INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: FIREBASE REALTIME DATABASE

Aula Prática 06

Parte 1: Preparação para usar o Firebase Realtime Database

Passo 1: Essa prática é uma continuação da Prática 05. A preparação do Firebase e da aplicação devem seguir as orientações nela contidas.

Passo 2: Adicione suporte ao Firebase Realtime Database, acionando novamente Tools->Firebase.

> Selecione a opção “Realtime Database” e depois “Save and retrieve data”. Execute o passo 2 nessa nova aba (o 1º passo já deve estar feito). Serão adicionadas as dependências necessárias ao script gradle da aplicação. O 3º passa explica sobre o controle de acesso ao banco de dados. Manteremos o acesso restrito a usuário logados. Os demais passos são trechos de código para manipulação dos dados.

Passo 3: Visualizando a base de dados no Firebase Console:

O Firebase Realtime Database salva os dados na forma de uma árvore JSON. Não há tabelas ou registros, portando deve haver um cuidado especial em como modelar os dados da sua aplicação.

Nesta prática será construída uma aplicação de chat simples. Para essa aplicação, a estrutura de dados usada (como vista no console) é mostrada na figura ao lado.

A recomendação geral é evitar que a árvore de dados seja muito profunda (vários níveis), isto é, fazer com que os dados sejam “planos”. Pela própria natureza da árvore, os dados estarão desnormalizados, isto é, haverá duplicação/redundância (na figura, o nome do usuário aparece no chat e no registro de usuários). Cabe ao programador manter a consistência.

Parte 2: Preparando o Model

Passo 1: Crie a classe `User` com o código a seguir, no pacote "`model`":

```
@IgnoreExtraProperties
public class User {
    private String name;
    private String email;

    public User() {}

    public User(String name, String email) {
        this.name = name;
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }
}
```

Essa classe modela o nosso usuário. O contrustor vazio e gets() públicos são necessário para usar os métodos do Firebase que salvam (e recuperam) objetos direto no banco.

Passo 2: Crie a classe `Message` com o código a seguir, também no pacote "`model`":

```
@IgnoreExtraProperties
public class Message {
    private final String name;
    private final String text;

    public Message() {
        this.name = null;
        this.text = null;
    }

    public Message(String name, String text) {
        this.name = name;
        this.text = text;
    }

    public String getName() {
        return name;
    }

    public String getText() {
        return text;
    }
}
```

Essa classe modela uma mensagem enviada por um usuário no chat.

Parte 3: Registrando o usuário na base

Passo 1: Em `SignUpActivity`, modifique o método `buttonSignUpClick()`, adicionando o código em destaque abaixo:

```
public void buttonSignUpClick(View view) {
     final String name = edName.getText().toString();
     final String email = edEmail.getText().toString();
     final String password = edPassword.getText().toString();

     final FirebaseAuth mAuth = FirebaseAuth.getInstance();
     mAuth.createUserWithEmailAndPassword(email, password)
             .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                 @Override
                 public void onComplete(@NonNull Task<AuthResult> task) {
                     String msg = task.isSuccessful() ? "SIGN UP OK!":
                                                        "SIGN UP ERROR!";
                     Toast.makeText(SignUpActivity.this, msg,
                                    Toast.LENGTH_SHORT).show();

                     if (task.isSuccessful()) {
                         User tempUser = new User(name, email);
                         DatabaseReference drUsers = FirebaseDatabase.
                                                     getInstance().getReference("users");
                         drUsers.child(mAuth.getCurrentUser().getUid()).
                                                     setValue(tempUser);
                     }
                 }
             });
 }
```

Na Prática 05 foi feito o registro do novo usuário na base de autenticação do Firebase (usando `createUserWithEmailAndPassword()`), que é uma base a parte contendo apenas e-mail e senha (criptografada) do usuário.

Nesse passo adicionamos o trecho de código que cria o usuário na nossa base. O processo sempre começa a pegando uma referência (`DatabaseReference`) para um ponto específico da árvore através de um endereço. No nosso caso o endereço é apenas “user”, mas poderia ser complexo como “user/12345/messages”, onde “12345” é o identificador do usuário e “messages” seria um dos filhos (obs.: exemplo ilustrativo, esse campo não existe nessa prática).

Ao usar `child()`, criamos um novo filho nesse ponto da árvore, nesse caso com o Id gerado para novo usuário (que está logado). Usamos `setValue()` para salvar nesse filho o novo usuário que criamos. Serão criados filhos nesse ponto de acordo com as propriedades do objeto `tempUser`.

