Message-Driven Integration

endy@artivisi.com - https://software.endy.muhardin.com

Tonight’s Menu

- What is Message Driven Integration
- RPC vs Messaging
- Message Broker Alternatives
- Case Study : STEI Tazkia Implementation
  - New Student Registration
  - Tuition Fee Payment
- Application Involved
  - New Student Registration
  - Account Receivable
  - Virtual Account Connector (2 apps)
  - Notification Connector (2 apps)
  - Academic Administration
  - Zahir Online

Message Driven Integration

Integration Options :

- No Integration - Monolithic App
- Remote Procedure Call
- Messaging
- File Transfer
- Database Sharing

Consideration : Coupling

- Runtime Coupling
- Tech Stack Coupling
- Data Schema Coupling
- Business Process Coupling

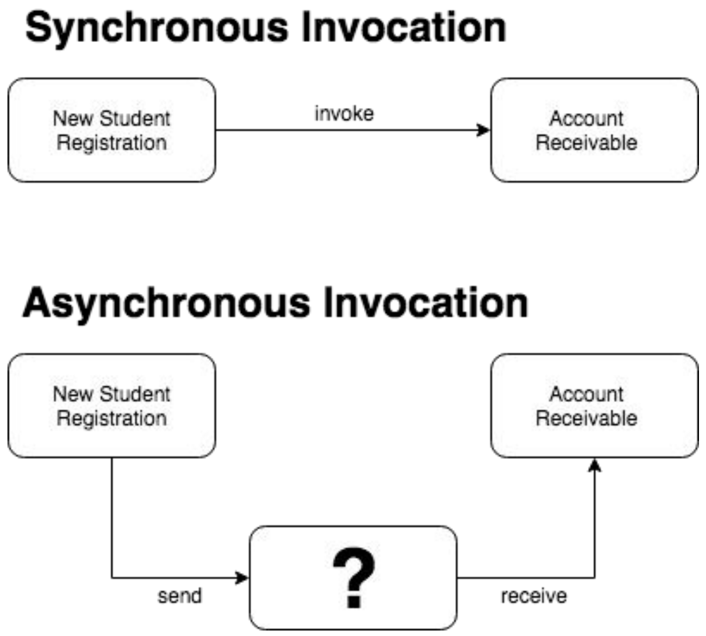
Runtime Coupling

What happens when A/R app offline?

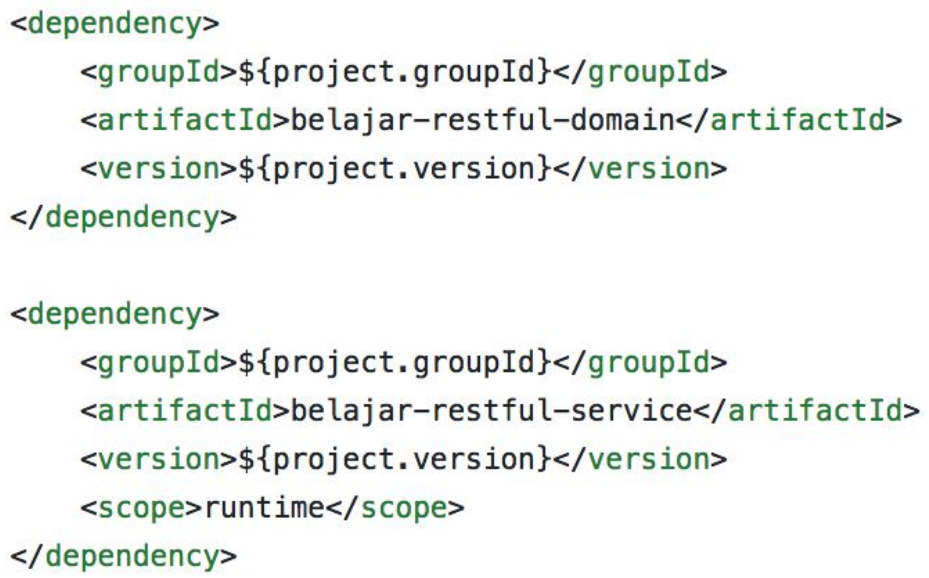
- Exception in Registration Application
- Do not know when it will be online again
- Retry + Exponential Back Off

What happens when A/R app back online?

- Before or After retry exceeded?

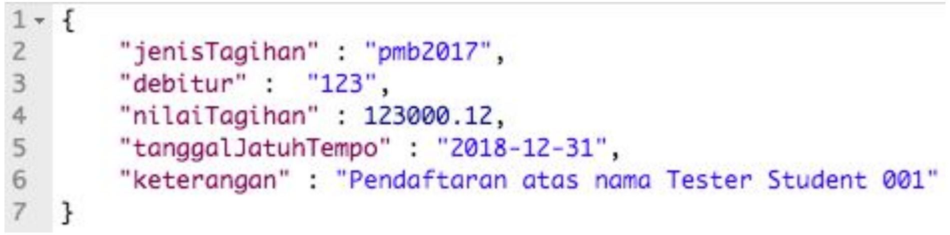


Tech Stack Coupling



```xml
<dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>belajar-restful-domain</artifactId>
    <version>${project.version}</version>
</dependency>

<dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>belajar-restful-service</artifactId>
    <version>${project.version}</version>
    <scope>runtime</scope>
</dependency>
```

