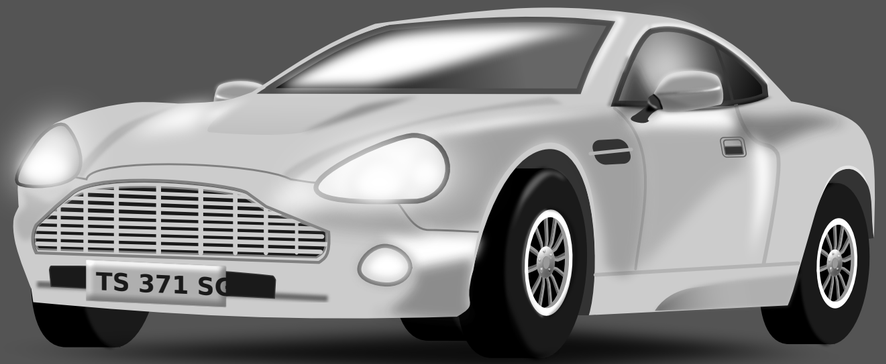
TRABALHO FINAL DO MÓDULO 4.3 E 4.4
KAFKA + SCHEDULE

SISTEMA DE ALUGUEL DE VEÍCULOS

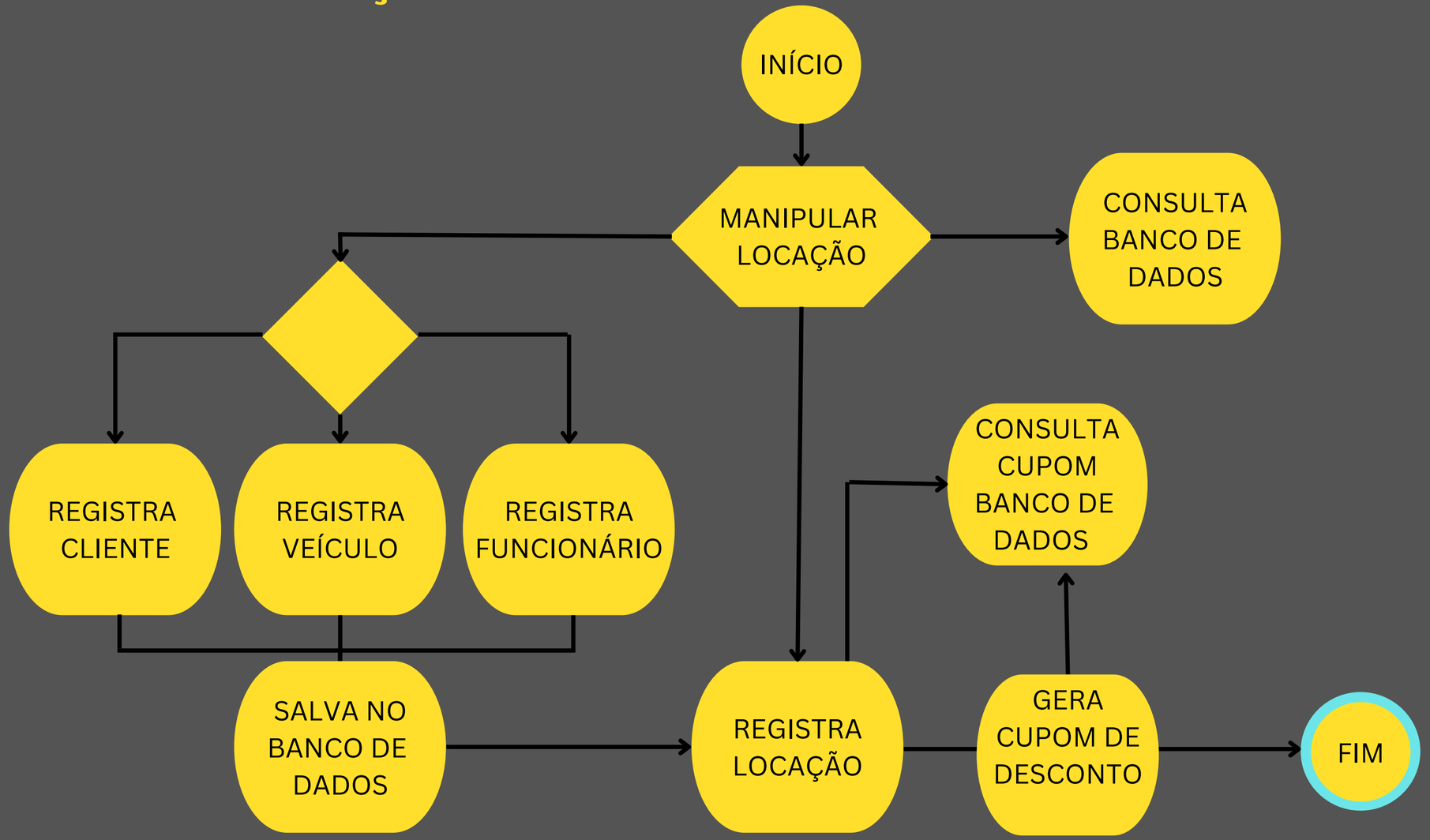


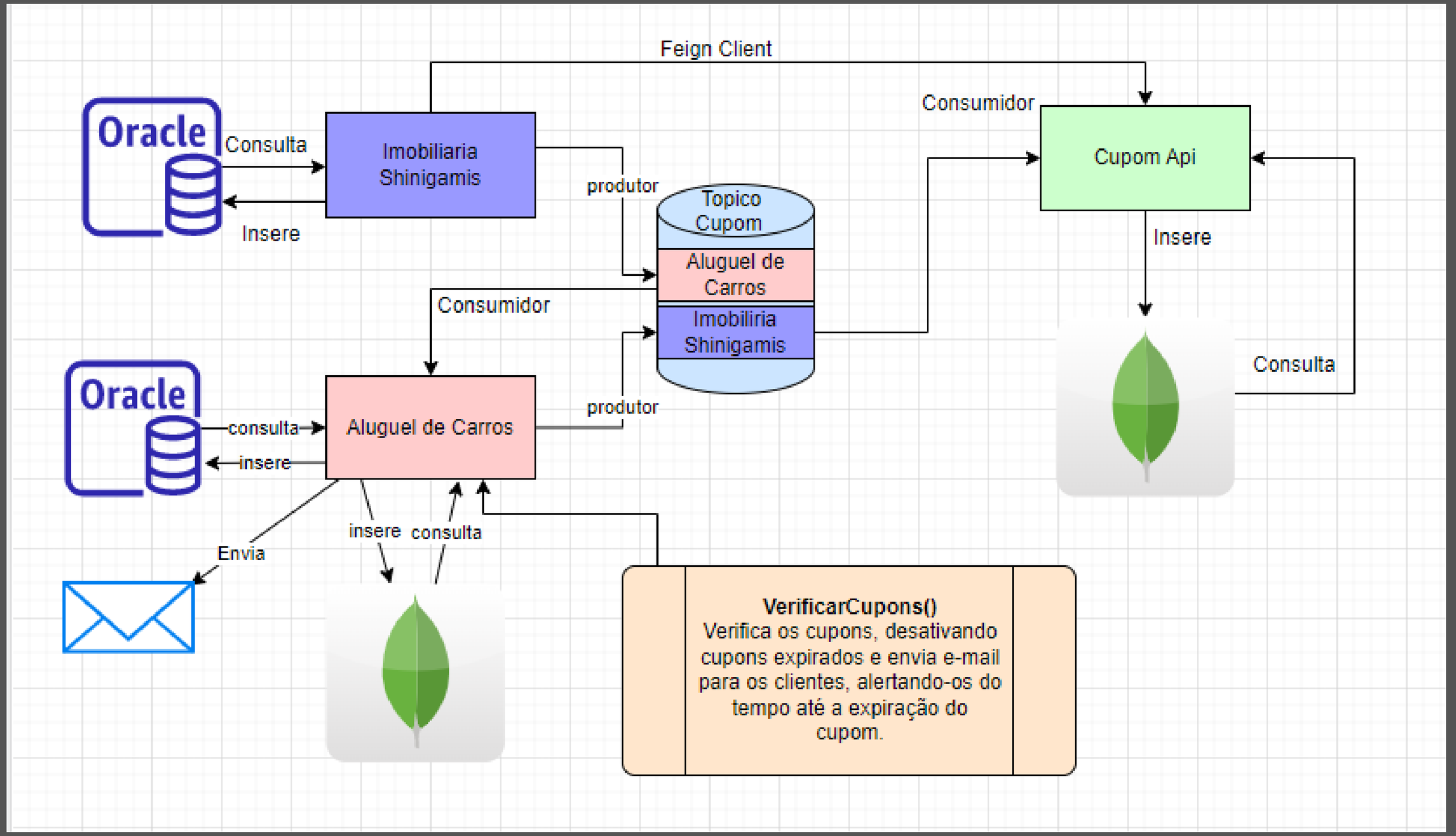
EQUIPE 8 (COFFEE CODE) : LUCAS MIGUEL- BRUNO PELLICIARI - JHENNYFER SOBRINHO

Sobre o app

- **Uma aplicação voltada para funcionários de locadoras de veículos que tem como função facilitar registros, consultas de locações em banco de dados com segurança, realizar relatórios específicos para locadoras e fazer pagamentos no cartão de credito.**
- **Ao registrar uma locação e pode usar cupom de desconto gerado pela imobiliária Shinigamis.**
- **Após registrar uma locação e realizar pagamento com cartão de credito gera cupom de desconto para próxima locação na imobiliária Shinigamis.**

FLUXOGRAMA DA APLICAÇÃO




Feign Client
Oracle
Consulta
Insere
Imobiliaria Shinigamis
produtor
Topico Cupom
Aluguel de Carros
Imobiliria Shinigamis
Consumidor
Cupom Api
Insere
Consulta
Consumidor
Oracle
consulta
insere
Aluguel de Carros
produtor
Envia
insere
consulta
