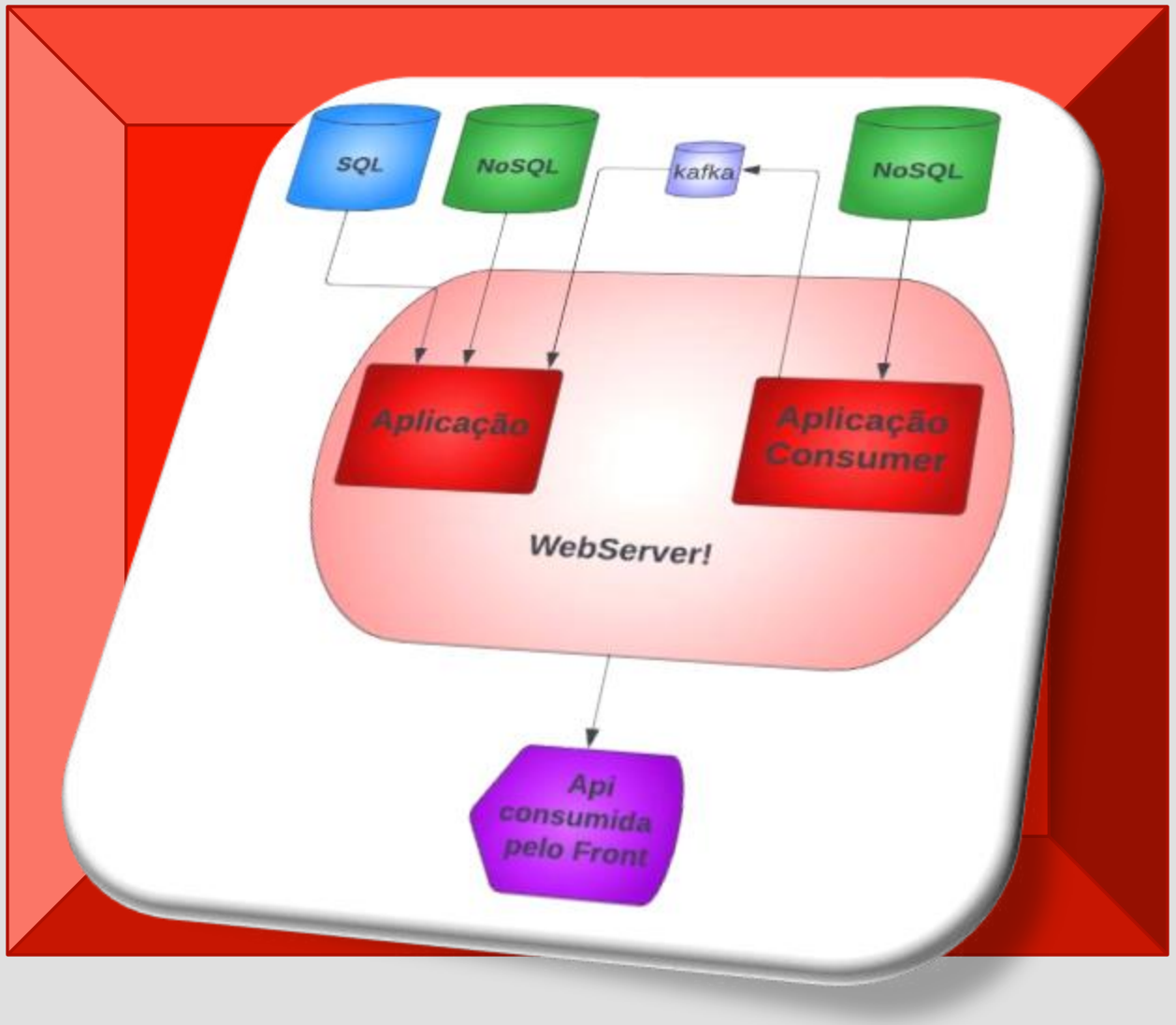
Cinedev!

Trazendo ainda mais novidades! Agora, garantimos um sistema especializado no envio e geração de relatórios de venda e notas fiscais!

CINEDEV!

Através do Kafka realizamos a comunicação entre nossa aplicação original, e uma nova aplicação especializada em monitorar, gerar e enviar notas fiscais, e o relatorio de vendas realizadas no dia!



O Fluxo começa com a compra de Ingressos na aplicação principal (**Cinedev**) que, primeiramente, salva no seu banco SQL a realização da compra, ***settando*** a disponibilidade do ingresso.
Depois, gera um log de compras no banco **NoSQL** da api principal.
Por último, chama o método de produção e envio de notas fiscais, através do sistema **kafka** de mensageria.
Que será consumido pela segunda api, e registrado no banco **NoSQL** da api consumidora.

