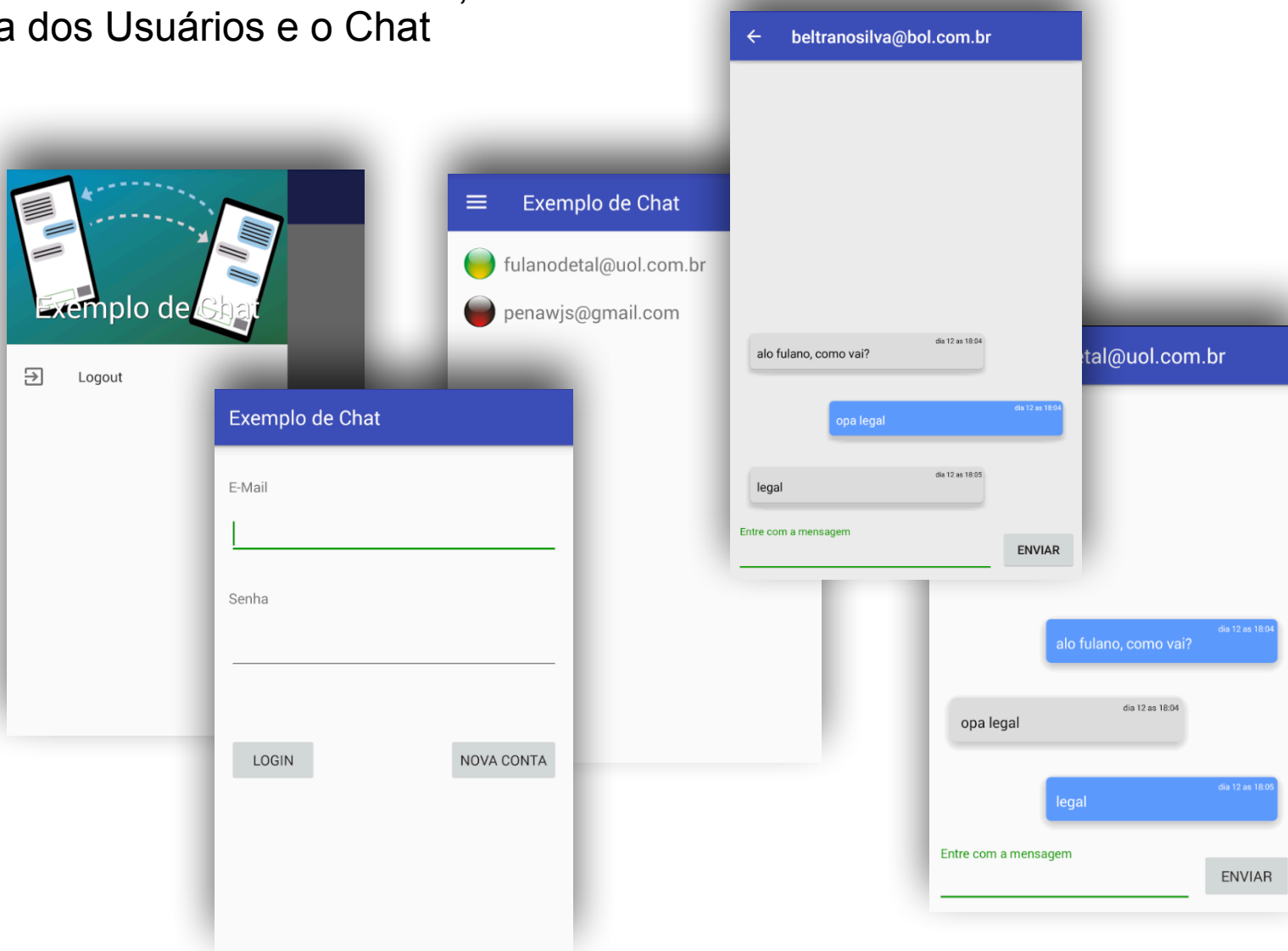




Chat

# A Aplicação

A aplicação consistem em três activities, o Login, a Lista dos Usuários e o Chat



# Estrutura de Dados

A estrutura foi adequada para possibilitar que cada usuário tenha acesso a todas as mensagens trocadas.