VerificarCupons()
Verifica os cupons, desativando cupons expirados e envia e-mail para os clientes, alertando-os do tempo até a expiração do cupom.

Principais Funcionalidades

- **Permite ao usuario se autenticar no sistema, garantindo permissão as funcionalidades.**
- **CRUD das entidades do projeto.**
- **Criação de locação, gerando um cupom para outra API.**
- **Geração de Relatórios.**
- **Permite utilizar cupom gerado por outra API.**

O método ao lado recebe o email do cliente que realizou uma locação, gera um cupom e o envia para a partição 1 do tópico de cupons, compartilhado com a imobiliária Shinigamis.

```java
public void enviarMensagem(String email) throws JsonProcessingException {
        CupomCreateDTO cupomCreateDTO = new CupomCreateDTO();
        cupomCreateDTO.setAtivo(true);
        cupomCreateDTO.setEmail(email);
        cupomCreateDTO.setDesconto(PORCENTAGEM_DESCONTO);
        cupomCreateDTO.setDataCriacao(LocalDate.now());
        cupomCreateDTO.setDataVencimento(LocalDate.now().plusDays(VALIDADE_CUPOM));

        String mensagemStr = objectMapper.writeValueAsString(cupomCreateDTO);
        MessageBuilder<String> stringMessageBuilder = MessageBuilder.withPayload(mensagemStr)
                .setHeader(KafkaHeaders.TOPIC, topic)
                .setHeader(KafkaHeaders.MESSAGE_KEY, UUID.randomUUID().toString())
                .setHeader(KafkaHeaders.PARTITION_ID, 1);

        Message<String> message = stringMessageBuilder.build();
        ListenableFuture<SendResult<String, String>> enviadoParaTopico = kafkaTemplate.send(message);
        enviadoParaTopico.addCallback(new ListenableFutureCallback<>() {
            @Override
            public void onSuccess(SendResult result) {
                String cpf =
SecurityContextHolder.getContext().getAuthentication().getPrincipal().toString();
                logService.salvarLog(new LogCreateDTO(TipoLog.CREATE, " CPF logado: " + cpf,
EntityLog.CUPOM));
            }

            @Override
            public void onFailure(Throwable ex) {
                log.error(" Erro ao publicar duvida no kafka com a mensagem");
            }
        });
    }
```