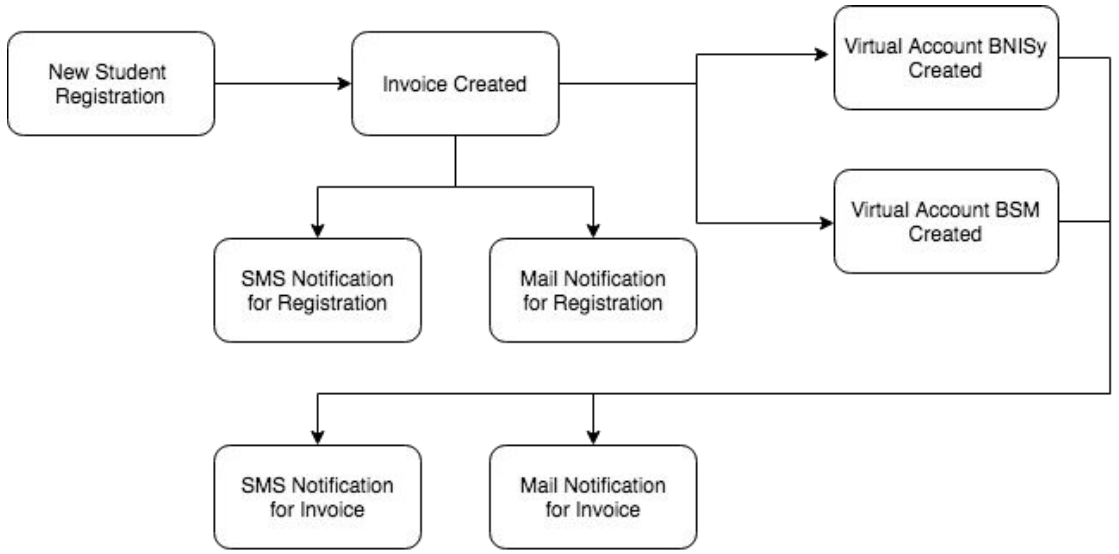
Data Schema Coupling



```json
{
    "jenisTagihan" : "pmb2017",
    "debitur" :  "123",
    "nilaiTagihan" : 123000.12,
    "tanggalJatuhTempo" : "2018-12-31",
    "keterangan" : "Pendaftaran atas nama Tester Student 001"
}
```

```java
@Data @Builder
public class TagihanRequest {
    private String jenisTagihan;
    private String debitur;
    private BigDecimal nilaiTagihan;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")
    private Date tanggalJatuhTempo;
    private String keterangan;
}
```

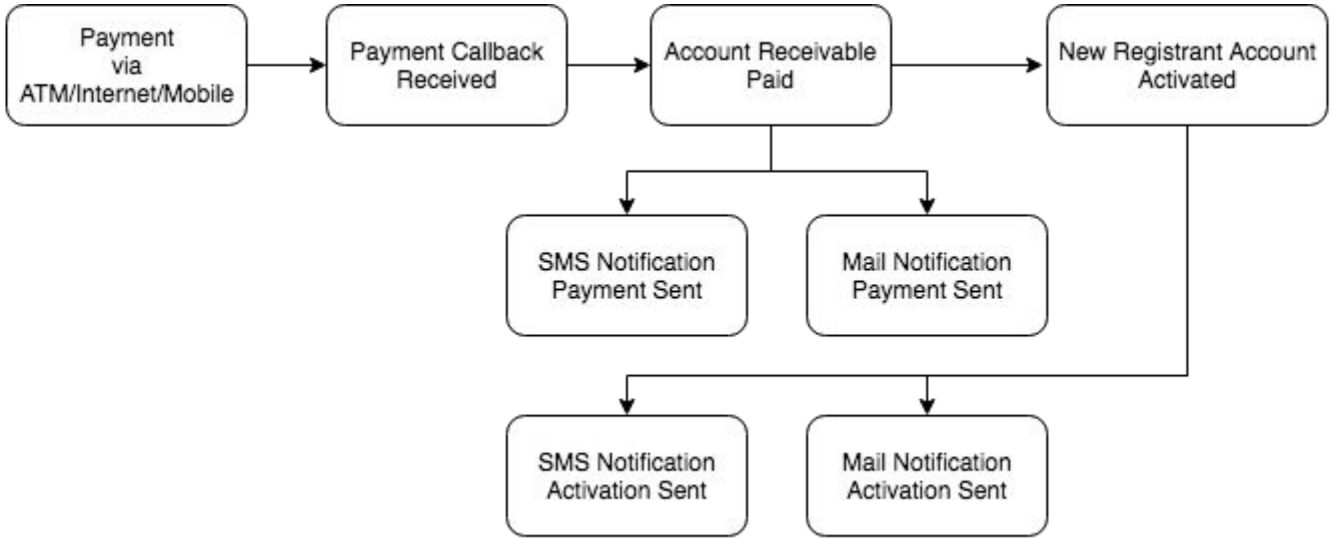
Business Process Coupling



New Student Registration
Invoice Created
Virtual Account BNISy Created
Virtual Account BSM Created
SMS Notification for Registration
Mail Notification for Registration
SMS Notification for Invoice
Mail Notification for Invoice

Registration - Invoice

Business Process Coupling


Payment via ATM/Internet/Mobile
Payment Callback Received
Account Receivable Paid
New Registrant Account Activated
SMS Notification Payment Sent
Mail Notification Payment Sent
SMS Notification Activation Sent
Mail Notification Activation Sent

Payment - Activation

Integration Options

Monolithic

- Runtime Coupling
- Tech Stack Coupling
- Data Schema Coupling
- Business Process Coupling

Remote Procedure Call

- Runtime Coupling
- ~~Tech Stack Coupling~~
- Data Schema Coupling
- Business Process Coupling

Integration Options

File Transfer

- ~~Runtime Coupling~~
- ~~Tech Stack Coupling~~
- Data Schema Coupling
- Business Process Coupling

Database Sharing

- ~~Runtime Coupling~~
- Tech Stack Coupling
- Data Schema Coupling
- Business Process Coupling

Integration Options

Messaging

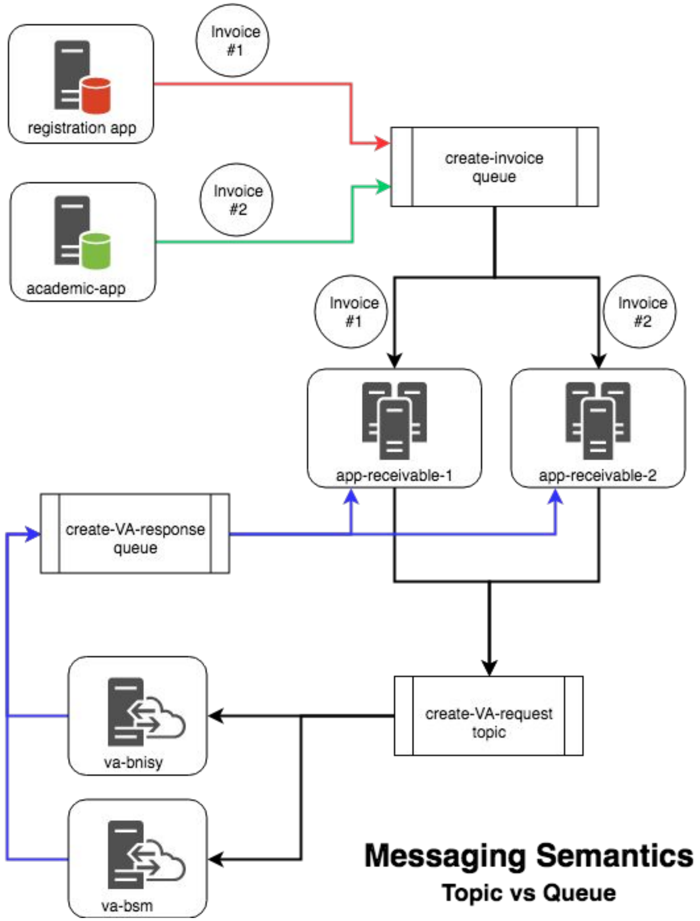
- ~~Runtime Coupling~~
- ~~Tech Stack Coupling~~
- ~~Data Schema Coupling~~
- Business Process Coupling

