DBC COMPANY

VemSer

TINDER PARA GEEKERS

Apresentado por
Kaio Antônio
Gustavo Linck
Jean Jardim

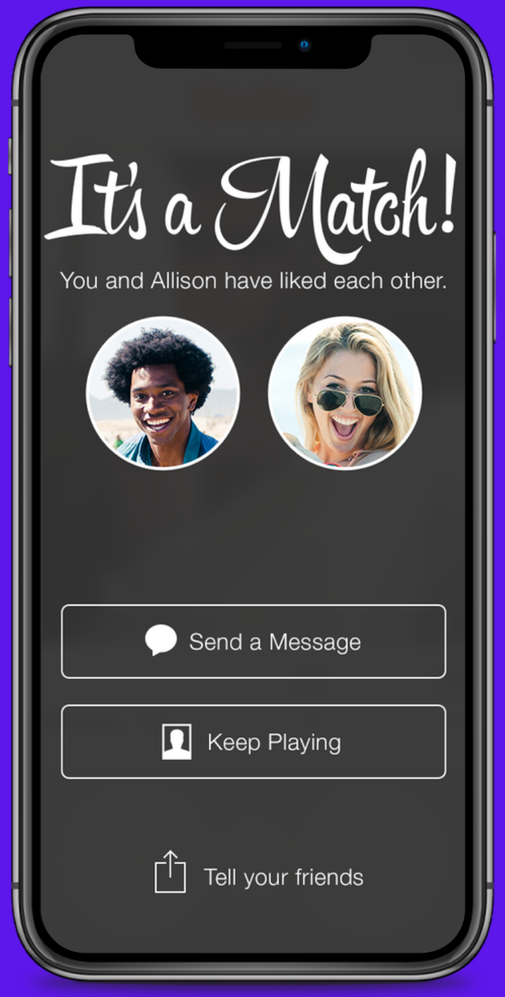
Data
25/11/2022

Ola, seja bem vindo(a) ao nosso projeto Geekers!

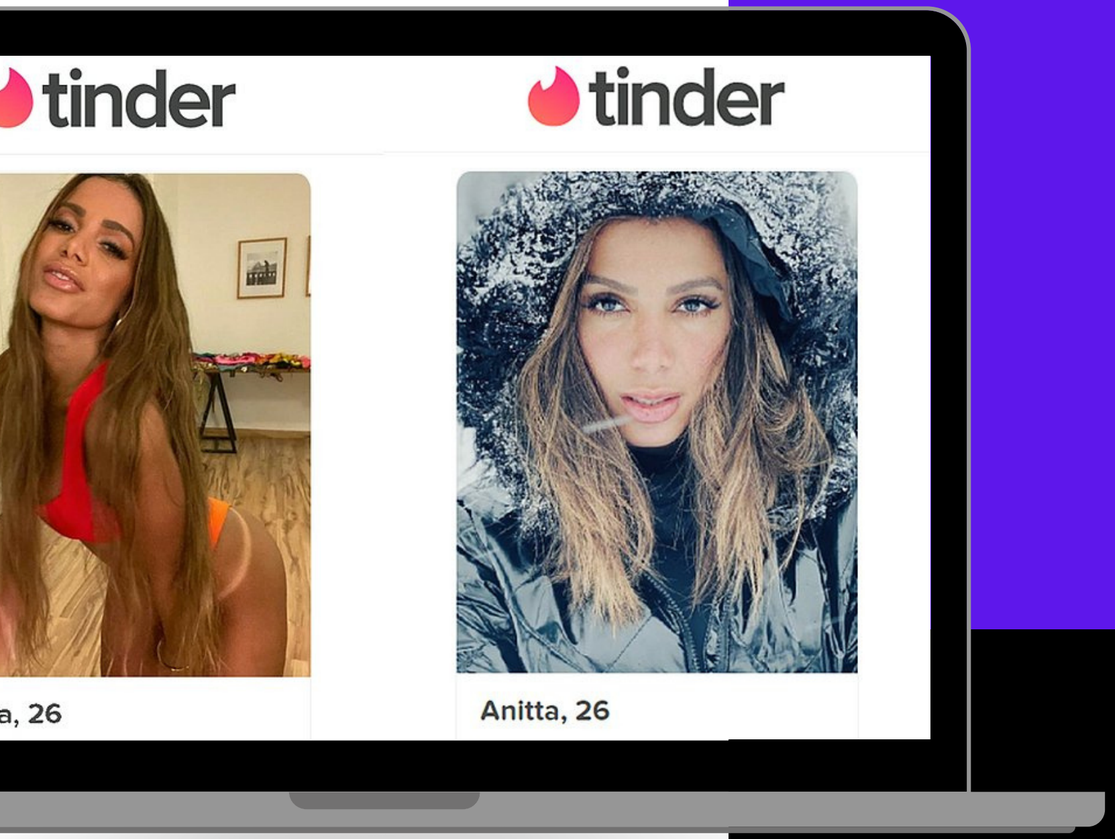
Nossa aplicacao foi feita para voce que tem procurado a sua alma gemea mas nao tem encontrado pois sente que falta aquela conexao com o mundo nerd! Agora com o Geekers voce pode encontrar essa pessoa com os mesmos gostos que voce!