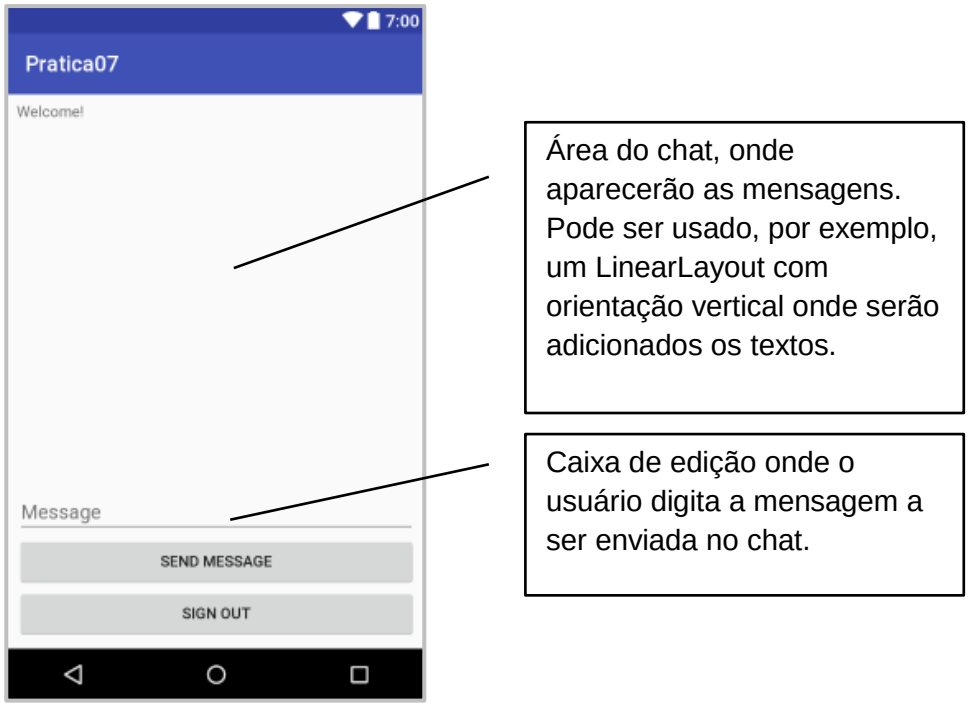
Passo 2: Rode e teste a aplicação, adicionando um novo usuário.

Passo 3: Verifique no Firebase se o usuário foi criado adequadamente.

Os usuários registrados anteriormente estarão presentes na base de autenticação do Firebase mas não na base de dados. Perceba que ao logar como um desses usuários os seus dados (no caso, o nome) não estarão disponíveis.

Parte 4: Preparando a interface do Chat

Passo 1: Modifique a interface de `HomeActivity`, de forma a se assemelhar a figura abaixo:



Passo 2: Crie o método auxiliar `showMessage()` em `HomeActivity`, como mostrado abaixo:

```
private void showMessage(Message message) {
     TextView tvMsg = new TextView(this);
     tvMsg.setText(message.getName() + ": " + message.getText());
     vgChat.addView(tvMsg);
 }
```

Esse método cria um `TextView` com a mensagem (Objeto `message`) e coloca na área de chat (objeto `vgChat`). O objeto `vgChat` é uma referência do tipo `ViewGroup` inicializada no `onCreate()` de `HomeActivity` que aponta para área de chat da figura acima. Modifique seu código de acordo.

Passo 3: Para testar a interface, associe um tratador temporário ao botão “Send Message” que utiliza o método acima para exibir a mensagem digitada pelo usuário na caixa de edição.

Passo 4: Rode e teste a aplicação.

Parte 5: Conectando a interface com o Firebase

Passo 1: Adicione o trecho em destaque ao final de `onCreate()` do `HomeActivity`:

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        FirebaseDatabase fbDB = FirebaseDatabase.getInstance();
        FirebaseUser fbUser = mAuth.getCurrentUser();
        drUser = fbDB.getReference("users/" + fbUser.getUid());
        drChat = fbDB.getReference("chat");

        drUser.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                User tempUser = dataSnapshot.getValue(User.class);
                if (tempUser != null) {
                    HomeActivity.this.user = tempUser;
                    txWelcome.setText("Welcome " + tempUser.getName() + "!");
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {  }
        });

        drChat.addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                Message message = dataSnapshot.getValue(Message.class);
                showMessage(message);
            }

            @Override
            public void onChildChanged(DataSnapshot dataSnapshot, String s) {  }
            @Override
            public void onChildRemoved(DataSnapshot dataSnapshot) {  }
            @Override
            public void onChildMoved(DataSnapshot dataSnapshot, String s) {  }
            @Override
            public void onCancelled(DatabaseError databaseError) { }
        });
    }
```

`drUser` e `drChat` são propriedades de `HomeActivity` do tipo `DatabaseReference`, apontando respectivamente para os ramo do usuário logado e o chat da árvore. Nesse método são registrados tratadores que capturam as mudanças nos dados e atualizam a UI quando estas acontecem.

Passo 2: Faça o tratador do botão “Send Message” ter o seguinte código:

```
public void buttonSendMsgClick(View view) {
    String message = edMessage.getText().toString();
    edMessage.setText("");

    drChat.push().setValue(new Message(user.getName(), message));
}
```

`edMessage` é o `EditText` onde o usuário digita a mensagem. O método `push()` cria uma folha no chat e `setValue()` joga os valores da nova mensagem.

Passo 3: Rode e teste a aplicação digitando mensagens. Experimente sair da aplicação e logar como outro usuário. Veja no Firebase as novas mensagens criadas.

Parte 6: (Desafio) Aprimoramentos

Passo 1: Colocar uma ActionBar (`ToolBar`) com menu e colocar a opção de “Sign Out” nele, oculta. Remover o botão de “Sign Out” para evitar toques acidentais.

Passo 2: Colocar no menu uma opção para cancelar assinatura (“Delete Account”). Para implementar esse método:

- Use `drUser.removeValue()` para remover da base.
- Use `mAuth.getCurrentUser().delete()` para remover as credenciais de autenticação do usuário.

Passo 3: Permita ao usuário modificar seus dados.

- Modifique `SignUpActivity` para funcionar também como página de edição de dados.
- Para atualizar o email, use: `mAuth.getCurrentUser().updateEmail()`
- Para atualizar a senha, use: `mAuth.getCurrentUser().updatePassword()`
- Para atualizar os dados do usuário na base, use:

  ```
  drUser.updateChildren(...)
  ```

  Esse método recebe um `map<String, Object>` onde a chave é campo a ser alterado e o valor é novo valor desse campo.