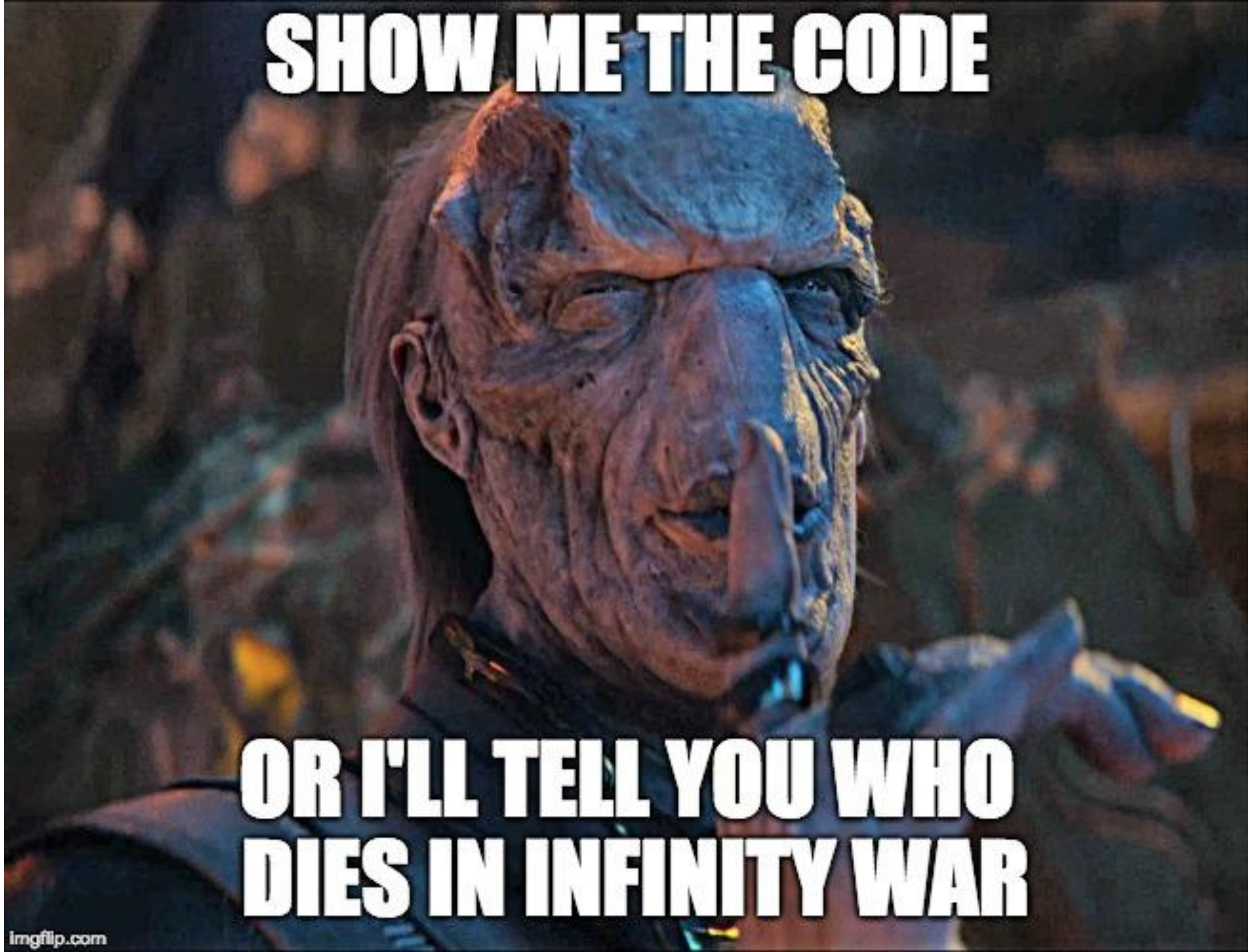
Message Broker Alternatives

- Email
    - Language-agnostic
    - Somewhat unreliable
    - Cumbersome to program
- SMS
    - Language-agnostic
    - More reliable
    - Tied to device
- Java Messaging Service (JMS)
    - Java-only
    - Not Hype Enough in 2018 ;p
- RabbitMQ
    - Language-agnostic
    - Mature
    - Reliable
- Apache Kafka
    - Language-agnostic
    - Hype-compatible (that’s all we need ;p)

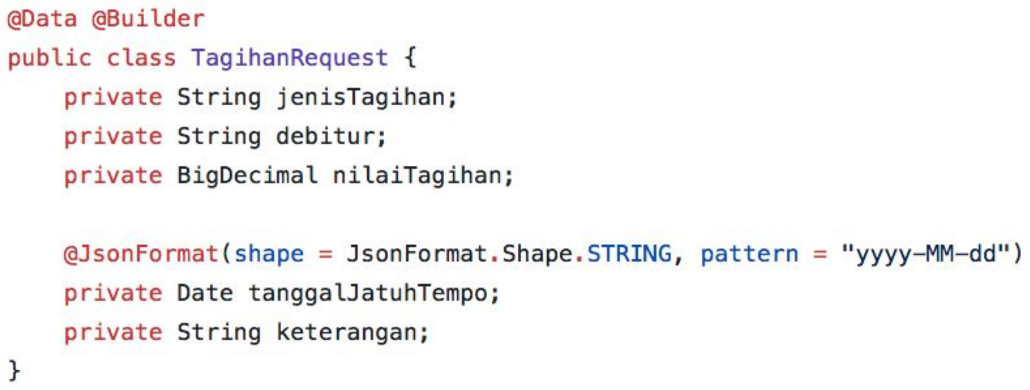
Messaging Semantics

- Queue
  - Each receiver gets different message
  - RabbitMQ : has queue feature
  - Kafka : use consumer-group-id
- Topic
  - Each receiver gets same message
- Broker vs Consumer
  - RabbitMQ : smart broker, dumb consumer
  - Kafka : dumb broker, smart consumer

