QtCon BR

TELEPATHY+QT
APLICATIVOS DE MENSAGENS

Gustavo Boiko – SUSE
gboiko@suse.com

OLÁ!

Quem é Gustavo Boiko?

1.

SOBRE O QUE VAMOS CONVERSAR?

SERVIÇOS DE VOZ, VÍDEO E MENSAGENS CADA VEZ MAIS POPULARES







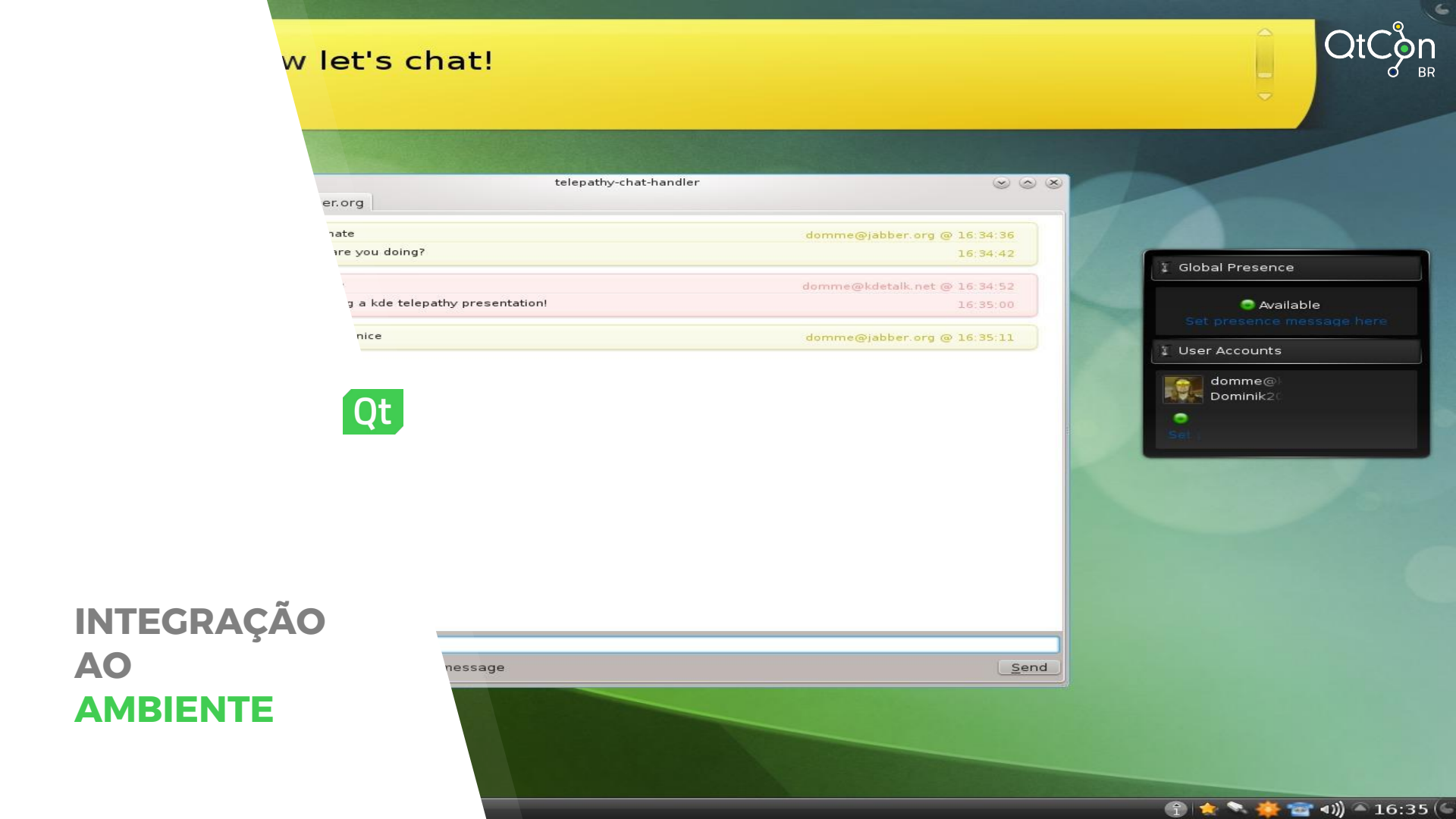







APLICATIVOS INTEGRANDO MÚLTIPLOS SERVIÇOS

- Cada desktop tem um
- Experiência unificada
- Integrado ao ambiente



INTEGRAÇÃO AO AMBIENTE

AÍ COMEÇAM OS PROBLEMAS!!!

