеровано та термін придатності токену, який складає 1 день, проте цей параметр можна змінити при необхідності. Після отримання токену з запитом сервре перевіряє його термін придатності і якщо токен ще дійсний то дістає роль користувача з БД, щоб перевірити чи має користувач доступ до ресурсу за запитом.

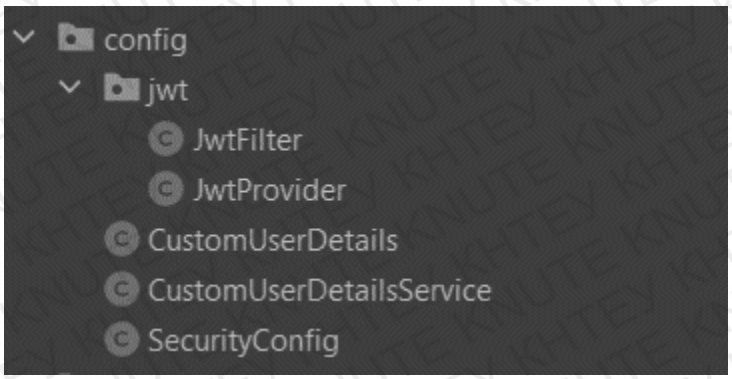

Рисунок 3.6 Security config

Джерело: Авторська розробка

Для безпечної взаємодії користувача з БД спілкування проходить через ендпойнти контроллера, в кожного мікросервісу є свій контроллер який забезпечує доступ до певних сервісів, а вже сервіси звертаються до БД за допомогою репозиторіїв. Також є допоміжні класи які використовуються для більш зручного створення запитів чи відображення результату для фронт енд розробників.

3.2 Розробка мікросервісів веб-системи

Отже, при розробці архітектури веб-систему було розділено на 4 окремих мікросервіси, і в кожному мікросервісі пристуні свої незалежні сервіси та

контроллери, тому при зміні одного мікросервісу(окрім зміни таблиць в БД) інші мікросервіси продовжуть працювати в звичайному режимі.

Перший створений мікросервіс буде відповідати за безпеку веб-системи, в даному мікросервісі буде використано технологію jwt, що дозволить використовувати одну форму логіну та реєстрації для всіх інших веб-систем, уникнути дублювання коду і забезпечити спокійних перехід від одного додатку до іншого без повторної авторизації.

Даний мікросервіс має всього 3 ендпойнти:

- POST api/security/ register (дозволяє зареєструвати користувача)
- POST api/security/auth (дозволяє авторизувати користувача)
- GET api/security/user/{id} (дозволяє отримати дані користувача за його id)

Доступ до даного мікросервісу має кожен користувач, незалежно від ролі.

Другий створений мікросервіс буде відповідати за функціонал доступний менеджеру на підприємстві і так як цей функціонал не є необхідним для дизайнера чи деректора, то доступ до мікросервісу мають лише користувачи з роллю “manager”.

В середині мікросервісу маємо такі ендпойнти:

- GET api/manager/order/{id}(дозволяє отримати замовлення за його id)
- POST api/manager/order(дозволяє зберегти нове замовлення)
- GET api/manager/orders(дозволяє отримати всі замовлення)
- POST api/manager/order/{id}/product(дозволяє додати новий продукт до замовлення за id замовлення)
- PUT api/manager/product/{id}(дозволяє оновити продукт за його id)
- DELETE api/manager/product/{id}(дозволяє видалити продукт за його id)
- GET api/manager/materials(дозволяє отримати всі матеріали)

- POST api/manager/material(дозволяєдодати новий матеріл до БД)
- GET api/manager/feedbacks(дозволяє отримати всі фідбеки)
- POST api/manager/order/{id}/feedback(дозволяє додати фідбек до замовлення за id замовлення)
- GET api/manager/feedback/{id}(дозволяє отримати фідбек за його id)
- DELETE api/manager/feedback/{id}(дозволяє видалити фідбек за його id)
- PUT api/manager/feedback/{id}(дозволяє оновити фідбек за його id)

Третій мікросервіс відповідає за функціонал доступний директору підприємства і доступ до даного мікросервісу мають лише користувачи з роллю “director”

В середині мікросервісу маємо такі ендпойнти:

- GET api/director/orders(дозволяє отримати всі замовлення)
- GET api/director /feedbacks(дозволяє отримати всі фідбеки)
- GET api/director /designs(дозволяє отримати всі дизайни)
- GET api/director /feedback/{id}(дозволяє отримати фідбек за його id)