Método que irá consumir os cupons inseridos pela imobiliária Shinigamis na partição 0 do tópico de cupons.

```java
@KafkaListener(
            clientIdPrefix = "${spring.kafka.consumer.client-id}",
            groupId = "${spring.kafka.consumer.group-id}",
            topicPartitions = {@TopicPartition(topic = "${kafka.topic}", partitions = {"0"})}
    )
    public void consumirMensagensIndividuais(@Payload String mensagem) throws JsonProcessingException {
        CupomDTO cupomDTO = objectMapper.readValue(mensagem, CupomDTO.class);
        cupomService.save(cupomDTO);
    }
```

Ao criar uma locação, será verificado se está sendo passado um cupom válido. Após a verificação de existência, caso o cupom seja válido, será aplicado o desconto lá especificado.

```java
public LocacaoDTO create(LocacaoCreateDTO locacaoCreateDTO) throws RegraDeNegocioException,
JsonProcessingException {
        LocacaoEntity locacaoEntity = criarLocacaoAPartirDeIds(locacaoCreateDTO);
        ContatoEntity contato =
locacaoEntity.getClienteEntity().getContatoEntities().stream().toList().get(0);

        if (!locacaoCreateDTO.getCupom().trim().isEmpty()) {
            CupomDTO cupomDTO = cupomService.findCupom(locacaoCreateDTO.getCupom());
            locacaoEntity.setValorLocacao(locacaoEntity.getValorLocacao()*(1-
(cupomDTO.getDesconto()/100)));
            cupomDTO.setAtivo(false);
            cupomService.save(cupomDTO);
        }
        produtorService.enviarMensagem(contato.getEmail());
        LocacaoEntity locacaoSave = locacaoRepository.save(locacaoEntity);

        String base = getString(locacaoSave);

        emailService.sendEmail(base, locacaoSave.getFuncionarioEntity().getEmail());

        RelatorioLocacaoDTO relatorioLocacaoDTO = getRelatorioLocacaoDTO(locacaoSave);

        relatorioLocacaoRepository.save(relatorioLocacaoDTO);
        String cpf = SecurityContextHolder.getContext().getAuthentication().getPrincipal().toString();
        logService.salvarLog(new LogCreateDTO(TipoLog.CREATE, "CPF logado: " + cpf,
EntityLog.LOCACAO));
        return converterEmDTO(locacaoSave);
    }
```

Métodos existentes em CupomService que são utilizados para verificar se os cupons informados são válidos, tanto no sentido de existirem no sistema, quanto no de checar se ainda não foram aplicados em uma outra locação.

```java
public CupomDTO findCupom(String idCupom) throws RegraDeNegocioException {
        CupomEntity cupomEntity = findById(idCupom);
        if (cupomEntity.isAtivo()) {
            return converterEmDTO(cupomEntity);
        } else {
            throw new RegraDeNegocioException("Cupom já utilizado!");
        }
    }

    private CupomEntity findById(String idCupom) throws RegraDeNegocioException {
        return cupomRepository.findById(idCupom)
                .orElseThrow(() -> new RegraDeNegocioException("Cupom invalido!"));
    }
```

Schedule que enviará, às 00h00, uma mensagem para os e-mails contidos em nosso banco que possuem um cupom válido para ser resgatado. Caso o cupom esteja prestes a expirar, será feita sua invalidação e o usuário associado será notificado. Ademais, o método notificará aos clientes, também, o tempo restante em cupons válidos antes de sua expiração.

```java
@Scheduled(cron = "0 0 * * * *")
    public void verificarCupons() {
        List<CupomEntity> cupomEntityList = findAllCupomValido();
        for (CupomEntity cupom : cupomEntityList) {
            String base = "";
            if (cupom.getDataVencimento().isBefore(LocalDate.now())) {
                cupom.setAtivo(false);
                cupomRepository.save(cupom);
                base = "Olá, seu cupom(" + cupom.getId() + "acaba de expirar!";
            } else if (cupom.getDataVencimento().isAfter(LocalDate.now())) {
                long tempo = LocalDate.now().until(cupom.getDataVencimento(), ChronoUnit.DAYS);
                base = "Olá, resta " + tempo + " dias para seu cupom expirar! O id do cupom é : " +
cupom.getId();
            }
            emailService.sendEmail(base, cupom.getEmail());
        }
    }
```

Principais Dificuldades

- **Testar o Scheduler**
- **Testar integração entre as aplicações**