O componente CloudDB é um componente invisível que permite armazenar dados em um servidor de banco de dados conectado à Internet (usando o software Redis).

Isso permite que os usuários do seu aplicativo compartilhem dados uns com os outros. Por padrão, os dados serão armazenados em um servidor mantido pelo MIT.

Este aplicativo é um aplicativo de chat simples que usa o componente CloudDB para armazenar as mensagens do usuário na nuvem e permite que vários usuários do aplicativo se comuniquem em vários dispositivos.

Design:

- 1 legenda de título;
- 1 organização horizontal com uma caixa de texto e um botão;
- 1 organização horizontal com um botão e uma legenda escondida;
- 1 visualizador de listas;

Componentes invisíveis de notificador e cloudDB;



Código:

Inicialize três variáveis:

NomeUsuario: armazenado quando o usuário faz login

entrar: booleano que é verdadeiro quando o usuário faz login

chatList: armazena todas as mensagens de chat em uma lista

Quando o usuário pressiona o botão Enviar, ele está inserindo seu nome para entrar no batepapo ou digitando uma mensagem de bate-papo. Você verificará primeiro se eles já entraram no chat ou não.

```
🔯 se
           não
                  obter global entrar
então
       ajustar global nome para TextBox1
                                                 Texto ▼
        ajustar Label2 . Texto
                                           🗯 juntar
                                  para
                                                        Olá ]
                                                       obter global nome *
                                                       " 🚺 "
        ajustar global entrar 🔻 para 📗 verdadeiro 🔻
       ajustar Label1
                          Texto ▼
                                   para
                                            Escreva a sua mensagem:
```

Caso contrário, Você salvará essas mensagens no CloudDB.

```
senão chamar CloudDB1 v .AppendValueToList rótulo ( batepapo " itemToAdd  juntar ( obter global nome v itemToAdd  itemToA
```

Quando o aplicativo é iniciado pela primeira vez, você precisa verificar se já existem mensagens de bate-papo enviadas por outros usuários e exibi-las.

```
quando Screen1 v .Inicializar
fazer chamar CloudDB1 v .ObterValor
rótulo valorSeRótuloNãoExistir criar lista vazia
```

Depois que você obtém o valor inicial, o CloudBD solicita o valor atual do seu rótulo. Portanto, há um evento de retorno de chamada que é acionado quando ele responde ao ObterValor.

```
quando CloudDB1 v .RecebeuValor
rótulo value
fazer se obter rótulo v = v " batepapo "
então ajustar global chatlist v para obter value v
ajustar ListView1 v . Elementos v para obter global chatlist v
```

O evento CloudDB1.DataChanged é acionado sempre que alguém está usando o aplicativo e atualiza o CloudDB, então, esse evento é acionado sempre que uma nova mensagem é enviada por alguém. Esse bloco faz a verificação coisa que o bloco ObterValor.

```
quando CloudDB1 v .DataChanged
rótulo value

fazer se obter rótulo v = v " batepapo "
então ajustar global chatlist v para obter value v
ajustar ListView1 v . Elementos v para obter global chatlist v
```

Para apagar as mensagens do visualizador de listas vamos usar o apagar rótulo

```
quando Button2 · .Clique

fazer chamar CloudDB1 · .ApagarRótulo

rótulo "batepapo"

ajustar ListView1 · . Elementos · para criar lista vazia
```

A melhor maneira de testar um aplicativo usando o CloudDB e vários usuários é usar vários dispositivos. O MIT AI2 Companion nem sempre permite que vários usuários se conectem ao mesmo tempo, portanto, a melhor opção é criar o apk e instalar seu aplicativo em mais de um dispositivo móvel.

https://youtu.be/VOxhMYnKyhl