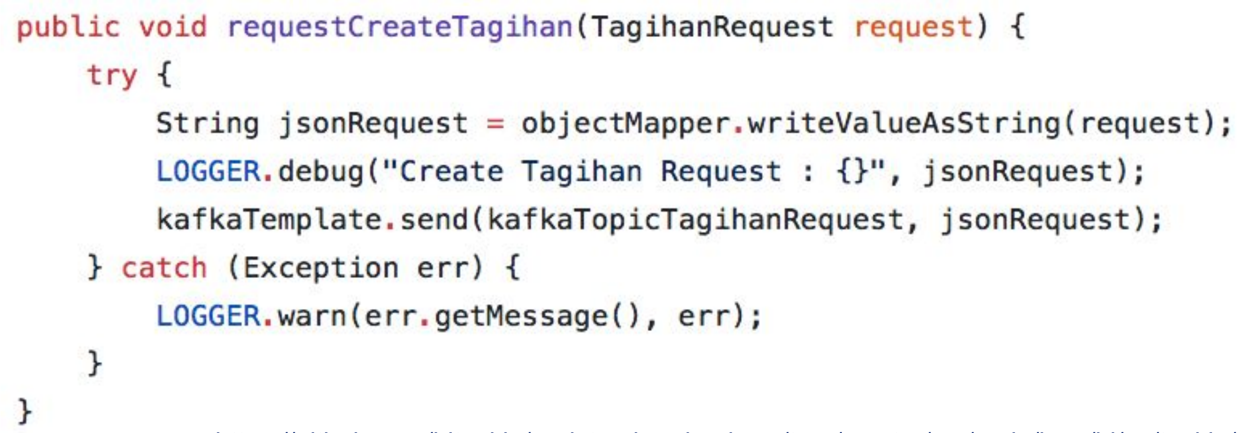



SHOW ME THE CODE
OR I'LL TELL YOU WHO DIES IN INFINITY WAR
imgflip.com

Registration App



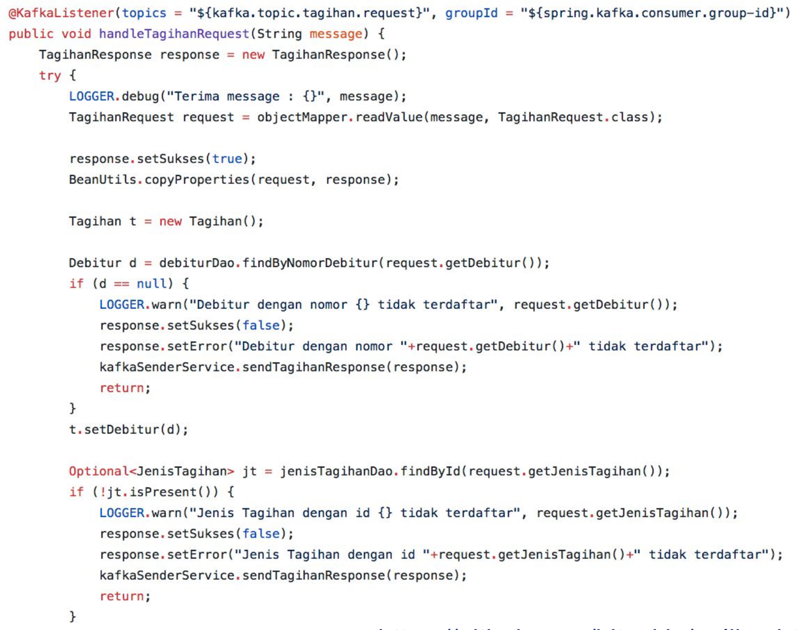
```
@Data @Builder
public class TagihanRequest {
    private String jenisTagihan;
    private String debitur;
    private BigDecimal nilaiTagihan;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")
    private Date tanggalJatuhTempo;
    private String keterangan;
}
```



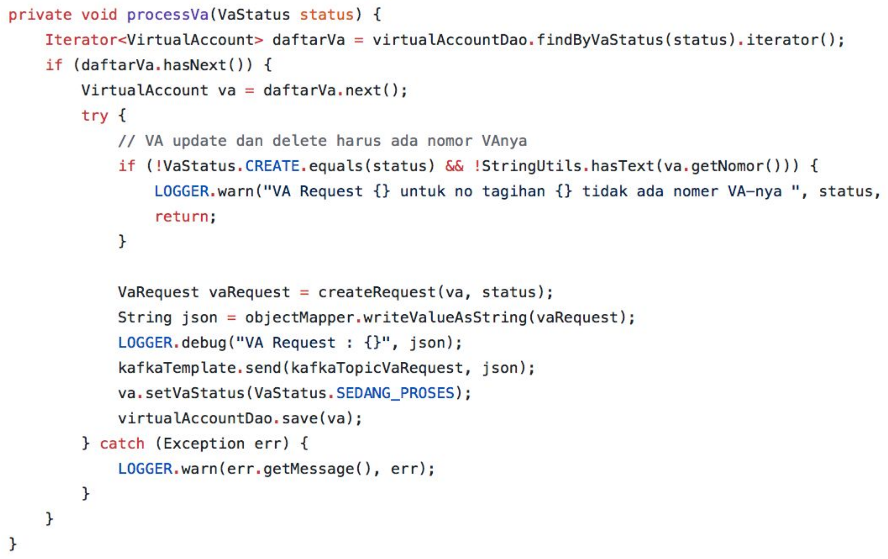
```
public void requestCreateTagihan(TagihanRequest request) {
    try {
        String jsonRequest = objectMapper.writeValueAsString(request);
        LOGGER.debug("Create Tagihan Request : {}", jsonRequest);
        kafkaTemplate.send(kafkaTopicTagihanRequest, jsonRequest);
    } catch (Exception err) {
        LOGGER.warn(err.getMessage(), err);
    }
}
```

https://github.com/idtazkia/registrasi-mahasiswa/tree/master/src/main/java/id/ac/tazkia/registration/registrasimahasiswa/service

