Dependent microservices description

In our electronic store project, all three microservices depend on each other, when an administrator adds a new buyer to the database using the register microservice, its database is updated. Then the front-end microservice needs to know of the buyer's existence to allow them to log in and finally the cart microservice needs to be aware of the buyer to allow them to add items to their cart.

Database Description

The RMS database has a Buyer table with the following attributes: buyerID, firstName, lastName, email, password and phoneNumber, when a request to add a buyer is made this database will be updated with a new buyer. A new message will be published in t a channel named “add_buyer_channel” in the following format: $BUYERID, $FIRSTNAME, $LASTNAME, $EMAIL, $PASSWORD and $PHONENUMBER

Code Description

In the register microservice if a buyer is successfully registered then a message is sent into the channel with the following code:

```
public static void registerBuyer(Buyer buyer){
    CRUD_User cu = new CRUD_User();
    boolean success = cu.createUser(buyer);
    if(!success)
        return;
    String message = buyer.getBuyerID() + ":" + buyer.getFirstName() + ":" + buyer.getLastName() + ":"
            + buyer.getEmail() + ":"+ buyer.getPassword() + ":" + buyer.getPhoneNumber();
    try {
        Messaging.sendmessage(message);
    } catch (IOException ex) {
        Logger.getLogger(Register.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

The message is then sent into to queue and whenever a new request is made to the front end or cart management microservice a message-receiving thread is run which checks the queue and adds a new buyer to the database there is a new buyer.

```java
@Override
public void contextInitialized(ServletContextEvent arg0) {
    Runnable r = new Runnable() {
        public void run() {

            try {
                Messaging.Receiving_Events_Store("add_buyer_channel");
            } catch (SSLException ex) {
                Logger.getLogger(MyAppServletContextListener.class.getName()).log(Level.SEVERE, null, ex);
            } catch (ServerAddressNotSuppliedException ex) {
                Logger.getLogger(MyAppServletContextListener.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    };

    new Thread(r).start();
}
```

Inside the subscriber microservices, they receive the message and add the buyer to their database

```java
@Override
public void onNext(EventReceive value) {
    try {
        String val = (String) Converter.FromByteArray(value.getBody());
        System.out.printf("Event Received: EventID: %s, Channel: %s, Metadata: %s, Body: %s",
                value.getEventId(), value.getChannel(), value.getMetadata(),
                Converter.FromByteArray(value.getBody()));
        String[] msgParts = val.split(":");
        CRUD_User cu = new CRUD_User();
        Buyer buyer = new Buyer(msgParts[1], msgParts[2], msgParts[3], msgParts[4], Long.parseLong(msgParts[5]));
        buyer.setBuyerID(Integer.parseInt(msgParts[0]));
        cu.createUser(buyer);
    } catch (ClassNotFoundException e) {
        System.out.printf("ClassNotFoundException: %s", e.getMessage());
        e.printStackTrace();
    } catch (IOException e) {
        System.out.printf("IOException: %s", e.getMessage());
        e.printStackTrace();
    }
}
```