Четвертий мікросервіс відповідає за функціонал доступний дизайнерам підприємства і доступ до даного мікросервісу мають лише користувачи з роллю “designer”

В середині мікросервісу маємо такі ендпойнти:

- GET api/designer/products(дозволяє отримати всі продукти зі статусом «створення дизайну» та статусом дизайну «новий»)
- POST api/designer/product/{id}(дозволяє отримати продукт за id)
- PUT api/designer/design/{id}(дозволяє оновити дизайн)
- GET api/designer/products/inprogress/{userId}(дозволяє отримати всі продукти зі статусом «створення дизайну», статусом дизайну «створення» та id користувача)

- GET api/designer/products/done/{userId}(дозволяє отримати всі продукти зі статусом дизайну «готово» та id користувача)

3.3 Зв’язок з Front-end веб-системи

Оскільки всі мікросервіси є RESTfull додатками то зв'язок з front-End здійснюється за допомогою ендпойнтів вказаних в контроллері, такий зв'язок дозволяє front-End розробнику вирішувати як саме відображати дані, який використовувати фреймворк та хост, все що портібно щоб отримати дані для відображення це звернутись за певним урлом та передати серверу необхідіні дані і отримати результат в JSON форматі. Також такий зв'язок дозволяє використовувати веб-систему одразу багатьма сайтами при необхідності, а також передбачає можливість створеня нового дизайну зі збреженням можливості використання старого, що значно спрощуе супровід та використання веб-системи.

Для більш зручного звязку з front-End окрім REST контроллерів також була створена велика кількість Data Transfer Object (DTO), це дозволяє при зверненні до серверу передавати не весь масив даних необхідних для збереження,оновлення чи зчитування даних в БД, а лише необхідний мінімум, інші дані вже додаються на стороні серверу. Також це дозволяє передавати на front-End не весь масив даних, а тільки ту частину яка потрібна для відображення на конкретній сторінці.

Ендпойнти в REST контроллерах приймають 3 типи параметрів :

- @RequestParam
- @PathVariable
- @RequestBody

@RequestParam використовується для того зоб задати парметри пагінації, сортування чи фільтрування. @PathVariable використовується щоб визначити id

з запиту користувача. @RequestBody використовується для передачі об’єктів до ендпойнтів для збереження чи оновлення даних в БД.

Пагінація та сортування виконується за допомогою можливостей JPA і потребує лише створення PageRequest і передачу його до JPA репозиторія. JPA також підтримує просту фільтрацію, де метод для створення запиту до БД створюється спеціально для певної фільтрації, але такий варінт не підтриує динамічну фільтрацію за замовчуванням. Для реалізації динамічної фільтрації було створено Custom Generic Specification(рис 3.7) і звичайний Search Criteria(рис 3.8) який включає в себе назву поля фільтрації, метод фільтрації(дорівнює,більше чи менше) і значеняя поля для фільтрації.

```
@Component
@AllArgsConstructor
@NoArgsConstructor
public class CustomSpecification<T> implements Specification<T> {

    private SearchCriteria criteria;

    @Override
    public Predicate toPredicate
            (Root<T> root, CriteriaQuery<?> query, CriteriaBuilder builder) {

        if (criteria.getOperation().equalsIgnoreCase( anotherString: ">")) {
            return builder.greaterThanOrEqualTo(
                    root.get(criteria.getKey()), criteria.getValue().toString());
        }
        else if (criteria.getOperation().equalsIgnoreCase( anotherString: "<")) {
            return builder.lessThanOrEqualTo(
                    root.get(criteria.getKey()), criteria.getValue().toString());
        }
        else if (criteria.getOperation().equalsIgnoreCase( anotherString: ":")) {
            if (root.get(criteria.getKey()).getJavaType() == String.class) {
                return builder.like(
                        root.get(criteria.getKey()),  s: "%" + criteria.getValue() + "%");
            } else {
                return builder.equal(root.get(criteria.getKey()), criteria.getValue());
            }
        }
        return null;
    }
}
```

Рисунок 3.7 Custom Generic Specification

Джерело: Авторська розробка

```java
@Getter
@Setter

@AllArgsConstructor
public class SearchCriteria {
    private String key;
    private String operation;
    private Object value;
}
```