É necessário selecionar um usuário para iniciar o chat, com isto a aplicação ontem a identificação de cada um para montar a relação **usuario -> destinatário** onde serão armazenadas as mensagens.

Sempre que uma mensagem é enviada, esta é gravada na estrutura de dados para ambos os usuários que participam do chat.

exemploautenticacao-c2c24

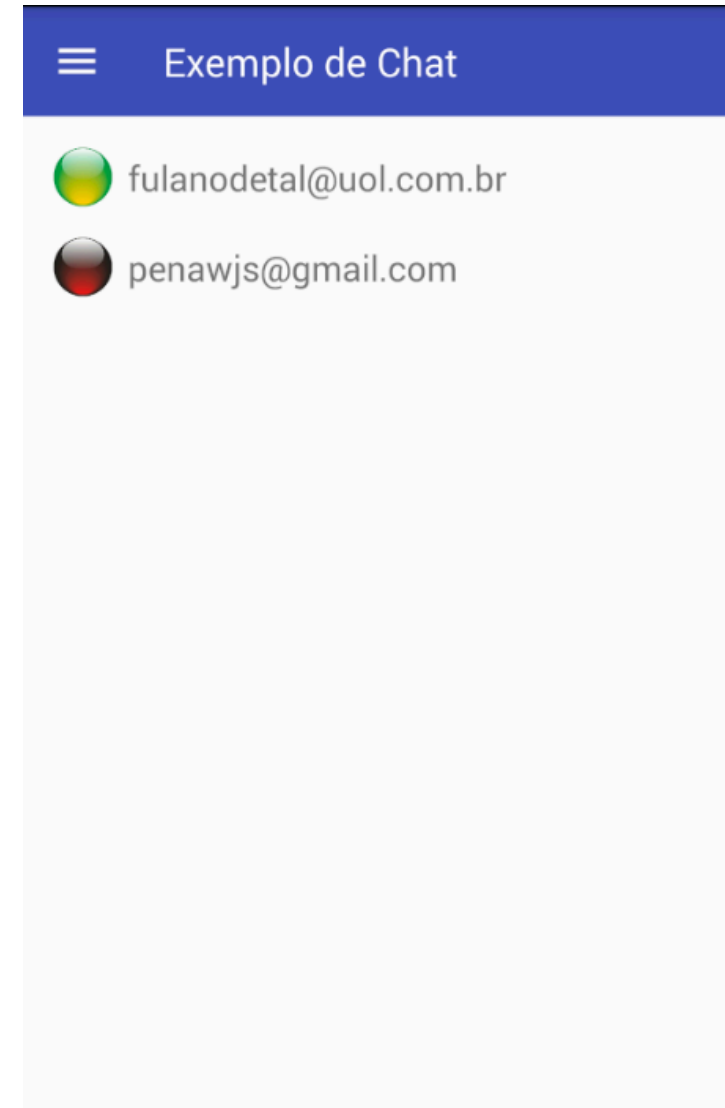


# A Lista dos Usuários

Na Activity que lista os usuários temos um *ListView* onde devem aparecer todos os usuários exceto o que fez login no Android.

Para obtermos a lista dos usuários deveremos acessar a referência do Firebase que aponta para os usuários.

O Firebase provê o Queries para efetuar consultas avançadas, porém não encontramos uma forma de criar uma expressão que negue (NOT) uma restrição.