Account Receivable App



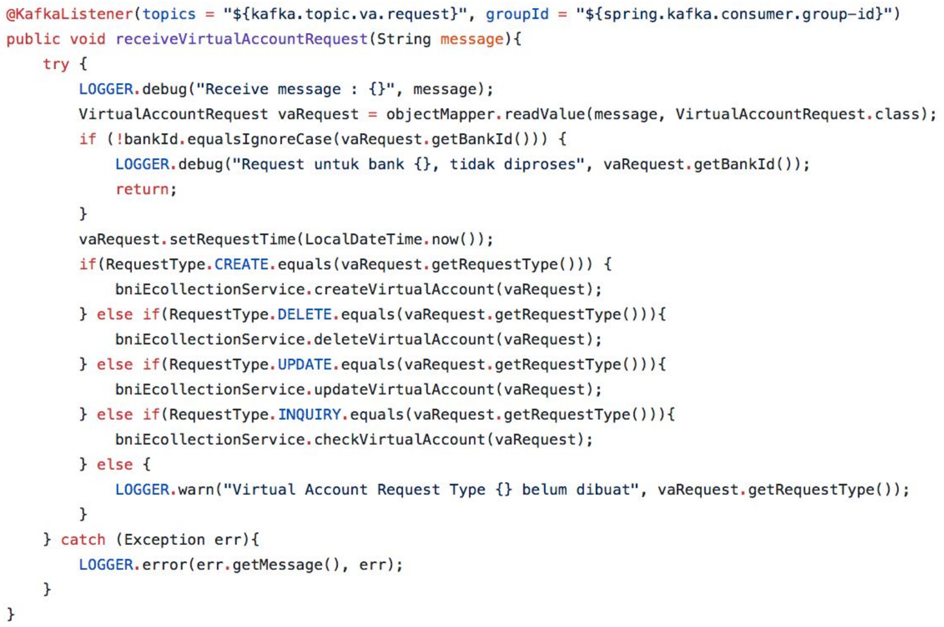
```java
@KafkaListener(topics = "${kafka.topic.tagihan.request}", groupId = "${spring.kafka.consumer.group-id}")
public void handleTagihanRequest(String message) {
    TagihanResponse response = new TagihanResponse();
    try {
        LOGGER.debug("Terima message : {}", message);
        TagihanRequest request = objectMapper.readValue(message, TagihanRequest.class);

        response.setSukses(true);
        BeanUtils.copyProperties(request, response);

        Tagihan t = new Tagihan();

        Debitur d = debiturDao.findByNomorDebitur(request.getDebitur());
        if (d == null) {
            LOGGER.warn("Debitur dengan nomor {} tidak terdaftar", request.getDebitur());
            response.setSukses(false);
            response.setError("Debitur dengan nomor "+request.getDebitur()+" tidak terdaftar");
            kafkaSenderService.sendTagihanResponse(response);
            return;
        }
        t.setDebitur(d);

        Optional<JenisTagihan> jt = jenisTagihanDao.findById(request.getJenisTagihan());
        if (!jt.isPresent()) {
            LOGGER.warn("Jenis Tagihan dengan id {} tidak terdaftar", request.getJenisTagihan());
            response.setSukses(false);
            response.setError("Jenis Tagihan dengan id "+request.getJenisTagihan()+" tidak terdaftar");
            kafkaSenderService.sendTagihanResponse(response);
            return;
        }
```



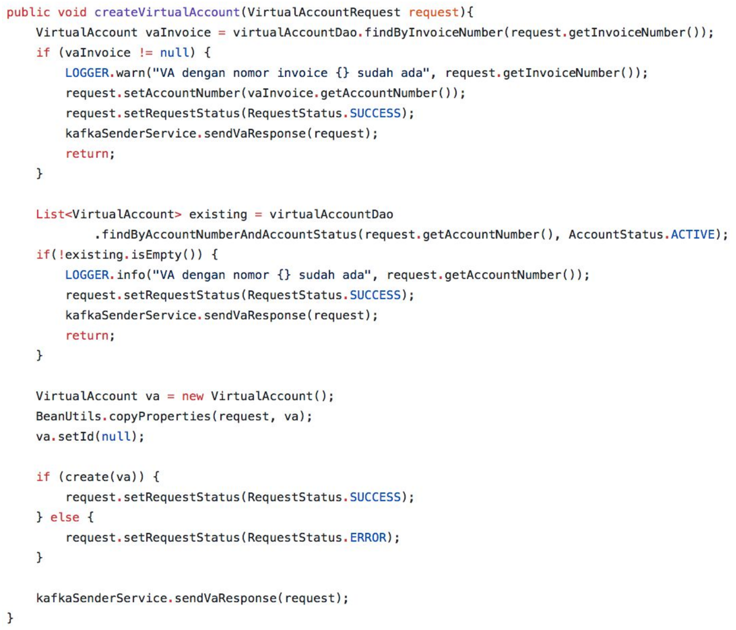
```java
private void processVa(VaStatus status) {
    Iterator<VirtualAccount> daftarVa = virtualAccountDao.findByVaStatus(status).iterator();
    if (daftarVa.hasNext()) {
        VirtualAccount va = daftarVa.next();
        try {
            // VA update dan delete harus ada nomor VAnya
            if (!VaStatus.CREATE.equals(status) && !StringUtils.hasText(va.getNomor())) {
                LOGGER.warn("VA Request {} untuk no tagihan {} tidak ada nomer VA-nya ", status,
                return;
            }

            VaRequest vaRequest = createRequest(va, status);
            String json = objectMapper.writeValueAsString(vaRequest);
            LOGGER.debug("VA Request : {}", json);
            kafkaTemplate.send(kafkaTopicVaRequest, json);
            va.setVaStatus(VaStatus.SEDANG_PROSES);
            virtualAccountDao.save(va);
        } catch (Exception err) {
            LOGGER.warn(err.getMessage(), err);
        }
    }
}
```

https://github.com/idtazkia/aplikasi-tagihan/tree/master/src/main/java/id/ac/tazkia/payment/virtualaccount/service