Рисунок 3.8 Search Criteria

Джерело: Авторська розробка

Тепер можна створювати специфікацію для фільтрації для будь-якого об’єкту і використовувати її для здійснення запиту до БД, але для кожної додаткової фільтрації потрібно створювати нову специфікацію і в ситуації коли кількість одночасних фільтрацій не відома буде зручно використовувати Builder(рис 3.9) і додавати кожну специфікацію до білдеру, щоб застосувати все разом

```java
@Component
public class SpecificationBuilder<T> {

    private final ArrayList<SearchCriteria> params;

    public SpecificationBuilder() { params = new ArrayList<>(); }

    public SpecificationBuilder with(String key, String operation, Object value) {
        params.add(new SearchCriteria(key, operation, value));
        return this;
    }

    public Specification<T> build() {
        if (params.size() == 0) {
            return null;
        }

        List<Specification> specs = params.stream()
                .map(CustomSpecification<T>::new)
                .collect(Collectors.toList());

        Specification result = specs.get(0);

        for (int i = 1; i < params.size(); i++) {
            result = Specification.where(result)
                    .and(specs.get(i));
        }
        return result;
    }
}
```

Рисунок 3.9 Builder

Джерело: Авторська розробка

При використанні білдеру тепер достатньо писати фільтрацію в запиті за патерном "(\\w+?)(:|-<|>)(\\w+?)," і відправляти одночасно будь-яку кількість парметрів фільрації. Якщо ж фільтрація не потрібна то можна отримати всі результати без вказування параметру “?filter=” в запиті.

Для використання ендпойнтів користувачо необхідно відправити первний перелік параметрів до кожного з методів і отримати результат від додатків : Polygraphy-manager(додаток З), Polygraphy-director(додаток И), Polygraphy-designer(додаток І), Polygraphy-jwt(додаток Й)

Окрім того в методах з використанням пагінації (параметри Page та size) разом з відповіддю зазначеною в таблицю користувач отримає дані про пагінацію результату(рис 3.18) з зазначенням сортування, фільтрації, наявності даних(!= empty), загальної кількості значень в БД, кількості сторінок, кількість елементів на сторінці, номер отриманої сторінки, а також чи є дана сторінка першою чи останньою.


```
"pageable": {
    "sort": {
        "empty": true,
        "unsorted": true,
        "sorted": false
    },
    "offset": 3,
    "pageSize": 3,
    "pageNumber": 1,
    "unpaged": false,
    "paged": true
},
"totalElements": 8,
"totalPages": 3,
"last": false,
"size": 3,
"number": 1,
"first": false,
"sort": {
    "empty": true,
    "unsorted": true,
    "sorted": false
},
"numberOfElements": 3,
"empty": false
```


Рисунок 3.18 Приклад Pagination

Джерело: Авторська розробка

Висновки до розділу 3

Отже, для створення веб-системи була обрана архітектура мікросервісів яку склали 4 RESTfull додатка (Polygraphy-manager, Polygraphy-director, Polygraphy-designer, Polygraphy-jwt) та спільна БД. Кожен додаток створений у вигляді API з контролерами та ендпойнтами для зв’язку з Front-end. В додатку забезпечена авторизація та аутентифікація користувачів за допомогою jwt з розмежуванням прав доступу для кожного виду співробітників поліграфії.

Веб-система має значний потенціал для розширення чи зміни функціоналу, за допомогою відключення мікросервісів, підключення нових та зміни вже готових. Також є можливість запуску декількох екземплярів кожного додатку при збільшенні потоку користувачів системи і роздробленні БД на окремі бази при розширенні підприємства.

ВИСНОВКИ

Отже, з розвитком мережі та збільшенням конкуренції на ринку інформаційних технологій вимоги до сайтів та веб-додатків стають все більш значними. Компанії намагаються відрізнятись одна від одної і здивувати користувача новими можливостями чи дизайнами. Можливості сайтів та веб-додатків зростають щодня, замовники бажають автоматизувати якомога більшу частину роботи, а користувачі бажають скоротити кількість необхідних дій для досягнення результату при використанні веб-системи. Однак всі компанії унікальні, мають різні розміри, можливості та потреби. Через такі відмінності ринок не може запропонувати єдиного універсального дизайну чи архітектури і кожній компанії потрібно обирати оптимальний варіант.

В першому розділі було досліджено підходи до створення веб-додатків, їх види та вже готові додатки, які використовують поліграфічні підприємства. Готові додатки допоможуть підприємствам вести облік та спростять багато труднощів при ведені підприємства, але більшої автоматизації можна досягти лише створенням власної системи для кожного підприємства, яка буде повністю відповідати бізнес-процесам підприємства та його цілям. Також було досліжено архітектурні підходи до створення веб-систем, монолітний та мікросервісний і визначено їх переваги та недоліки.

