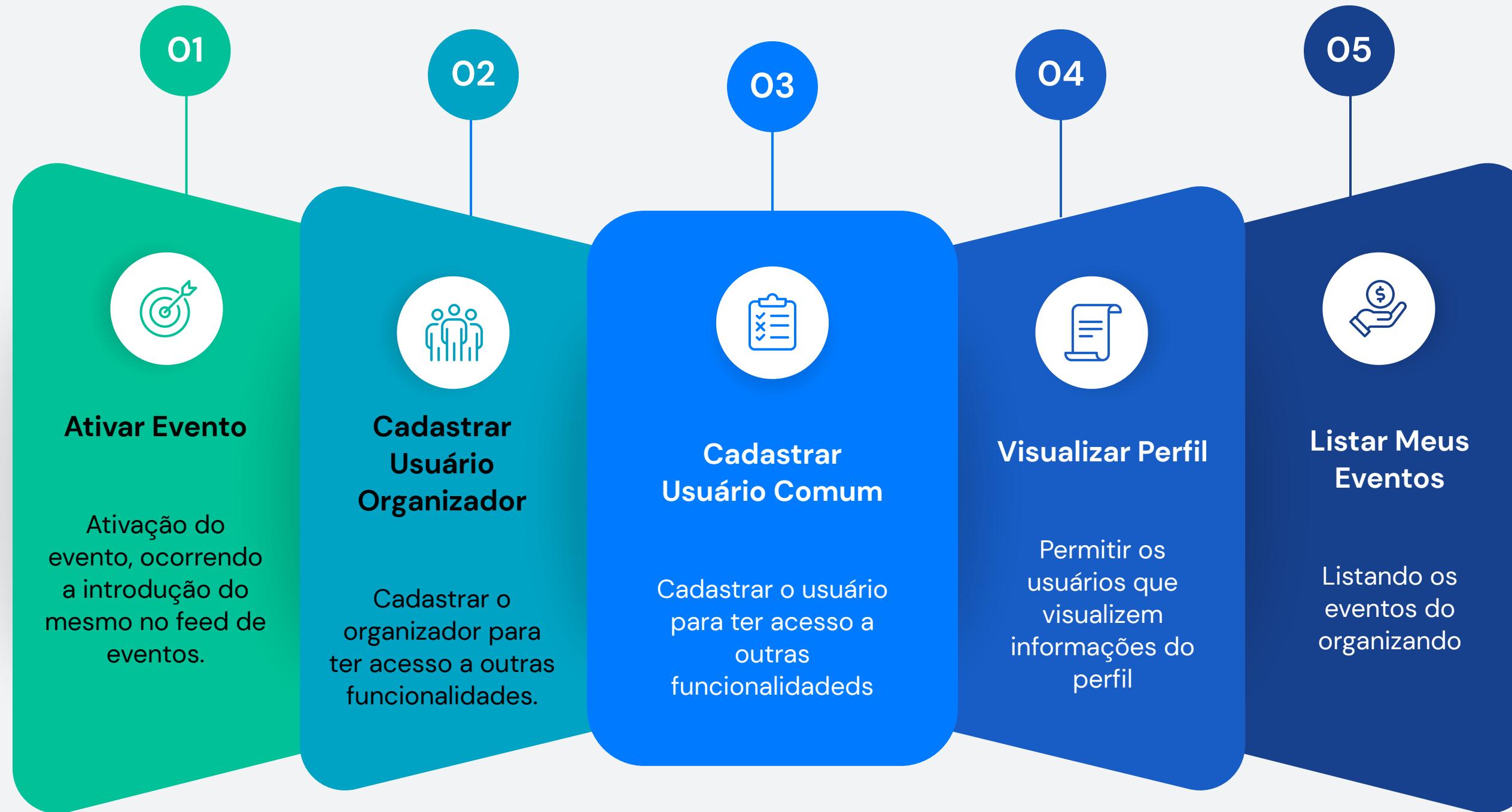


APRESENTAÇÃO  
**DESCOBRINDO**  
**KOTLIN**

ALUNOS:  
RAFAEL JESUS B. CERQUEIRA  
LEVI FERREIRA DE OLIVEIRA BAPTISTA

# FUNCIONALIDADES



# CADASTRAR USUÁRIO

...

```
"1" -> {
    println("===== CADASTRO USUÁRIO =====")
    println("Digite seu nome: ")
    nomeUsuario = readln()

    print("Digite sua data de nascimento (dd/mm/aaaa): ")
    dataNascimentoUsuario = readln()

    println("Digite seu sexo: ")
    sexoUsuario = readln()

    println("Digite seu melhor email: ")
    emailUsuario = readln()

    val emailExiste = listaDeUsuarios.any { usuario ->
        usuario.email == emailUsuario
    }

    when (emailExiste) {
        true -> {
            println("ERRO: Este e-mail já está sendo usado por outra pessoa!")
            continue
        }
        false -> {
            print("Digite a senha: ")
            senhaUsuario = readln()
        }
    }

    val novoUsuario = Usuario(
        nome = nomeUsuario,
        email = emailUsuario,
        senha = senhaUsuario,
        sexo = sexoUsuario,
        dataNascimento = dataNascimentoUsuario
    )

    listaDeUsuarios.add(novoUsuario)
    println("Usuário criado com sucesso!")
}
```

1

Lendo as variáveis para o cadastro do usuário

2

Verificação com .any que percorre o array de objetos procurando a condição estabelecida `usuario.email == emailusuario`, retornando um valor booleano

3

Criando um novo objeto do tipo Usuário e realizando a inserção na lista

```
"2" -> {
    println("===== CADASTRO ORGANIZADOR =====")
    println("Será cadastrado como empresa? (Sim ou Não)")
    val seEmpresa = readln().lowercase()

    when (seEmpresa) {
        "sim" -> {
            println("Digite o CNPJ da empresa:")
            cnpjEmpresa = readln()

            println("Digite a razão social da empresa:")
            razaoSocialEmpresa = readln()

            println("Digite o nome fantasia da empresa:")
            nomeFantasiaEmpresa = readln()
        }

        "não" -> {

        }
        else -> {
            println("Error. Comando inválido!")
            continue
        }
    }

    println("Digite seu nome:")
    nomeUsuario = readln()

    print("Digite sua data de nascimento (dd/mm/aaaa): ")
    dataNascimentoUsuario = readln()

    println("Digite seu sexo:")
    sexoUsuario = readln()

    println("Digite seu melhor email:")
    emailUsuario = readln()

    val emailExiste = listaDeUsuarios.any { usuario ->
        usuario.email == emailUsuario
    }

    if (emailExiste) {