# A Lista dos Usuários

A fim de suprir esta deficiência, foi construída a classe `UsuarioChatArray` que implementa o mecanismo que filtra os usuários que serão listados com base no UID do usuário logado.

```
public class UsuarioChatArray extends FirebaseArray<Usuario> {
    private UsuarioDao dao = UsuarioDao.dao;
    private String chavePreviaAdicionada;

    public UsuarioChatArray() {
        super(UsuarioDao.dao.getReference().orderByChild("email"),
            new SnapshotParser<Usuario>() {
                public Usuario parseSnapshot(@NonNull DataSnapshot snapshot) {
                    return snapshot.getValue(Usuario.class);
                }
            }
        );
    }

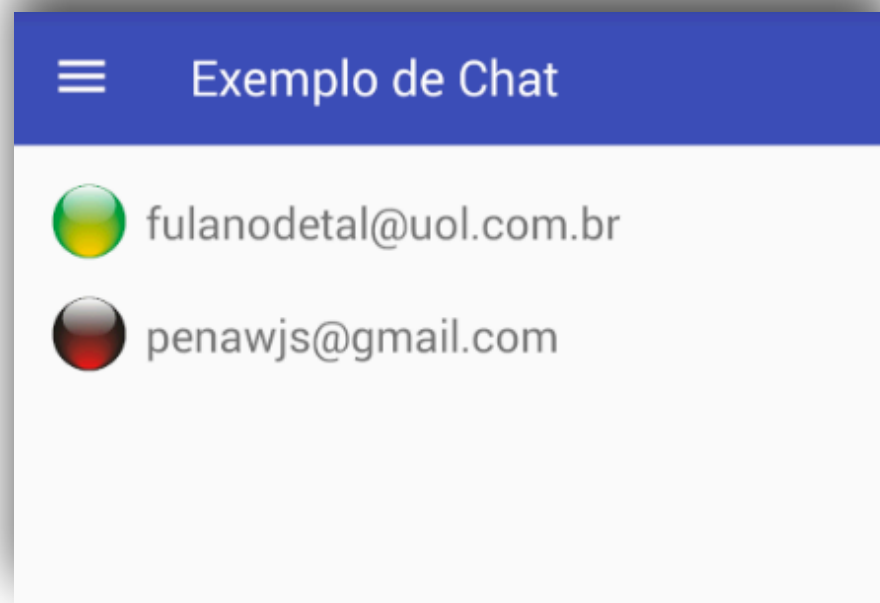
    public void onChildAdded(DataSnapshot snapshot, String previousChildKey) {
        if(previousChildKey == null) chavePreviaAdicionada = null;

        if(!snapshot.getKey().equals(dao.getUserId())) {
            super.onChildAdded(snapshot, chavePreviaAdicionada);
            chavePreviaAdicionada = snapshot.getKey();
        }
    }
}
```