Для невеликого поліграфічного підприємства на початку роботи ідеально підходить монолітна архітектура, адже невелика кількість процесів та працівників не потребує явного розмежування. Проте при розширенні розмірів чи можливостей компанії зміна, вдосконалення чи додавання нового коду може викликати труднощі, особливо це стосується компаній без власного IT-відділу, які замовляють продукт і в подальшому лише використовують його. В такому випадку доцільно буде розглянути можливість використання архітектури мікросервісів яка дозволить компанії замовляти при потребі нові додатки і підключати їх до вже готової системи. З такою архітектурою новій команді

розробників не доведеться розбиратись зі старим кодом, вивчати його структуру та намагатись вносити зміни не пошкодивши інший функціонал, достатньо написати новий незалежний додаток і використовувати його разом з вже готовими.

Для створення веб-системи для підприємства Starter в другому розділі було виділено основні бізнес-процеси(процес прийому замовлення, процес створення макету, процес виробництва, процес передачі замовлення клієнту, процес збору відгуків від клієнта), побудовано їх паспорти та моделі, а також було визначено організаційну структуру підприємства.

Для наглядності схеми бізнес-процесів були відображені в нотації BPMN за допомогою bpmn.io.

В третьому розділі була створення веб-система поліграфічного підприємства Starter. Для її створення була обрана архітектура мікросервісів яку склали 4 RESTfull додатка (Polygraphy-manager, Polygraphy-director, Polygraphy-designer, Polygraphy-jwt) та спільна БД. Розподіл веб-системи на окремі додатки відбувався за ролями персоналу та бізнес-процесами. Ролі в додатку були визначені виходячи з організаційної структури підприємства: директор, дизайнери та менеджери.

Кожен додаток створений у вигляді API з контролерами та ендпойнтами для зв’язку з Front-end. В додатку забезпечена авторизація та аутентифікація користувачів за допомогою jwt з розмежуванням прав доступу для кожного виду співробітників поліграфії.

Додатки для веб системи створювались за допомогою таких технологій:

- Spring + Spring Boot
- Java 8
- MySQL
- Java Persistence API та Hibernate
- JSON Web Token

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Герасимов В. В. Пономарьов І. В., Ліщенко О. О. Аналіз основних технологій розробки веб-додатків на платформі Java. *Системні технології*. 2015. Вип. 1. С. 103-108. URL: http://nbuv.gov.ua/UJRN/st_2015_1_16 (дата звернення: 11.02.2021)

2. Маліцький О.С. Порівняльний аналіз сучасних засобів для розробки систем веб-застосунків. *Інститут докторантури та аспірантури:* веб-сайт. URL: http://inmad.vntu.edu.ua/portal/static/00156A66-D5B6-4671-926B-7AF5BFFA2345.pdf (дата звернення: 11.02.2021)

3. Бурлаков А.А., Балишин О.О. Вибір сучасних веб-орієнтованих технологій програмування під час розробки програмного забезпечення на java-платформі. *Технічні науки*. 2016. № 3. URL: http://elar.khnu.km.ua/jspui/bitstream/123456789/4841/1/%D0%91%D0%90%D0%9B%D0%98%D0%A8%D0%98%D0%9D.pdf (дата звернення: 11.02.2021)

4. Joshua Bloch. Effective Java. Boston: Addison-Wesley Professional, 2018. P. 392. URL: https://www.amazon.de/dp/0134685997 (дата звернення: 17.02.2021)

5. Cay S. Horstmann. Core Java: book. New Jersey : Prentice Hall, 2018. P. 1008. URL: https://www.amazon.de/dp/0135166306 (дата звернення: 17.02.2021)

6. Bruce Eckel. Thinking in Java. New Jersey : Prentice Hall, 2006. P. 1150. URL: https://www.amazon.de/Thinking-Java-introduction-object-oriented-programming/dp/0131872486/ref=sr_1_3 (дата звернення: 17.02.2021)

7. Щедріна О.І., Агутін М.М. Інтеренет технології в бізнесі: навч. посібник. Київ : КНЕУ, 2012. 304 с. URL: https://ktpu.kpi.ua/wp-content/uploads/2014/02/SHHedrino-O.I.-Agutin-M.M.-Internet-tehnologiyi-v-biznesi.pdf (дата звернення: 03.03.2021)

8. Website. *Techopedia*: веб-сайт. URL: https://www.techopedia.com/definition/5411/website (дата звернення: 03.03.2021)

9. Web application (web app). *Search software quality*: веб-сайт. URL: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app (дата звернення: 03.03.2021)