DO METODO COMPRA DE INGRESSOS E A CHAMADA DO METODO ***PRODUTORSERVICE.ENVIARMENSAGEM!*** EXPLICAÇÃO SEGUE NO PROXIMO SLIDE.

```
blic IngressoCompradoDTO comprarIngresso(Integer idCliente, Integer idIngresso) throws RegraDeNegocioException, Js
    IngressoEntity ingressoRecuperado = findById(idIngresso);

    ClienteEntity clienteRecuperado = clienteService.findById(idCliente);
    ingressoRecuperado.setCliente(clienteRecuperado);
    ingressoRecuperado.setIdCliente(clienteRecuperado.getIdCliente());
    ingressoRecuperado.setDisponibilidade(Disponibilidade.N);
    ingressoRecuperado.setPreco(30.00);
    ingressoRecuperado = ingressoRepository.save(ingressoRecuperado);

    IngressoCompradoDTO ingressoDTO = objectMapper.convertValue(ingressoRecuperado, IngressoCompradoDTO.class);
    UsuarioDTO usuarioDTO = objectMapper.convertValue(clienteRecuperado, UsuarioDTO.class);
    usuarioDTO.setEmail(clienteRecuperado.getUsuario().getEmail());
    ingressoDTO.setNomeCinema(ingressoRecuperado.getCinema().getNome());
    ingressoDTO.setNomeCliente(ingressoRecuperado.getCliente().getPrimeiroNome());
    ingressoDTO.setIdCliente(ingressoRecuperado.getIdCliente());
    ingressoDTO.setNomeFilme(ingressoRecuperado.getFilme().getNome());
    ingressoDTO.setDataHora(LocalDateTime.now());
    emailService.sendEmail(usuarioDTO, TipoEmails.ING_COMPRADO,  token: null);
    ingressoDTO.setPreco(ingressoRecuperado.getPreco());
    produtorService.enviarMensagem(ingressoDTO);
    LogCreateDTO logCreateDTO = new LogCreateDTO(ingressoDTO.getNomeCliente(), TipoLog.INGRESSOS,  LocalDate.now());

    return ingressoDTO;
```

EXPLICANDO OS METODOS!


```
Julio Rocha
public void enviarMensagem(IngressoCompradoDTO ingresso) throws JsonProcessingException {
        NotasFiscaisCinemaDTO notasFiscaisCinemaDTO = new NotasFiscaisCinemaDTO();
        notasFiscaisCinemaDTO.setIdFilme(ingresso.getIdFilme());
        notasFiscaisCinemaDTO.setIdIngresso(ingresso.getIdIngresso());
        notasFiscaisCinemaDTO.setIdCinema(ingresso.getIdCinema());
        notasFiscaisCinemaDTO.setIdCliente(ingresso.getIdCliente());
        notasFiscaisCinemaDTO.setNomeFilme(ingresso.getNomeFilme());
        notasFiscaisCinemaDTO.setNomeCinema(ingresso.getNomeCinema());
        notasFiscaisCinemaDTO.setNomeCliente(ingresso.getNomeCliente());
        notasFiscaisCinemaDTO.setDataHora(ingresso.getDataHora());
        notasFiscaisCinemaDTO.setPreco(ingresso.getPreco());
        notasFiscaisCinemaDTO.setCpf(ingresso.getCpf());
        String msg = objectMapper.writeValueAsString(notasFiscaisCinemaDTO);
```

A COMPRA DE INGRESSOS, COMO MOSTRADO NO SLIDE ANTERIOR LOCALIZA-SE NA API PRINCIPAL,E REALIZA O UPDATE DO STATUS DE DISPONIBILIDADE DO INGRESSO, ASSIM COMO, "SETTA" O VALOR DA COMPRA REALIZADA NO BANCO DE DADOS DA API PRINCIPAL CINEDEV. ASSIM COMO, GERA O LOG DE COMPRAS PARA O BANCO NoSQL DA API PRINCIPAL.

LOGO APÓS, ELE CHAMA O METODO "ENVIAR MENSAGEM" DA PRODUTOR SERVICE, QUE SE ENCARREGA DE CONSTRUIR A NOTA FISCAL E ENVIA-LA O TOPICO DE PARTIÇÃO ÚNICA NO KAFKA(notasfiscais), QUE SERÁ CONSUMIDO PELA SEGUNDA API, QUE É ESPECIALIZADA PELA PERCISTENCIA DAS NOTAS NO MONGODB.