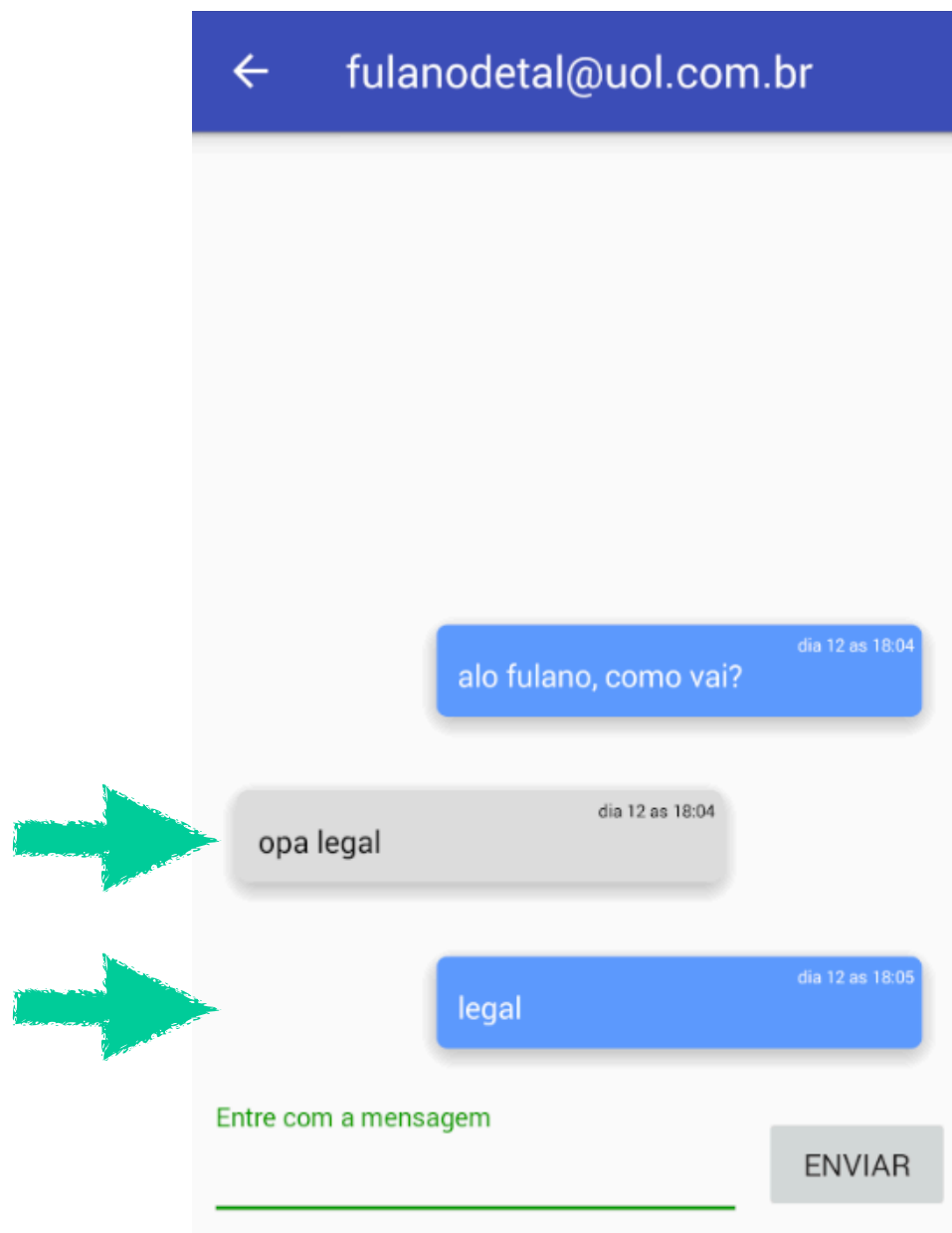
# A Lista dos Usuários

A classe **UsuarioChatArray** é passado para o **FirebaseListOptions** utilizar como mecanismo de construção da lista a ser apresentada.

```
if(adapter == null) {  
    FirebaseListOptions<Usuario> options = new FirebaseListOptions.Builder<Usuario>()  
        .setSnapshotArray(new UsuarioChatArray())  
        .setLayout(R.layout.layout_usuario)  
        .setLifecycleOwner(this)  
        .build();  
  
    adapter = new UsuarioAdapter(options);  
  
    listView.setAdapter(adapter);  
}
```



A activity do Chat é necessário diferenciar as mensagem enviadas das recebidas, para isto necessitamos de dois layouts utilizados no detalhe do ListView




Quando utilizamos o Firebase necessitamos utilizar o `FirebaseListOptions` para a construção dos requisitos de acesso e parametrização do Adapter.

O Adapter é responsável pela construção do detalhe, neste caso das mensagens.

Esta técnica nos oferece a opção de apresentar um único layout.

```
if(adapter == null) {  
    FirebaseListOptions<Mensagem> options = new FirebaseListOptions.Builder<Mensagem>()  
        .setQuery(dao.getReference(), Mensagem.class)  
        .setLayout(R.layout.layout_mensagem)  
        .setLifecycleOwner(this)  
        .build();  
  
    adapter = new MensagemAdapter(options);  
  
    listView.setAdapter(adapter);  
}
```





Para construir o layout adequado dependendo do sentido da mensagem, foi utilizado o método **getView()** herdado da classe **FirestoreListAdapter**.

Neste método é possível determinar qual o layout adequado antes de sua apresentação.

```
public class MensagemAdapter extends FirestoreListAdapter<Mensagem> {
    @SuppressWarnings("SimpleDateFormat")
    private SimpleDateFormat fmt = new SimpleDateFormat( pattern: "'dia' dd 'as' HH:mm");
    public MensagemAdapter(@NonNull FirestoreListOptions<Mensagem> options) {...}

    @Override
    public View getView(int position, View view, ViewGroup viewGroup) {
        Mensagem model = getItem(position);


        if(destinatarioId.equals(model.getOrigem())) {
            view = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.layout_mensagem2, viewGroup, attachToRoot: false);
        } else {
            view = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.layout_mensagem1, viewGroup, attachToRoot: false);
        }

        return super.getView(position, view, viewGroup);
    }

    @Override
    protected void populateView(View view, Mensagem model, int position) {
        hideProgressDialog();

        TextView tvMsg = view.findViewById(R.id.tvUsuario);
        TextView tvData = view.findViewById(R.id.tvData);

        tvMsg.setText(model.getMensagem());
        tvData.setText(fmt.format(new Date(model.getData())));
    }
}
```



FIM