BNI Syariah Connector



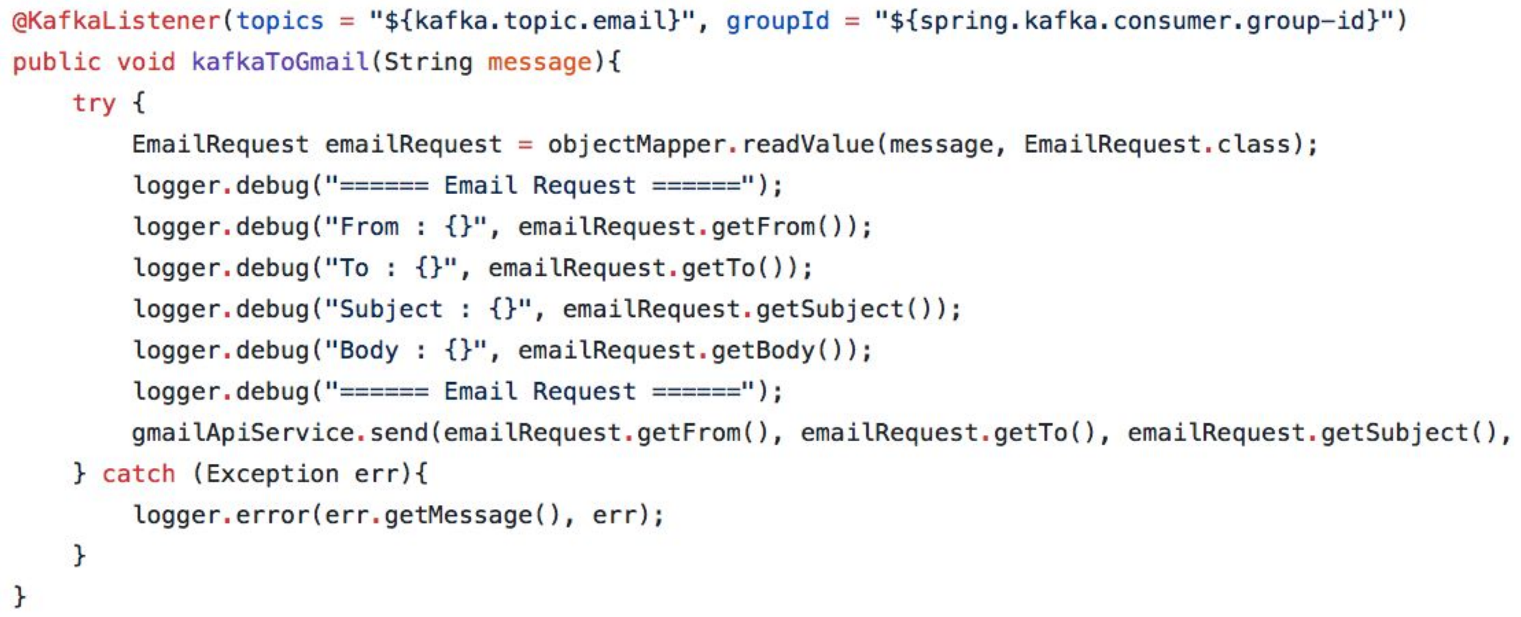
```
@KafkaListener(topics = "${kafka.topic.va.request}", groupId = "${spring.kafka.consumer.group-id}")
public void receiveVirtualAccountRequest(String message){
    try {
        LOGGER.debug("Receive message : {}", message);
        VirtualAccountRequest vaRequest = objectMapper.readValue(message, VirtualAccountRequest.class);
        if (!bankId.equalsIgnoreCase(vaRequest.getBankId())) {
            LOGGER.debug("Request untuk bank {}, tidak diproses", vaRequest.getBankId());
            return;
        }
        vaRequest.setRequestTime(LocalDateTime.now());
        if(RequestType.CREATE.equals(vaRequest.getRequestType())) {
            bniEcollectionService.createVirtualAccount(vaRequest);
        } else if(RequestType.DELETE.equals(vaRequest.getRequestType())){
            bniEcollectionService.deleteVirtualAccount(vaRequest);
        } else if(RequestType.UPDATE.equals(vaRequest.getRequestType())){
            bniEcollectionService.updateVirtualAccount(vaRequest);
        } else if(RequestType.INQUIRY.equals(vaRequest.getRequestType())){
            bniEcollectionService.checkVirtualAccount(vaRequest);
        } else {
            LOGGER.warn("Virtual Account Request Type {} belum dibuat", vaRequest.getRequestType());
        }
    } catch (Exception err){
        LOGGER.error(err.getMessage(), err);
    }
}
```



```
public void createVirtualAccount(VirtualAccountRequest request){
    VirtualAccount vaInvoice = virtualAccountDao.findByInvoiceNumber(request.getInvoiceNumber());
    if (vaInvoice != null) {
        LOGGER.warn("VA dengan nomor invoice {} sudah ada", request.getInvoiceNumber());
        request.setAccountNumber(vaInvoice.getAccountNumber());
        request.setRequestStatus(RequestStatus.SUCCESS);
        kafkaSenderService.sendVaResponse(request);
        return;
    }

    List<VirtualAccount> existing = virtualAccountDao
            .findByAccountNumberAndAccountStatus(request.getAccountNumber(), AccountStatus.ACTIVE);
    if(!existing.isEmpty()) {
        LOGGER.info("VA dengan nomor {} sudah ada", request.getAccountNumber());
        request.setRequestStatus(RequestStatus.SUCCESS);
        kafkaSenderService.sendVaResponse(request);
        return;
    }

    VirtualAccount va = new VirtualAccount();
    BeanUtils.copyProperties(request, va);
    va.setId(null);

    if (create(va)) {
        request.setRequestStatus(RequestStatus.SUCCESS);
    } else {
        request.setRequestStatus(RequestStatus.ERROR);
    }

    kafkaSenderService.sendVaResponse(request);
}
```

https://github.com/idtazkia/bnisyariah-ecollection/tree/master/src/main/java/id/ac/tazkia/payment/bnisyariah/ecollection/service