10. What is the difference between a mobile app and a web app? *Careerfoundry*: веб-сайт. URL: https://careerfoundry.com/en/blog/web-development/what-is-the-difference-between-a-mobile-app-and-a-web-app/ (дата звернення: 10.03.2021)

11. Web Application. *Ring central*: веб-сайт. URL: https://www.ringcentral.co.uk/gb/en/blog/definitions/web-application/ (дата звернення: 10.03.2021)

12. Introduction into Microservices. *Specify*: веб-сайт. URL: https://specify.io/concepts/microservices (дата звернення: 03.03.2021)

13. Microservices. *Microservices*: веб-сайт. URL: https://microservices.io/ (дата звернення: 12.03.2021)

14. Application Architecture Guide. *Microsoft*: веб-сайт. URL: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices (дата звернення: 13.03.2021)

15. Pattern: Microservice Architecture. *Microservices* : веб-сайт. URL: https://microservices.io/patterns/microservices.html (дата звернення: 24.03.2021)

16. Micro services, what even are they? *Rea group*: веб-сайт. URL: https://www.rea-group.com/blog/micro-services-what-even-are-they/ (дата звернення: 24.03.2021)

17. Small Teams Are Dramatically More Efficient than Large Teams. *Spin atomic object*: веб-сайт. URL: https://spin.atomicobject.com/2012/01/11/small-teams-are-dramatically-more-efficient-than-large-teams/ (дата звернення: 28.03.2021)

18. Printeffect. *Printeffect*: веб-сайт. URL: https://printeffect.ru/ (дата звернення: 28.03.2021)

19. 1С:Поліграфия. *1С*: веб-сайт. URL: https://1cpoly.ru/poligrafiy2/ (дата звернення: 28.03.2021)

20. Дубініна В.В. Теоретичні аспекти класифікації бізнес-процесів підприємства. *Економічні науки*. 2014. № 7. URL: http://www.ej.kherson.ua/journal/economic_07/104.pdf (дата звернення: 15.04.2021)

21. Демиденко В. В. Управління бізнес-процесами як складова процесного підходу до управління підприємством. *Ефективна екеноміка.* 2015. № 11. URL: http://www.economy.nayka.com.ua/pdf/11_2015/44.pdf (дата звернення: 15.04.2021)

22. Драган О. І. Підходи до формування системи управління бізнес-процесами на підприємстві. *Ефективна економіка*. 2019. № 2. URL: http://www.economy.nayka.com.ua/pdf/2_2019/17.pdf (дата звернення: 15.04.2021)

23. Web-based tooling for BPMN, DMN and Forms. *BPMN*: веб-сайт. URL: https://bpmn.io/ (дата звернення: 20.05.2021)

24. What is BPMN? *Visual paradigm*: веб-сайт. URL: https://www.visual-paradigm.com/guide/bpmn/what-is-bpmn/ (дата звернення: 20.05.2021)

25. Java. *Java*: веб-сайт. URL: www.java.com (дата звернення: 17.09.2021)

26. Spring Boot. *Spring*: веб-сайт. URL: https://spring.io/projects/spring-boot (дата звернення: 17.09.2021)

27. MySQL. *MySQL*: веб-сайт. URL: https://www.mysql.com/ (дата звернення: 17.09.2021)

28. Spring Data. *Spring*: веб-сайт. URL: https://spring.io/projects/spring-data-jpa (дата звернення: 17.09.2021)

29. Hibernate. *Hibernate*: веб-сайт. URL: https://hibernate.org/ (дата звернення: 17.09.2021)

30. Lombok. *Project Lombok*: веб-сайт. URL: https://projectlombok.org/ (дата звернення: 17.09.2021)

ДОДАТКИ

Додаток А

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table( name="design",
        uniqueConstraints={@UniqueConstraint(columnNames={"id"})})
public class Design {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    @ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JsonManagedReference
    @JoinColumn(name = "user_id")
    private User user;
    @Column(name = "description")
    private String description;
    @Column(name = "result_url")
    private String resultUrl;
    @Column(name = "status")
    private String status;
    @Column(name = "date")
    private LocalDate dateOfReady;
}
```

Рис А.1. Entity Design

Джерело: Авторська розробка

Додаток Б

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor

@Table(name = "order_feedback",
        uniqueConstraints={@UniqueConstraint(columnNames={"id"})})
@Entity
public class Feedback {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    @ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JsonManagedReference
    @JoinColumn(name = "user_id")
    private User user;
    @Column(name = "service_feedback", nullable = false)
    private String serviceFeedback;
    @Column(name = "products_feedback", nullable = false)
    private String productsFeedback;
    @Column(name = "feedback_date", nullable = false)
    private LocalDate feedbackDate;
    @OneToOne(fetch = FetchType.EAGER)
    @JsonBackReference
    @JoinColumn(name = "order_id", updatable = false, insertable = false)
    private Order order;
}
```

