Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний

інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3 з дисципліни

«Компоненти програмної інженерії. Частина 3.

Архітектура програмного забезпечення»

«Мікросервісна архітектура»

Виконали студенти ІП-11 Лесів Владислав, Прищепа Владислав

Перевірив Баришич Лука Маріянович

Київ 2023

**Лабораторна робота 3**

**Мікросервісна архітектура**

**Мета**: застосувати мікросервісну архітектуру для проєктування складних систем.

**Завдання:**

1. Визначити, які сервіси між собою спілкуються синхронно.
2. Визначити, які сервіси між собою спілкуються асинхронно, за допомогою яких систем (kafka/rabbitmq/…)
3. Оновити діаграму, зобразити всі зв’язки між сервісами.
4. За допомогою діаграми послідовностей описати 2 сценарії успішної комунікації.

**Хід роботи.**

Existing services: Customer`s version of the app, Manager`s version of the app, Authentication, Order processing, Customer`s order creation, Manager`s order preparation, Payment, Shipment.

External services (systems): Payment gateways, Shipment providers.

1. Визначити, які сервіси між собою спілкуються синхронно.

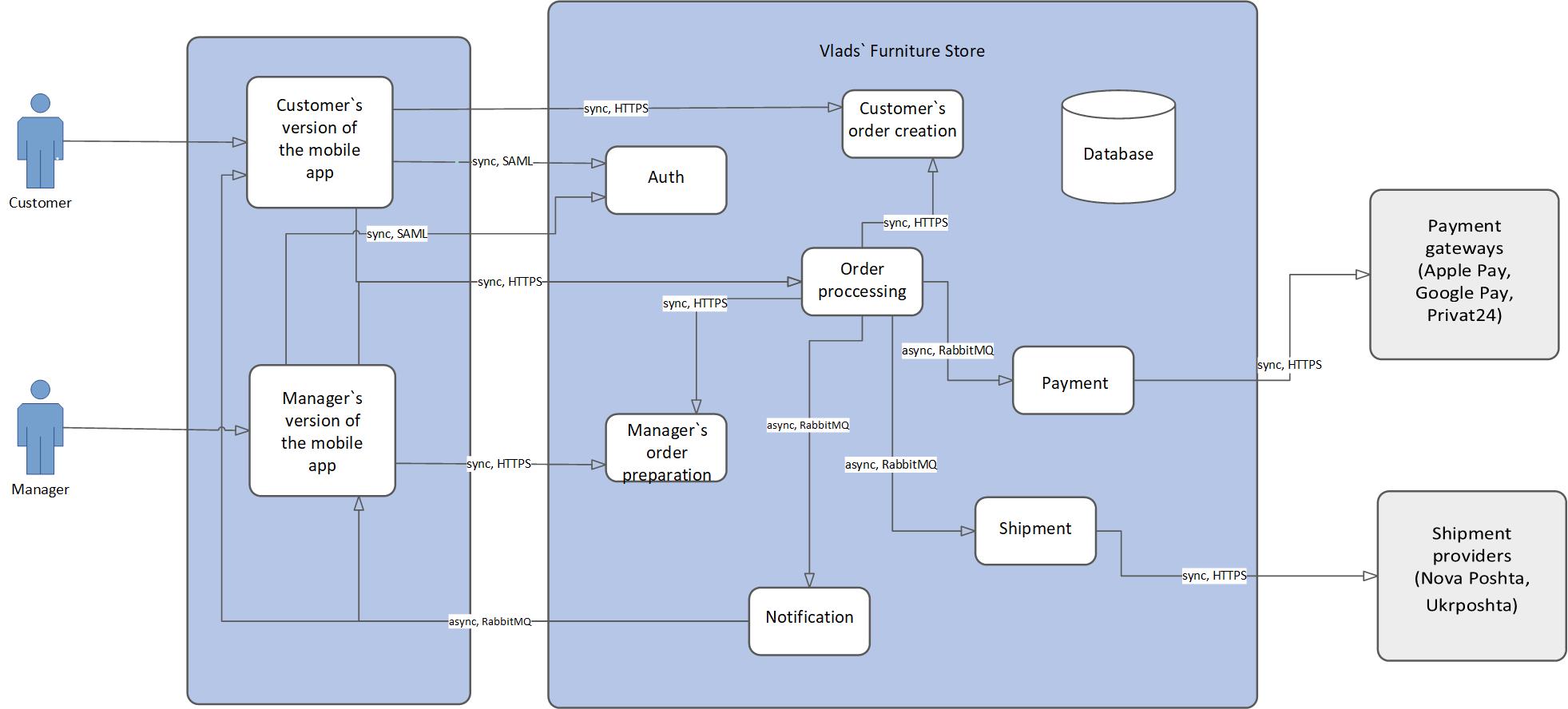
Synchronous microservices communication:

* *Authentication* and *Customer`s version of the app, Authentication* and *Manager`s version of the app*. This service is responsible for authenticating users and allowing them to access the system. The operation result is needed to move forward in the current process so it is necessary to use synchronous communication here.
* *Order processing* and *Customer`s version of the app, Order processing* and *Manager`s version of the app.* By interacting synchronously, the Order processing service can provide immediate feedback and visibility into the order proceeding process for both the customer and the manager. This allows for a more efficient order processing workflow.
* *Customer`s order creation* and *Customer`s version of the app.* When a customer places an order on the app, the order creation service needs to immediately receive and validate the customer's input, and then update the order status accordingly.
* *Manager`s order preparation* and *Manager`s version of the app.* When a manager receives an order for preparation, they need to immediately see the details of the order, validate it, and then update the order status accordingly. This requires a synchronous communication pattern to ensure that the manager has up-to-date information about the order.
* *Order processing* and *Customer`s order creation, Order processing* and *Manager`s order preparation.* When a customer places an order or a manager takes it, the order creation/preparation service needs to immediately send the order details to the order processing service to begin the fulfillment process.
* *Payment* and *External* *Payment gateways.* Payment sends a request to the payment gateway to initiate the payment process and waits for a response before proceeding*.*
* *Shipment* and *External* *Shipment providers.* Shipment sends a request to the shipment provider to initiate the shipment data confirmation and shipment process and waits for a response before proceeding.

1. Визначити які сервіси між собою спілкуються асинхронно, за допомогою яких систем.