O que tá esperando?
Vamos nessa!

Match apenas após resolver o desafio!



Desafios cadastrados pelo próprio usuário!

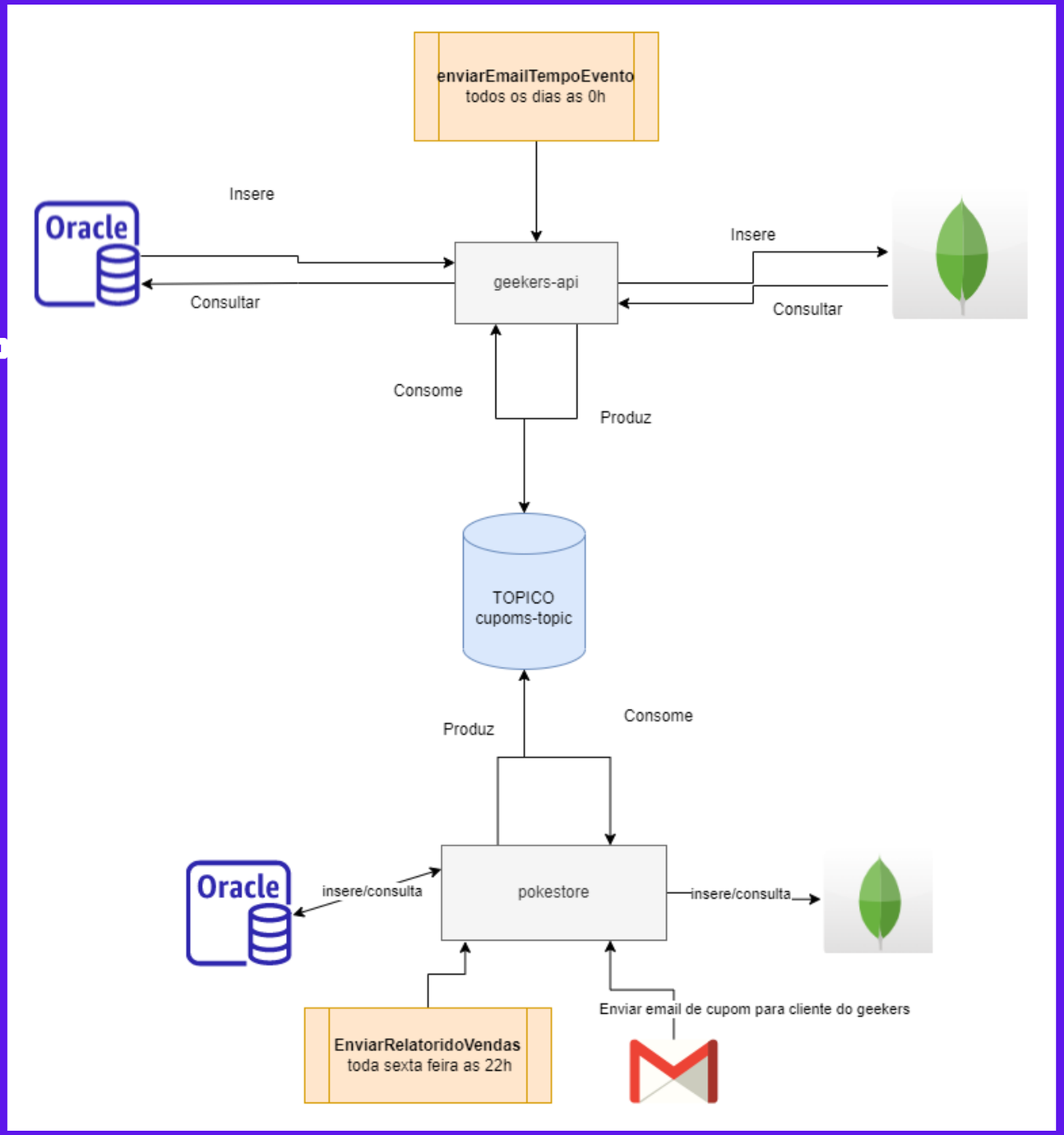


Guardar os dados do contato que você deu match

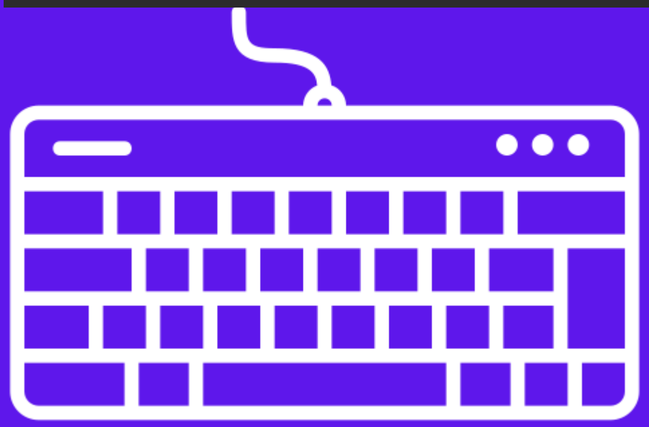
Use o Tinder Geek da maneira mais facil de achar seu amor!

Diagrama de Arquitetura

O fluxo do diagrama foi feito pensando na comunicação das aplicações, ou seja a pessoa-api que possui suas conexões com banco de dados, ela produz uma mensagem para o tópico de cupom para uma partição e consome as mensagens de outra partição enviadas pelo cliente PokeStore.



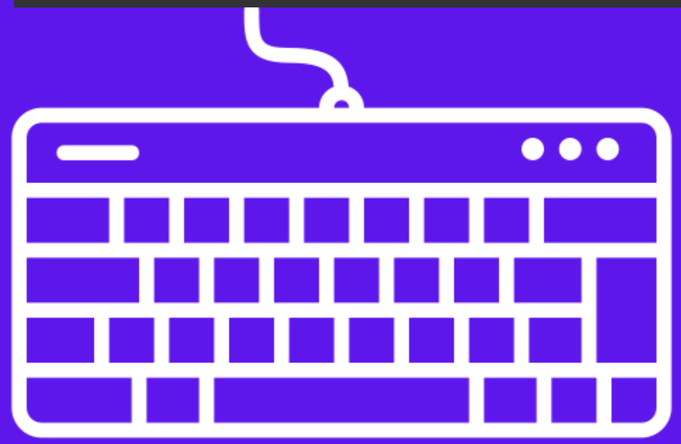
```
1 usage   KaioAntonio
@Scheduled(cron = "0 0 * * * *")
public void enviarEmailTempoEvento() throws RegraDeNegocioException {
    List<UsuarioDTO> usuarios = usuarioService.list();
    for (var usuario : usuarios){
        sendEmail(usuario, usuario2: null, TipoEmail.TEMPO_EVENTO);
    }
}
```