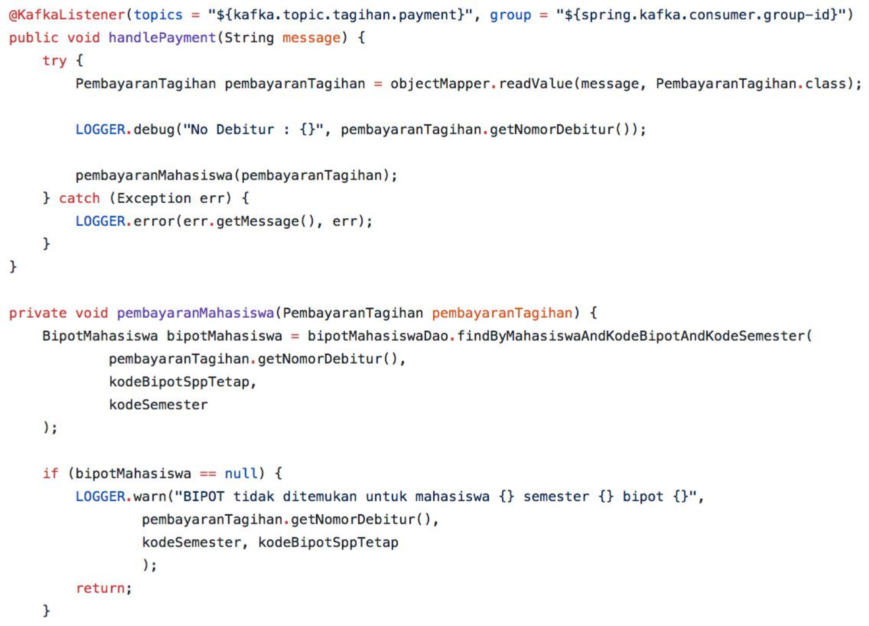
GMail Notification



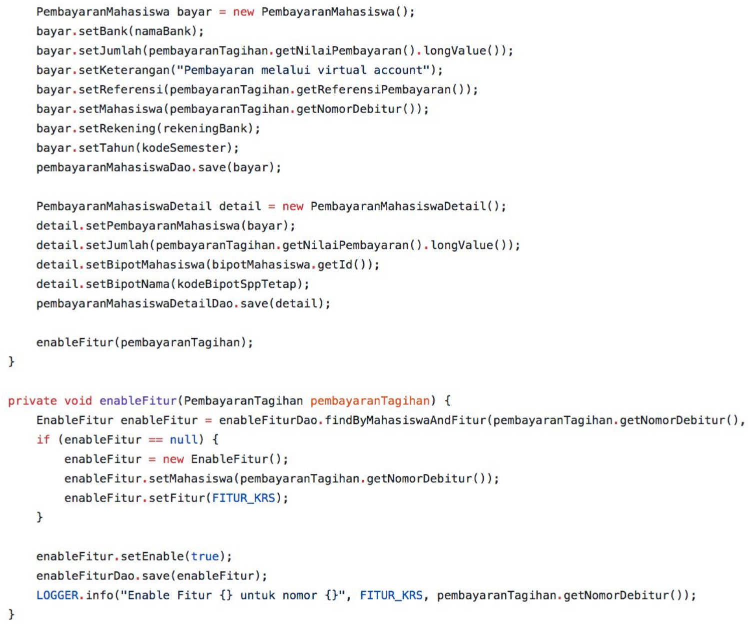
```
@KafkaListener(topics = "${kafka.topic.email}", groupId = "${spring.kafka.consumer.group-id}")
public void kafkaToGmail(String message){
    try {
        EmailRequest emailRequest = objectMapper.readValue(message, EmailRequest.class);
        logger.debug("====== Email Request ======");
        logger.debug("From : {}", emailRequest.getFrom());
        logger.debug("To : {}", emailRequest.getTo());
        logger.debug("Subject : {}", emailRequest.getSubject());
        logger.debug("Body : {}", emailRequest.getBody());
        logger.debug("====== Email Request ======");
        gmailApiService.send(emailRequest.getFrom(), emailRequest.getTo(), emailRequest.getSubject(),
    } catch (Exception err){
        logger.error(err.getMessage(), err);
    }
}
```

https://github.com/idtazkia/notifikasi-gmail/tree/master/src/main/java/id/ac/tazkia/notifikasi/gmail/service

Academic Integration



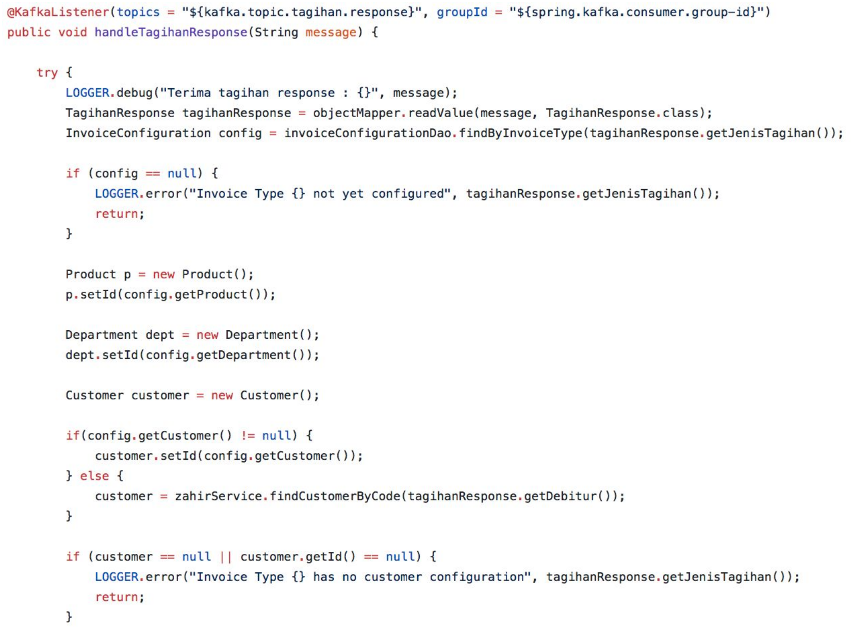
```java
@KafkaListener(topics = "${kafka.topic.tagihan.payment}", group = "${spring.kafka.consumer.group-id}")
public void handlePayment(String message) {
    try {
        PembayaranTagihan pembayaranTagihan = objectMapper.readValue(message, PembayaranTagihan.class);

        LOGGER.debug("No Debitur : {}", pembayaranTagihan.getNomorDebitur());

        pembayaranMahasiswa(pembayaranTagihan);
    } catch (Exception err) {
        LOGGER.error(err.getMessage(), err);
    }
}

private void pembayaranMahasiswa(PembayaranTagihan pembayaranTagihan) {
    BipotMahasiswa bipotMahasiswa = bipotMahasiswaDao.findByMahasiswaAndKodeBipotAndKodeSemester(
            pembayaranTagihan.getNomorDebitur(),
            kodeBipotSppTetap,
            kodeSemester
    );

    if (bipotMahasiswa == null) {
        LOGGER.warn("BIPOT tidak ditemukan untuk mahasiswa {} semester {} bipot {}",
                pembayaranTagihan.getNomorDebitur(),
                kodeSemester, kodeBipotSppTetap
                );
        return;
    }
```