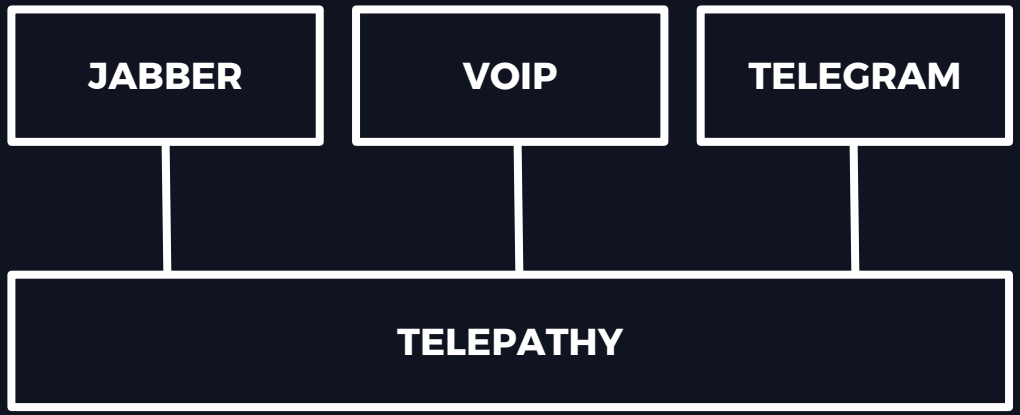
- Serviços com API e funcionalidades diferentes
- Cada aplicativo precisa reimplementar tudo
- Atualizações nos serviços “quebram” e causam inconsistências

Qt

TELEPATHY!!

API **UNIFICADA**
PARA ACESSAR
SERVIÇOS

Qt



PRINCIPAIS ASPECTOS

- Design modular
- Fácil substituição de componentes
- Divisão de tarefas
- Componentes isolados

VISÃO GERAL

OS PROTOCOLOS

- Serviços são definidos como Protocolos
- Instâncias do protocolo são Contas
- Capacidades gerais
- Capacidades extras por Conta

Observadores

Tratadores

Aprovadores

OS CLIENTES

Observadores

- “read-only”
- Histórico
- Indicador
- Notificação em Tela

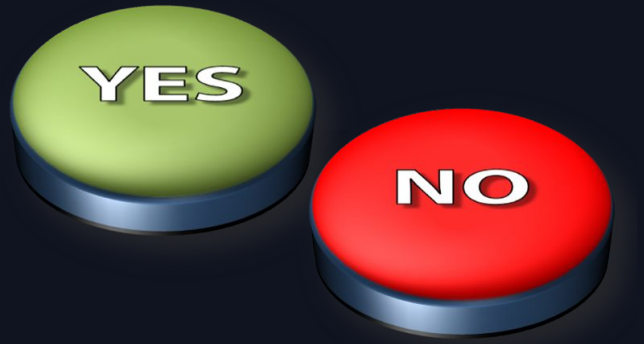
OS CLIENTES

Qt

Tratadores

- Interagem com serviços
- Janela de chat
- Janela de vídeo
- Lista de contatos

OS CLIENTES



Qt

Aprovadores

OS CLIENTES

MISSION CONTROL

TELEPATHY QT

- Telepathy é definido na camada IPC (DBus)
- Converte a API do Telepathy em algo Qt-friendly
- Operações assíncronas
- Introspecção seletiva
- Classes para tarefas simples

CHAMADAS ASSÍNCRONAS

- Chamadas baseadas em PendingObjects, exemplo:

```
PendingChannelReady *pr;
pr = account->ensureTextChat("myfriend@suse.com");

connect(pr, &Tp::PendingObject::finished,
            this, &onTextChatRequestFinished);

// controle retorna para o loop de eventos
```

INTROSPECÇÃO SELETIVA

- Prepara objetos para o uso baseado em “Features”:

```
callChannel->becomeReady(FeatureCallState |
                         FeatureCallMembers);
```

- Evita tráfego desnecessário entre processos

CLASSES PARA TAREFAS SIMPLES

Exemplos:

- Tp::SimpleTextObserver

- Tp::ContactMessenger

EXEMPLO: PASSOS PARA IMPLEMENTAR UM OBSERVADOR

- Preparar recursos do Tp::AccountManager
- Criar uma classe herdando de Tp::AbstractClientObserver
- Reimplementar o método observeChannels()
- Informar a lista de filtros de canais a observar
- Registrar o observador com Tp::ClientRegistrar::registerClient()

QtCon BR

OBRIGADO!

Perguntas?

Gustavo Boiko @gustavoboiko &
gustavo.boiko@gmail.com

Qt