Рис Б.1. Entity Feedback

Джерело: Авторська розробка

Додаток В

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table( name="material",
        uniqueConstraints={@UniqueConstraint(columnNames={"id"})})
public class Material {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    @Column(name = "name", nullable = false)
    private String name;
    @Column(name = "price", nullable = false)
    private int price;
}
```

Рис В.1. Entity Material

Джерело: Авторська розробка

Додаток Г

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table( name="order_table",
        uniqueConstraints={@UniqueConstraint(columnNames={"id"})})
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    @ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JsonManagedReference
    @JoinColumn(name = "user_id")
    private User user;
    @Column(name = "first_name", nullable = false)
    private String firstName;
    @Column(name = "last_name", nullable = false)
    private String lastName;
    @Column(name = "middle_name")
    private String middleName;
    @Column(name = "phone", nullable = false)
    private String phone;
    @Column(name = "mail")
    private String mail;
    @Column(name = "company")
    private String company;
    @Column(name = "price", nullable = false)
    private int price;
    @Column(name = "date_of_order", nullable = false)
    private LocalDate dateOfOrder;
    @Column(name = "date_of_ready")
    private LocalDate dateOfReady;
    @Column(name = "status", nullable = false)
    private String status;
    @OneToMany(cascade=CascadeType.ALL,fetch = FetchType.EAGER)
    @JsonBackReference
    @JoinColumn(name = "order_id")
    private List<Product> products = new ArrayList<>();
}
```

Рис Г.1. Entity Order

Джерело: Авторська розробка

Додаток Д

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table( name="product",
        uniqueConstraints={@UniqueConstraint(columnNames={"id"})})
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    @Column(name = "name", nullable = false)
    private String name;
    @OneToMany(cascade=CascadeType.ALL, fetch = FetchType.EAGER)
    @JsonBackReference
    @JoinColumn(name = "product_id")
    private List<Stuff> stuff = new ArrayList<>();
    @Column(name = "cost_price", nullable = false)
    private int costPrice;
    @Column(name = "extra_charge", nullable = false)
    private int extraCharge;
    @Column(name = "final_price", nullable = false)
    private int finalPrice;
    @Column(name = "status", nullable = false)
    private String status;
    @Column(name = "is_design_needed", nullable = false)
    private boolean isDesignNeeded;
    @Column(name = "date_of_ready")
    private LocalDate dateOfReady;
    @OneToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinColumn(name = "design_id")
    private Design design;
    @ManyToOne(fetch = FetchType.EAGER)
    @JsonManagedReference
    @JoinColumn(name = "order_id", insertable = false, updatable = false)
    private Order order;
}
```

Рис Д.1. Entity Product

Джерело: Авторська розробка

Додаток Е

```java
@Entity
@Table(name = "role")
@Data
public class Role {

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

    @Column
    private String name;
}
```

Рис Е.1. Entity Role

Джерело: Авторська розробка

Додаток Є

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table( name="stuff",
        uniqueConstraints={@UniqueConstraint(columnNames={"id"})})
public class Stuff {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    @Column(name = "quantity", nullable = false)
    private int quantity;
    @ManyToOne(fetch = FetchType.EAGER)
    @JsonManagedReference
    @JoinColumn(name = "product_id", insertable = false, updatable = false)
    private Product product;
    @ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JsonManagedReference
    @JoinColumn(name = "material_id")
    private Material material;
}
```

Рис Є.1. Entity Stuff

Джерело: Авторська розробка

Додаток Ж

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table( name="user",
        uniqueConstraints={@UniqueConstraint(columnNames={"login"})})
public class User {


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

    @Column
    private String login;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column
    private String password;

