DBC MOVIES

GRUPO 7 – ALISON AILSON, LUIZ MARTINS E EDUARDO SEDREZ

MÉTODO PRODUTOR

Nos métodos sendTo e LocarFilme procura um filme pelo id, pega o usuario logado com loggedUser, na locadoraDto seta o usuario, o filme, dias de locação, dataatual e envia.

```
1 usage  Alison8547 +1
public void sendTo(LocadoraDto locadora) throws JsonProcessingException {

    String mensagemStr = objectMapper.writeValueAsString(locadora);

    MessageBuilder<String> stringMessageBuilder = MessageBuilder.withPayload(mensagemStr)
            .setHeader(KafkaHeaders.TOPIC, topico)
            .setHeader(KafkaHeaders.MESSAGE_KEY, UUID.randomUUID().toString())
            .setHeader(KafkaHeaders.PARTITION_ID, particao);

    Message<String> message = stringMessageBuilder.build();

    ListenableFuture<SendResult<String, String>> enviadoParaTopico = kafkaTemplate.send(message);
    Alison8547 +1
    enviadoParaTopico.addCallback(new ListenableFutureCallback<>() {
        Alison8547 +1
        @Override
        public void onSuccess(SendResult result) {
            log.info("{} Sua mensagem foi enviado com sucesso!", locadora.getUsuario().getNome());
        }

        Alison8547
        @Override
        public void onFailure(Throwable ex) { log.error(" Erro ao enviar: {}", locadora, ex); }
    });
}
1 usage  Luiz Martins
public void locarFilme(Integer idFilme, Integer qtdDiasLocacao) throws RegraDeNegocioException, JsonProcessingException {
    ItemEntretenimentoEntity itemEntity = itemService.findById(idFilme);
    FilmeDisponivelDto filme = objectMapper.convertValue(itemEntity, FilmeDisponivelDto.class);

    Integer idUsuario = usuarioService.getLoggedUser().getIdUsuario();
    UsuarioLocacaoDto usuarioDto = objectMapper.convertValue(usuarioService.findById(idUsuario), UsuarioLocacaoDto.class);

    LocadoraDto locadora = new LocadoraDto();

    locadora.setUsuario(usuarioDto);
    locadora.setFilme(filme);
    locadora.setQtdDiasLocacao(qtdDiasLocacao);
    locadora.setData(LocalDateTime.now());

    locadoraProdudorService.sendTo(locadora);
}
```

MÉTODO CONSUMIDOR

No método consumirEventoLocacao utiliza o payload para consumir os dados vindos do método produtor da API DBC-Movies

```
Luiz Martins +2
@KafkaListener(
        clientIdPrefix = "locadora",
        groupId = "locadora",
        topicPartitions = {@TopicPartition(topic = "${kafka.topic}", partitions = {"${kafka.partition}"})}
)
public void consumirEventoLocacao(@Payload String mensagem) throws JsonProcessingException {
    locacaoCreateDto = objectMapper.readValue(mensagem, LocadoraCreateDto.class);
    log.info("Nova locação recebida\n ### Dados Locação ###\n" +
                    "Nome do Usário: {}\n" +
                    "Email do Usuário: {}\n" +
                    "Idade: {}\n" +
                    "Nome do filme: {}\n" +
                    "Data: {}\n" +
                    "Quantidade de dias alugado: {}",
            locacaoCreateDto.getUsuario().getNome(),
            locacaoCreateDto.getUsuario().getEmail(),
            locacaoCreateDto.getUsuario().getIdade(),
            locacaoCreateDto.getFilme().getNome(),
            locacaoCreateDto.getData(),
            locacaoCreateDto.getQtdDiasLocacao());
    Double valorTotalLocacao = locacaoCreateDto.getQtdDiasLocacao() * locacaoCreateDto.getFilme().getPreco();
    LocadoraEntity locacaoEntity = objectMapper.convertValue(locacaoCreateDto, LocadoraEntity.class);
    locacaoEntity.setValorTotal(valorTotalLocacao);

    locadoraRepository.save(locacaoEntity);
    log.info("Locação salva com sucesso");
}
```

```
Edyfiva +1
@Scheduled(cron = "0 0 * * * *")
public void reportarEmailLocacao() {
    List<LocadoraEntity> emailGeral = locadoraRepository.findAll();
    emailGeral.stream().forEach(locadora -> emailService.sendEmailUsuario(objectMapper.convertValue(locadora, LocadoraDto.class)));
}
```

SCHEDULE

Utilizando o schedule nós geramos e-mails todos os dias à meia noite informando a data de locação e valor total ao usuário que locou o filme

DIAGRAMA DE FLUXO

No nosso diagrama de fluxo temos duas API's as quais fazem uso do mongo e oracle, trocando informações através do kafka. Também temos um relatório gerado pelo schedule o qual envia e-mail de notificação aos usuários a respeito de suas locações