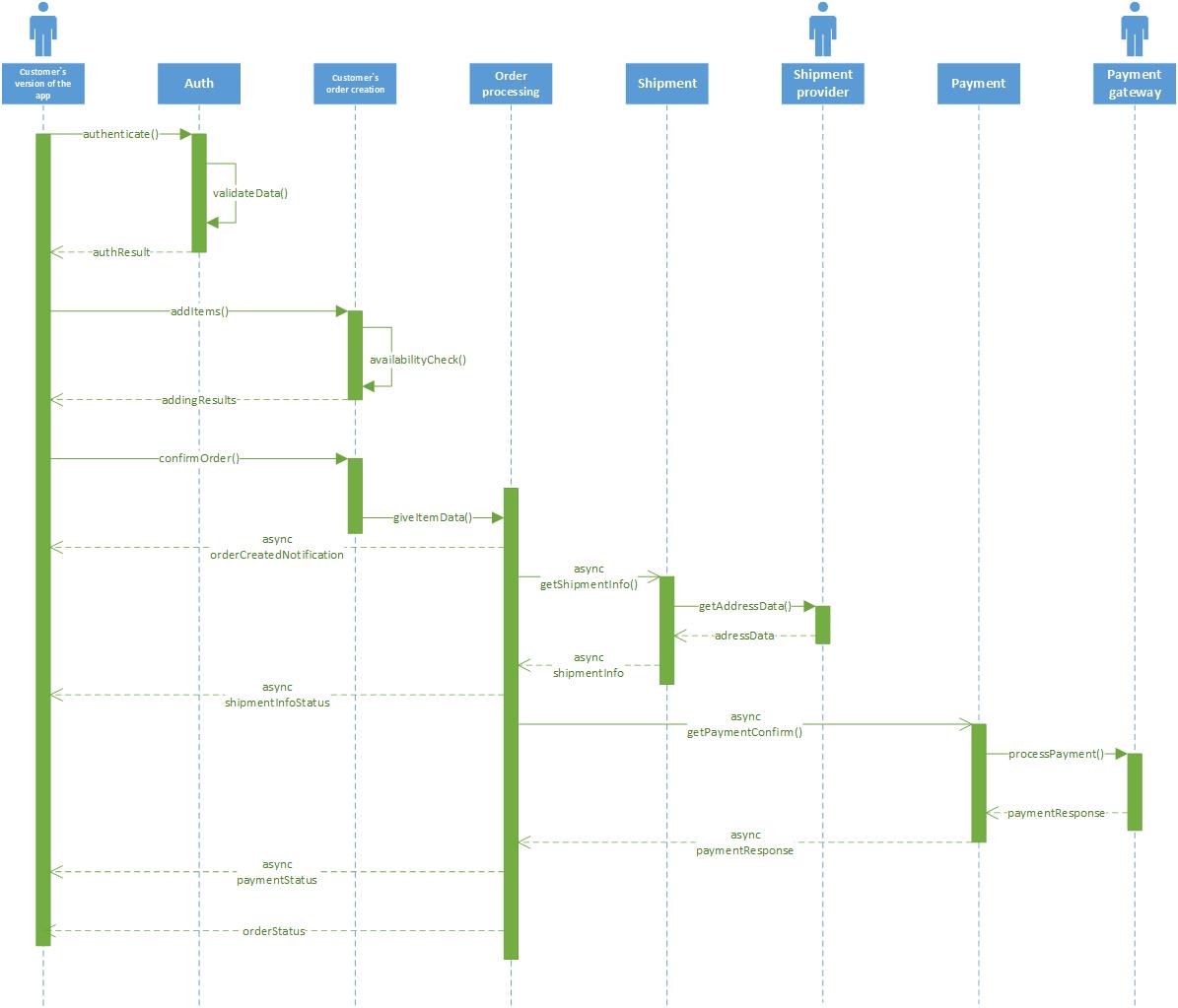
* *Payment and Order processing.* For example, if Payment is experiencing high traffic and is unable to process payments in real-time, it may store payment requests in a queue and process them asynchronously. Order processing would then retrieve payment statuses from Payment periodically or through event notifications to ensure that it has the latest payment information for each order. We use RabbitMQ here as the message queue is the best way to communicate between these services.
* *Shipment and Order processing.* If the shipment provider is experiencing high traffic and is unable to process shipment requests in real-time, it may store shipment requests in a queue and process them asynchronously. We use RabbitMQ here too.
* *Notification and Order processing.* The Order processing service may trigger one or more notifications to inform the customer or manager about the status of the order. To handle such notifications, Order processing can send the notification events to a message queue or publish them to a notification service*.* We use RabbitMQ here.
* *Notification and Manager`s version of the app, Notification and Customer`s version of the app.* They communicate asynchronously because notifications can take some time to be processed and sent, and we don't want the user to have to wait for the notification to be delivered before they can continue using the app. We use RabbitMQ here.

1. Оновити діаграму, зобразити всі зв’язки між сервісами.