    @ManyToOne
    @JoinColumn(name = "role_id")
    private Role roleEntity;
}
```

Рис Ж.1. Entity User

Джерело: Авторська розробка

Додаток З

Вхідні та вихідні параметри додатку Polygraphy-manager

getOrder	
id	{ "id": "int", "firstName": "string", "lastName": "string", "middleName": "string", "phone": "string", "mail": "string", "company": "string", "price": "int", "status": "string", "dateOfOrder": "date", "products": [{ "id": "int", "name": "string", "materials": [{ "quantity": "int", "material": { "id": "int", "name": "string", "price": "int" } }], "costPrice": "int", "extraCharge": "int", "finalPrice": "int", "status": "string", "isDesignNeeded": "boolean" }

	] }
saveOrder	

{
"order": {
"firstName": "string",
"lastName": "string",
"middleName": "string",
"phone": "string",
"mail": "string",
"company": "string",
"price": "int",
"status": "string"
},
"products": [
{
"name": "string",
"materials": [
{
"quantity": "int",
"materialId": "int"
}
],
"costPrice": "int",
"extraCharge": "int",
"status": "string",
"isDesignNeeded": "boolean",
"design": {
"description": "string",
"resultUrl": "string",
"status": "string",
"dateOfReady": "date"
}
}
]
	{
	"id": "int",
	"firstName": "string",
	"lastName": "string",
	"middleName": "string",
	"phone": "string",
	"mail": "string",
	"company": "string",
	"price": "int",
	"status": "string",
	"dateOfOrder": "date",
	"products": [
	{
	"id": "int",
	"name": "string",
	"materials": [
	{
	"quantity": "int",
	"material": {
	"id": "int",
	"name": "string",
	"price": "int"
	}
	}
	],
	"costPrice": "int",
	"extraCharge": "int",
	"finalPrice": "int",
	"status": "string",
	"isDesignNeeded": "boolean"
	}
	]

}	}
getAllOrders	

page,
size,
direction,
property,
filter
	{
	"id": "int",
	"firstName": "string",
	"lastName": "string",
	"company": "string",
	"price": "int",
	"status": "string",
	"dateOfOrder": "date"
	}

addProductToOrder	

Id(of order),
 	
{
"name": "string",
"materials": [
{
"quantity": "int",
"materialId": "int"
}
],
"costPrice": "int",
"extraCharge": "int",
"status": "string",
"isDesignNeeded": "boolean",
"design": {
"description": "string",
"resultUrl": "string",
"status": "string",
"dateOfReady": "date"
}
}
	{
	"id": "int",
	"firstName": "string",
	"lastName": "string",
	"middleName": "string",
	"phone": "string",
	"mail": "string",
	"company": "string",
	"price": "int",
	"status": "string",
	"dateOfOrder": "date",
	"products": [
	{
	"id": "int",
	"name": "string",
	"materials": [
	{
	"quantity": "int",
	"material": {
	"id": "int",
	"name": "string",
	"price": "int"
	}

	}], "costPrice": "int", "extraCharge": "int", "finalPrice": "int", "status": "string", "isDesignNeeded": "boolean" }] }
updateProductInOrder	

Id(of product),
 	
{
"name": "string",
"materials": [
{
"quantity": "int",
"materialId": "int"
}
],
"costPrice": "int",
"extraCharge": "int",
"status": "string",
"isDesignNeeded": "boolean",
"design": {
"description": "string",
"resultUrl": "string",
"status": "string",
"dateOfReady": "date"
}
}
	{
	"id": "int",
	"firstName": "string",
	"lastName": "string",
	"middleName": "string",
	"phone": "string",
	"mail": "string",
	"company": "string",
	"price": "int",
	"status": "string",
	"dateOfOrder": "date",
	"products": [
	{
	"id": "int",
	"name": "string",
	"materials": [
	{
	"quantity": "int",
	"material": {
	"id": "int",
	"name": "string",
	"price": "int"
	}
	}

	], "costPrice": "int", "extraCharge": "int", "finalPrice": "int", "status": "string", "isDesignNeeded": "boolean" }] }
removeProductFromOrder	
Id	Boolean
getAllMaterials	

	[
	{
	"id": "int",
	"name": "string",
	"price": "int"
	}
	]

addMaterial	

{
"id": "int",
"name": "string",
"price": "int"
}
	{
	"id": "int",
	"name": "string",
	"price": "int"
	}

getAllFeedbacks	

page,
size,
direction,
property,
filterValue
	[
	{
	"id": "int",
	"orderId": "int",
	"serviceFeedback": "string",
	"productsFeedback": "string",
	"date": "date",
	"user": {
	"login": "string",

	"firstName": "string", "lastName": "string" } }]
saveFeedback	

Id (of order),
 	
{
"serviceFeedback": "string",