O Scheduled foi utilizado para enviar o email do tempo restante que falta para acabar o evento, é passado no sendEmail o usuario e o tipo do evento que é o template que o email vai usar. Além disso, ele vai enviar o email todos os dias à meia noite.

Esse método faz a alteração do cargo do usuario comum para o usuario Gold,Assim que um usuario realizar uma venda no sistema da pokestore iremos consumir esse cupom.

```
public void consumirCupom(@Payload String cupom,
                          @Header(KafkaHeaders.RECEIVED_MESSAGE_KEY) String key,
                          @Header(KafkaHeaders.RECEIVED_PARTITION_ID) Integer partition,
                          @Header(KafkaHeaders.OFFSET) Long offset) throws JsonProcessingException, RegraDeNegocioException  {
    TopicoCupomDTO topicoCupomDTO = objectMapper.readValue(cupom, TopicoCupomDTO.class) ;
    log.info("CUPOM RECEBIDO -> '{}' Nome -> '{}' Valor -> '{}' -> Data de Vencimento -> '{}' Partition -> '{}' ", topicoCupomDTO.getN
            topicoCupomDTO.getDataVencimento(), partition);
    cupomRepository.save(objectMapper.convertValue(topicoCupomDTO, CupomEntity.class));
    cupomGold(topicoCupomDTO);
}
```