        println("ERRO: Este e-mail já está sendo usado por outra pessoa!")
        continue
    } else {
        print("Digite a senha:")
        senhaUsuario = readln()
    }

    val novoUsuarioOrganizador = UsuarioOrganizador(
        nome = nomeUsuario,
        email = emailUsuario,
        senha = senhaUsuario,
        sexo = sexoUsuario,
        dataNascimento = dataNascimentoUsuario,
        cnpj = cnpjEmpresa,
        razaoSocial = razaoSocialEmpresa,
        nomeFantasia = nomeFantasiaEmpresa
    )

    listaDeUsuarios.add(novoUsuarioOrganizador)

    println("Usuário Organizador criado com sucesso!")
}
```

# CADASTRAR ORGANIZADOR

1

Verificação se o cadastrado será para empresa

2

Lendo as informações do organizador

3

Criando um novo objeto do tipo Usuário Organizador e realizando a inserção na lista

```
"4" -> {  
    println("===== VISUALIZAR PERFIL DO USUÁRIO =====")  
  
    if (listaDeUsuarios.isEmpty()) {  
        println("Nenhum usuário encontrado.")  
        continue  
    }  
  
    println("Digite seu e-mail: ")  
    val validarEmail = readln()  
  
    println("Digite sua senha: ")  
    val validarSenha = readln()  
  
    val validarUsuario = listaDeUsuarios.find { it.email == validarEmail && it.senha ==  
        validarSenha }  
  
    val encontrarUsuario = listaDeUsuarios.filter { it.email == validarEmail }  
  
    if (validarUsuario == null) {  
        println("Usuário não encontrado.")  
        println("Verifique seu e-mail e senha novamente.")  
        continue  
    } else {  
        encontrarUsuario.forEach { usuario ->  
            println("Nome: ${usuario.nome}")  
  
            val separarDataNascimento = usuario.dataNascimento.split("/")  
            val diaNasc = separarDataNascimento[0].toInt()  
            val mesNasc = separarDataNascimento[1].toInt()  
            val anoNasc = separarDataNascimento[2].toInt()  
  
            var idade = 2026 - anoNasc  
  
            println("Data de Nascimento: ${usuario.dataNascimento} (Idade:%d)".format(idade))  
            println("E-mail: ${usuario.email}")  
  
            if (usuario is UsuarioOrganizador) {  
                println("CNPJ da empresa: ${usuario.cnpj}")  
                println("Razão social da empresa: ${usuario.razaoSocial}")  
                println("Nome fantasia da empresa: ${usuario.nomeFantasia}")  
            }  
        }  
    }  
}
```

# VISUALIZAR PERFIL

1

Validando usuário e senha com `.find` e resgatando o objeto usuário através do `.filter`

2

Utilizando um laço `forEach` para imprimir as informações do usuário.