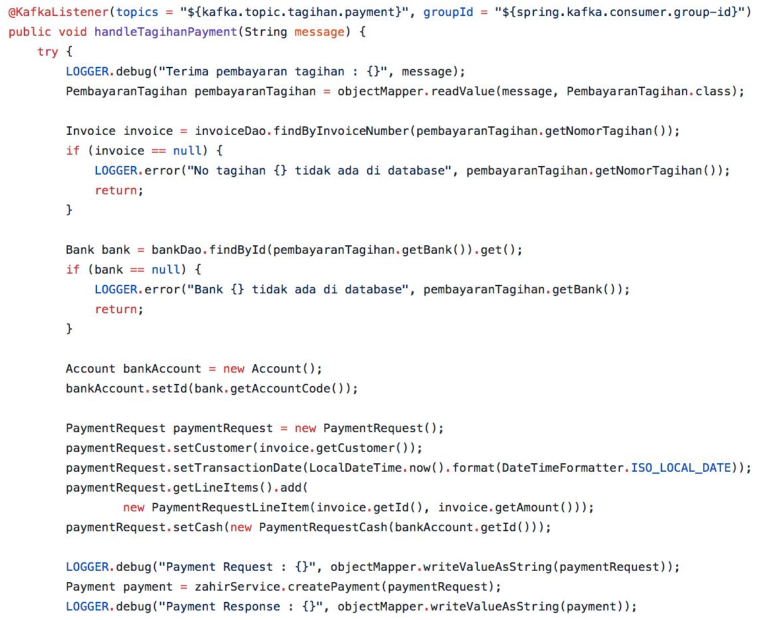
```java
    PembayaranMahasiswa bayar = new PembayaranMahasiswa();
    bayar.setBank(namaBank);
    bayar.setJumlah(pembayaranTagihan.getNilaiPembayaran().longValue());
    bayar.setKeterangan("Pembayaran melalui virtual account");
    bayar.setReferensi(pembayaranTagihan.getReferensiPembayaran());
    bayar.setMahasiswa(pembayaranTagihan.getNomorDebitur());
    bayar.setRekening(rekeningBank);
    bayar.setTahun(kodeSemester);
    pembayaranMahasiswaDao.save(bayar);

    PembayaranMahasiswaDetail detail = new PembayaranMahasiswaDetail();
    detail.setPembayaranMahasiswa(bayar);
    detail.setJumlah(pembayaranTagihan.getNilaiPembayaran().longValue());
    detail.setBipotMahasiswa(bipotMahasiswa.getId());
    detail.setBipotNama(kodeBipotSppTetap);
    pembayaranMahasiswaDetailDao.save(detail);

    enableFitur(pembayaranTagihan);
}

private void enableFitur(PembayaranTagihan pembayaranTagihan) {
    EnableFitur enableFitur = enableFiturDao.findByMahasiswaAndFitur(pembayaranTagihan.getNomorDebitur(),
    if (enableFitur == null) {
        enableFitur = new EnableFitur();
        enableFitur.setMahasiswa(pembayaranTagihan.getNomorDebitur());
        enableFitur.setFitur(FITUR_KRS);
    }

    enableFitur.setEnable(true);
    enableFiturDao.save(enableFitur);
    LOGGER.info("Enable Fitur {} untuk nomor {}", FITUR_KRS, pembayaranTagihan.getNomorDebitur());
}
```

https://github.com/idtazkia/simak-kafka/blob/master/src/main/java/id/ac/tazkia/simak/kafka/service/KafkaListenerService.java

Zahir Online Integration



```
@KafkaListener(topics = "${kafka.topic.tagihan.response}", groupId = "${spring.kafka.consumer.group-id}")
public void handleTagihanResponse(String message) {

    try {
        LOGGER.debug("Terima tagihan response : {}", message);
        TagihanResponse tagihanResponse = objectMapper.readValue(message, TagihanResponse.class);
        InvoiceConfiguration config = invoiceConfigurationDao.findByInvoiceType(tagihanResponse.getJenisTagihan());

        if (config == null) {
            LOGGER.error("Invoice Type {} not yet configured", tagihanResponse.getJenisTagihan());
            return;
        }

        Product p = new Product();
        p.setId(config.getProduct());

        Department dept = new Department();
        dept.setId(config.getDepartment());

        Customer customer = new Customer();

        if(config.getCustomer() != null) {
            customer.setId(config.getCustomer());
        } else {
            customer = zahirService.findCustomerByCode(tagihanResponse.getDebitur());
        }

        if (customer == null || customer.getId() == null) {
            LOGGER.error("Invoice Type {} has no customer configuration", tagihanResponse.getJenisTagihan());
            return;
        }
```



```
@KafkaListener(topics = "${kafka.topic.tagihan.payment}", groupId = "${spring.kafka.consumer.group-id}")
public void handleTagihanPayment(String message) {
    try {
        LOGGER.debug("Terima pembayaran tagihan : {}", message);
        PembayaranTagihan pembayaranTagihan = objectMapper.readValue(message, PembayaranTagihan.class);

        Invoice invoice = invoiceDao.findByInvoiceNumber(pembayaranTagihan.getNomorTagihan());
        if (invoice == null) {
            LOGGER.error("No tagihan {} tidak ada di database", pembayaranTagihan.getNomorTagihan());
            return;
        }

        Bank bank = bankDao.findById(pembayaranTagihan.getBank()).get();
        if (bank == null) {
            LOGGER.error("Bank {} tidak ada di database", pembayaranTagihan.getBank());
            return;
        }

        Account bankAccount = new Account();
        bankAccount.setId(bank.getAccountCode());

        PaymentRequest paymentRequest = new PaymentRequest();
        paymentRequest.setCustomer(invoice.getCustomer());
        paymentRequest.setTransactionDate(LocalDateTime.now().format(DateTimeFormatter.ISO_LOCAL_DATE));
        paymentRequest.getLineItems().add(
                new PaymentRequestLineItem(invoice.getId(), invoice.getAmount()));
        paymentRequest.setCash(new PaymentRequestCash(bankAccount.getId()));

        LOGGER.debug("Payment Request : {}", objectMapper.writeValueAsString(paymentRequest));
        Payment payment = zahirService.createPayment(paymentRequest);
        LOGGER.debug("Payment Response : {}", objectMapper.writeValueAsString(payment));
```

https://github.com/idtazkia/zahir-kafka/tree/master/src/main/java/id/ac/tazkia/zahir/service

Thank You