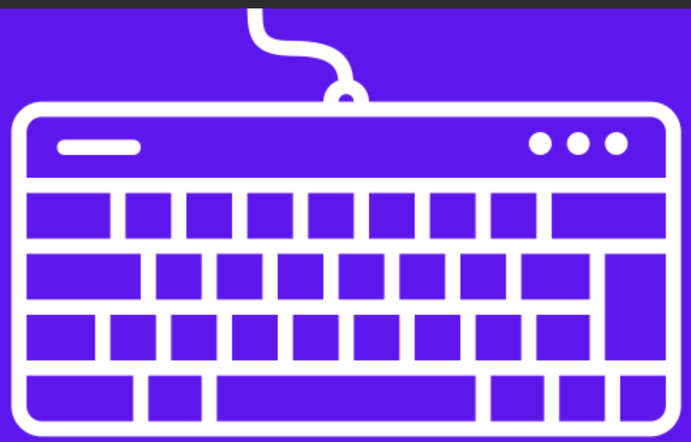
```
public void cupomGold(TopicoCupomDTO topicoCupomDTO) throws RegraDeNegocioException {
    UsuarioEntity usuarioGold = usuarioService.findUserByEmail(topicoCupomDTO.getEmail());
    if(usuarioGold != null){
        usuarioLoginService.atualizarCargo(usuarioGold.getIdUsuario());
    }
}
```

E ele é chamado no metodo de consumir

```java
@Data
public class TopicoCupomDTO extends CupomDTO{
    private String email;
}
```

```java
@Data
public class CupomDTO {
    @NotBlank
    private String nome;
    @NotNull
    private Double preco;
    @NotNull
    private LocalDate DataVencimento;
}
```

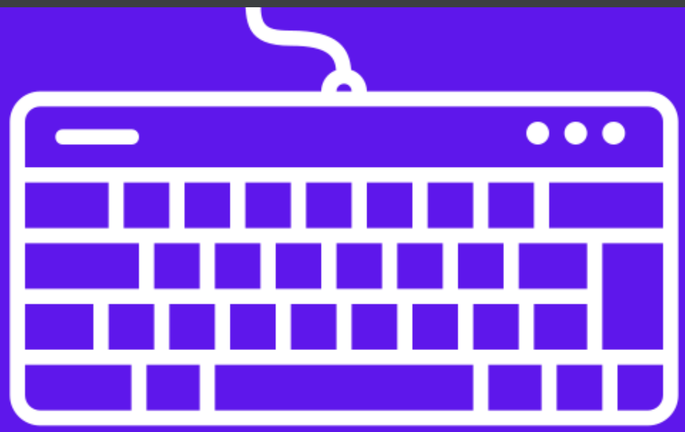
Criamos dois DTO's que foram definidos para não haver conflitos entre as aplicações e assim não houver problemas.



```java
@RequiredArgsConstructor
@Slf4j
@RequestMapping("/cupom")
@RestController
public class CupomController {
    private final ProdutorService produtorService;

    @PostMapping("/send-to")
    public ResponseEntity<Void> sendTo(@RequestBody TopicoCupomDTO topicoCupomDTO) throws JsonProcessingException {
        produtorService.enviarMensagem(topicoCupomDTO);
        return ResponseEntity.noContent().build();
    }
}
```

```java
public void enviarMensagem(TopicoCupomDTO topicoCupomDTO) throws JsonProcessingException {
    String mensagemStr = objectMapper.writeValueAsString(topicoCupomDTO);

        // mensagem, chave, topico
        MessageBuilder<String> stringMessageBuilder = MessageBuilder.withPayload(mensagemStr)
                .setHeader(KafkaHeaders.TOPIC, topic)
                .setHeader(KafkaHeaders.MESSAGE_KEY, UUID.randomUUID().toString())
                .setHeader(KafkaHeaders.PARTITION_ID, PARTICAO) // NÃO É OBRIGATÓRIO TER UM ENUM, PODE SER SOMENTE O NUMERO
                // COMEÇA POR ZERO (0)
                ;

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

        ListenableFuture<SendResult<String, String>> enviadoParaTopico = kafkaTemplate.send(message);
        enviadoParaTopico.addCallback(new ListenableFutureCallback<>() {
            @Override
            public void onSuccess(SendResult result) {
                log.info(" Log enviado para o kafka com o texto: {} ", topicoCupomDTO.getNome());
            }

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

Método enviarMensagem utilizado para a mensagem para o cliente Pokémon e o controller para o usuario com cargo de Admin para criar um cupom.



Principais dificuldades

- Tivemos problemas para comunicar com o cliente pokestore