3

Separando data de nascimento através do `.split` e realocando em 3 variáveis , assim realizando a conta da idade

```
"9" -> {
    println("===== ATIVAR EVENTO =====")

    if (listaDeEventos.isEmpty()) {
        println("Nenhum usuário encontrado.")
        println("Cadastre um usuário, que seja organizador, antes de cadastrar um evento.")
        continue
    }

    println("Digite seu e-mail: ")
    val validarEmail = readln()

    println("Digite sua senha: ")
    val validarSenha = readln()

    val validarUsuario = listaDeUsuarios.find { it.email == validarEmail && it.senha == validarSenha }

    when (validarUsuario) {
        null -> {
            println("Usuário não encontrado.")
            println("Verifique seu e-mail e senha novamente.")
            continue
        }
        is UsuarioOrganizador -> {
            val listarEventos = listaDeEventos.filter { it.emailOrganizador == validarEmail }

            var i = 1
            listarEventos.forEach { evento ->
                print("Evento ${i}")
                println("Nome: ${evento.nomeEvento}")
                println("Data: ${evento.periodoRealizacao}")
                println("Local: ${evento.localEvento}")
                println("Preço Unitário: ${"%2f".format(evento.precoUnitarioIngresso})")
                println("Capacidade máxima de pessoas: ${evento.capacidadeMaximaPessoas}")
                println("-----")
                i++
            }

            println("Digite o nome do evento que deseja ativar: ")
            val escolhaEvento = readln()

            val encontrarEvento = listaDeEventos.find { it.nomeEvento == escolhaEvento }

            if (encontrarEvento != null) {
                when (encontrarEvento.estaAtivo) {
                    true -> {
                        println("O evento já está no feed eventos")
                    }
                    false -> {
                        listaFeedEventos.add(encontrarEvento)
                        encontrarEvento.estaAtivo = true
                        println("Evento adicionado no feed com sucesso!")
                    }
                }
            } else {
                println("Não pode ser vazio.")
                continue
            }
        }
        else -> {
            println("Torne-se organizador para poder criar eventos.")
            continue
        }
    }
}
```

# ATIVAR EVENTOS

1

Condição para verificar se o usuário é comum ou organizador

2

Buscando o evento através do .find e realizando uma condição caso o valor seja diferente de vazio

3

Condição when para verificar se o evento já está ativo, caso não esteja será inserido na listaFeedEventos e atualizando o valor booleando de estaAtivo para true

# LISTAR MEUS EVENTOS

```
"11" -> {
    println("===== LISTAR EVENTOS =====")

    if (listaDeEventos.isEmpty()) {
        println("Nenhum evento encontrado.")
        continue
    }

    println("Digite seu e-mail: ")
    val validarEmail = readln()

    println("Digite sua senha: ")
    val validarSenha = readln()

    val validarUsuario = listaDeUsuarios.find { it.email == validarEmail && it.senha == validarSenha }

    val listarEventos = listaDeEventos.filter { it.emailOrganizador == validarEmail }

    when (validarUsuario) {
        null -> {
            println("Usuário não encontrado.")
            println("Verifique seu e-mail e senha novamente.")
            continue
        }

        is UsuarioOrganizador -> {
            println("--- LISTA DE EVENTOS ---")

            listarEventos.forEach { evento ->
                println("Nome: ${evento.nomeEvento}")
                println("Data: ${evento.periodoRealizacao}")
                println("Local: ${evento.localEvento}")
                println("Preço Unitário: ${"%2f".format(evento.precoUnitarioIngresso})")
                println("Capacidade máxima de pessoas: ${evento.capacidadeMaximaPessoas}")
                println("-----")
            }
        }

        else -> {
            println("Torne-se organizador para poder criar eventos e listá-los.")
        }
    }
}
```

1

Verificação de usuário

2

Resgatando um listarEventos com  
.filter

3

Verificação do tipo de usuário e laço  
de repetição para imprimir as  
informações

**OBRIGADO!!!**