"productsFeedback": "string"
}
	{
	"id": "int",
	"orderId": "int",
	"serviceFeedback": "string",
	"productsFeedback": "string",
	"date": "date",
	"user": {
	"login": "string",
	"firstName": "string",
	"lastName": "string"
	}
	}

deleteFeedback	
Id	Boolean
getFeedbackById	

Id	{
	"id": "int",
	"orderId": "int",
	"serviceFeedback": "string",
	"productsFeedback": "string",
	"date": "date",
	"user": {
	"login": "string",
	"firstName": "string",
	"lastName": "string"
	}
	}

updateFeedback	
Id (of feedback),	{

{
"serviceFeedback": "string",
"productsFeedback": "string"
}
	"id": "int",
	"orderId": "int",
	"serviceFeedback": "string",
	"productsFeedback": "string",
	"date": "date",
	"user": {
	"login": "string",
	"firstName": "string",
	"lastName": "string"
	}
	}

Додаток И

Вхідні та вихідні параметри додатку Polygraphy-director

<table>
<tr><td colspan="2">getAllOrders</td></tr>
<tr><td>page,
size,
direction,
property,
filter</td><td>[
{
"id": "int",
"firstName": "string",
"lastName": "string",
"company": "string",
"price": "int",
"dateOfOrder": "date",
"dateOfReady": "date",
"status": "string",
"user" : {
"login": "string",
"firstName": "string",
"lastName": "string"
}
}
]</td></tr>
<tr><td colspan="2">getAllFeedbacks</td></tr>
<tr><td>page,
size,
direction,
property,
filterValue</td><td>[
{
"id": "int",
"orderId": "int",
"serviceFeedback": "string",
"productsFeedback": "string",
"date": "date",
"user": {
"login": "string",
"firstName": "string",
"lastName": "string"
}
}</td></tr>
</table>

	]
getAllDesigns	

page,
size,
direction,
property,
filter
	[
	{
	"id": "int",
	"date": "date",
	"status": "string",
	"user": {
	"login": "string",
	"firstName": "string",
	"lastName": "string"
	}
	}
	]

getFeedbackById	

Id	{
	"id": "int",
	"orderId": "int",
	"serviceFeedback": "string",
	"productsFeedback": "string",
	"date": "date",
	"userId": "int"
	}

Додаток І

Вхідні та вихідні параметри додатку Polygraphy-designer

<table>
<tr><td colspan="2">getAllProducts</td></tr>
<tr><td>page
+
size</td><td>[
"firstName": "string",
"lastName": "string",
"company": "string",
"dateOfOrder": "date",
"product": {
"id": "int",
"name": "string",
"material": {
"materialName": "string",
"quantity": "int"
}
},
"design": {
"id": "int",
"status": "string",
"userId": "int"
}
]</td></tr>
<tr><td colspan="2">getProductById</td></tr>
<tr><td>id</td><td>{
"company": "string",
"dateOfOrder": "date",
"product": {
"id": "int",
"name": "string",
"material": {
"materialName": "string",
"quantity": "int"
}
},</td></tr>
</table>

	"design": {
	"id": "int",
	"description": "string",
	"resultUrl": "string",
	"status": "string",
	"dateOfReady": "date",
	"userId": "int"
	}
	}

updateDesignById	

Id
+
{
"description": "string",
"resultUrl": "string",
"status": "string",
"dateOfReady": "string",
"userId": "string"
}
	{
	"company": "string",
	"dateOfOrder": "date",
	"product": {
	"id": "int",
	"name": "string",
	"material": {
	"materialName": "string",
	"quantity": "int"
	}
	},
	"design": {
	"id": "int",
	"description": "string",
	"resultUrl": "string",
	"status": "string",
	"dateOfReady": "date",
	"userId": "int"
	}
	}

getInProgressProductsByUserId	
userId + Page +	["firstName": "string", "lastName": "string", "company": "string",

size	"dateOfOrder": "date",
	"product": {
	"id": "int",
	"name": "string",
	"material": {
	"materialName": "string",
	"quantity": "int"
	}
	},
	"design": {
	"id": "int",
	"status": "string",
	"userId": "int"
	}
	]

getDoneProductsByUserId	

userId
+
Page
+
size
	[
	"company": "string",
	"dateOfOrder": "date",
	"product": {
	"id": "int",
	"name": "string",
	"status": "string"
	}
	]

Додаток Й

Вхідні та вихідні параметри додатку Polygraphy-jwt

registerUser(Director)	

{
"login": "string",
"password": "string",
"firstName": "string",
"lastName": "string",
"role": "string"
}
	{
	login : "string"
	}

Auth	

{
"login": "string",
"password": "string"
}
	{
	"id": "int",
	"token": "string"
	}

getUser	

id	{
	"login": "string",
	"id": "int",
	"firstName": "string",
	"lastName": "string",
	"role": "string"
	}