DO METODO RESPONSÁVEL PELO CONSUMO DAS MENSAGENS DO TOPICO


```
        groupId = "{$spring.kafka.consumer.group-id}",
        topicPartitions = {@TopicPartition(topic = "${kafka.topic}", partitions = {"0"})}
)
public void consumirGeral(@Payload String msg) throws JsonProcessingException {
    NotasFiscaisCinemaDTO notaFiscalRecebida = objectMapper.readValue(msg, NotasFiscaisCinemaDTO.cl
    NotaEntity nota = new NotaEntity();
    nota.setNomeCliente(notaFiscalRecebida.getNomeCliente());
    nota.setNomeFilme(notaFiscalRecebida.getNomeFilme());
    nota.setNomeCinema(notaFiscalRecebida.getNomeCinema());
    nota.setIdIngresso(notaFiscalRecebida.getIdIngresso());
    nota.setData(notaFiscalRecebida.getDataHora().minusHours(3));
    nota.setQuantidade(1);
    nota.setPreco(notaFiscalRecebida.getPreco());
    nota.setCpf(notaFiscalRecebida.getCpf());
    notaRepository.save(nota);
}
```

```
@Data
public class NotasFiscaisCinemaDTO {

    private String usuario;
    private String mensagem;
    private Integer idIngresso;
    private Integer idFilme;
    private Integer idCinema;
    private Integer idCliente;
    private String cpf;
    private String nomeCliente;
    private String nomeFilme;
    private String nomeCinema;
    private LocalDateTime dataHora;
    private Disponibilidade disponibilidade;
    private Double preco;
    private String ativo;
```

Este método localizado na aplicação *notascinedevconsumidor.service.ConsumidorService*, se responsabiliza de consumir as mensagens enviadas(para cada compra) na forma NotasFiscaisCinemaDTO, montar a entidade NotaEntity, e realizar o Registro no Banco de Dados NoSQL "NotasFiscaisCinedev".

Envio de Email programado por Schedule!


```java
@Scheduled(cron ="0 0 18 * * *")
public void enviarEmailNotaFiscal(){
    List<NotaEntity> notas = notaRepository.findAllByDataBetween(LocalDateTime.now().minusHours(24), LocalDateTime.now());
    NotaEntity notaFinal = new NotaEntity();
    Double precoFinal = 0.0;
    Integer quantidadeFinal = 0;
    for(int i = 0; i < notas.size(); i++){
        precoFinal += notas.get(i).getPreco();
        quantidadeFinal += notas.get(i).getQuantidade();
    }
    notaFinal.setPreco(precoFinal);
    notaFinal.setQuantidade(quantidadeFinal);
    emailService.sendEmail(notaFinal);
    notas.stream().forEach(System.out::println);
    System.out.println("Julio é lindo");
```

Este método localizado na aplicação ***notascinedevconsumidor.service.AgendamentoService***, é responsável por recuperar da DataBase NotasFiscaisCinedev, usando um find All by Date Between, configurando o período de 24 horas anteriores, para recuperar as notas enviadas neste período. E assim, construir um relatório considerando, a quantidade de vendas e o valor total das vendas. E enviar um email com um Cupom Fiscal, com o fechamento das vendas das umas 24 horas.

DO METODO RESPONSÁVEL PELO CONSUMO DAS MENSAGENS DO TOPICO

Olá Administração!

Segue a seguir, o Cupom fiscal referente ao lucro do dia!

CINEDEV

DATA

CUPOM FISCAL

Ingressos

R$30.00 — Quantidade: ${texto2} — Preço final: R$ ${texto3}

TOTAIS

SUB-TOTAL	R$ ${TEXTO3}
TAXA DE SERVIÇO	10.00%
TOTAL	R$ ${TEXTO4}

CineDev, fazendo o melhor para você!

www.cinedev.com

Qualquer dúvida entre em contato com o suporte pelo email: ${email}.

Segue aqui um exemplo da estrutura do Cupom Fiscal, responsável por apresentar ao administrador o saldo total de vendas das últimas 24 horas.

E PELA ULTIMA VEZ, GOSTARIAMOS DE AGRADECER PELA IMENSA OPORTUNIDADE DE APRESENTARMOS O CINEDEV NO VEMSER!
SENHORES, AS FAVAS.

CINEDEV!

- MATHEUS GONÇALVES
- NOAH BISPO
- JULIO ROCHA