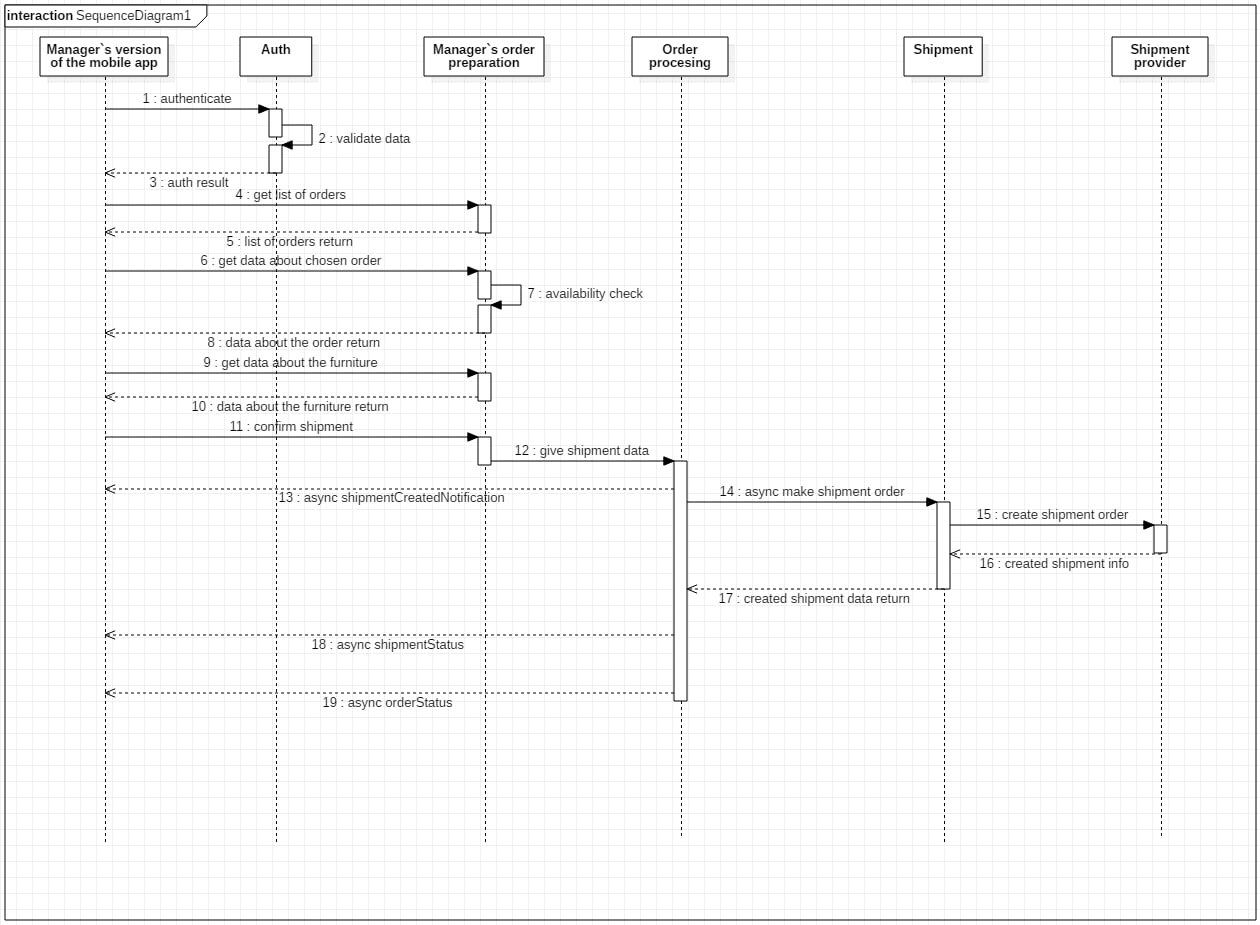


1. За допомогою діаграми послідовностей описати 2 сценарії успішної комунікації.

Scenario of successful order creation by the customer.



Scenario of successful order preparation by the manager.



**Висновок.**

Отже, у цій роботі ми отримали навички застосування мікросервісної архітектури для проєктування складних систем. У результаті лабораторної роботи відповідно до проєкту для розробки - Платформи е-комерції для магазину меблів – ми визначили, які сервіси спілкуються синхронно, а які асинхронно, оновили діаграму відповідно до взаємодії сервісів, а також побудували діаграми послідовностей для сценаріїв успішної комунікації клієнта та менеджера. Використовуючи засоби специфікування й програмний засіб MS Visio для побудови діаграм, отримуємо коректний результат.