Permissões no Android 6

Nós já conseguimos instalar outras funcionalidades no aplicativo: envio de SMS, visualização de endereço no mapa e acesso ao site cadastrado.

Agora, queremos utilizar novamente o telefone do aluno, mas dessa vez, para fazermos uma ligação direto do menu.

Voltando ao trecho de menu de contexto, vamos adicionar no código um novo item, com menu.add. Vamos adicionar Ligar, depois, criaremos uma variável, que chamaremos itemLigar. Em seguida, criaremos um comportamento, faremos isto usando uma Intent implícita. Mas a action que criaremos agora, será específica. ACTION\_CALL. Então, o usuário poderá escolher aplicativos como Skype e telefone.

MenuItem itemLigar = menu.add("Ligar");  
new Intent(Intent.ACTION\_CALL);

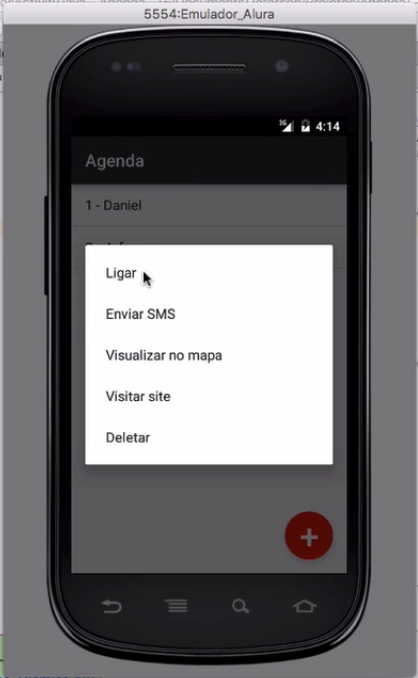
Se vamos ligar, precisamos informar o número do telefone. Para isto, utilizaremos o método setData para informar para a Intent qual o parâmetro da nossa activity. Usaremos o método Uri.parse e o protocolo. No nosso caso, adicionaremos tel:. Logo depois, iremos concatenar o número de telefone.

intentLigar.setData(Uri.parse("tel:" + aluno.getTelefone()));

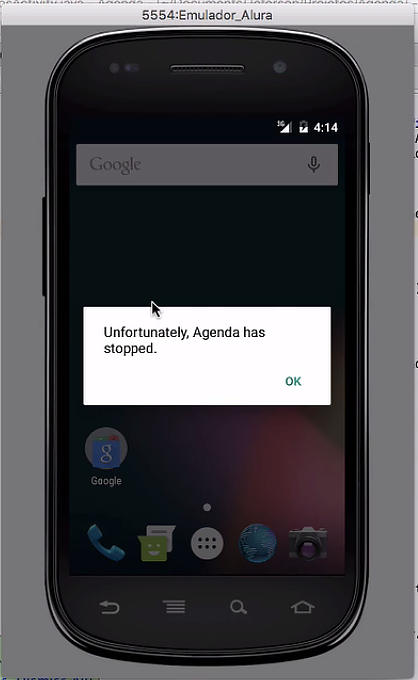
Iremos agora, associar o itemLigar com a intentLigar. O trecho do nosso código ficou assim:

MenuItem itemLigar = menu.add("Ligar");  
new Intent(Intent.ACTION\_CALL);  
itemLigar.setIntent(intentLigar);

Vamos a funcionalidade no nosso app. Após abrirmos o menu e clicarmos sobre o nome de um aluno, veremos a opção "Ligar".



Porém, teremos um problema...



Isto está acontecendo, porque até o Android 5, teríamos que fazer exatamente o que foi feito aqui. E nossa aplicação funcionaria normalmente. Com a chegada do Android 6, a parte de permissões do sistema operacional foi alterada. Usaremos o Facebook como exemplo. Ao instalarmos o app, ele informa várias permissões necessárias, mas não diz quando cada uma será utilizada. No modelo Android 6, a aplicação começa sem nenhuma permissão e à medida, que o app precisa acessar algum recurso, como o GPS, é o momento em que a permissão se torna necessária e o usuário precisa aceitá-la ou não.

Vamos ver o que muda no código para conseguirmos fazer a ligação: teremos que incluir o pedido de permissão para o usuário.

Antes, criando uma Intent e associando com um item, nós não conseguimos colocar comportamentos extras. Para adicionar novos comportamentos precisaremos, teremos que fazer como no Deletar, em que invés de criar uma Intent, usávamos o Listener para o MenuItem. Então, quando sobre ele, tínhamos o método onMenuItemClick e adicionávamos código Java, como no seguinte trecho:

MenuItem deletar = menu.add("Deletar");  
deletar.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 AlunoDAO dao = new AlunoDAO(ListaAlunoActivity.this);  
 dao.deleta(aluno;  
 dao.close();  
  
 carregaLista();  
 return false;  
 }  
 });

Fazíamos tudo dentro do método. Iremos fazer o mesmo com o itemLigar. Vamos adicionar o setOnMenuItemClickListener, seguido por uma classe anônima.

MenuItem itemLigar = menu.add("Ligar");  
itemLigar.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 return false;  
 }  
})

Já temos o template do método. Vamos recortar duas linhas do código, em que estava a nossa action.

Intent intentLigar = new Intent(Intent.ACTION\_CALL);  
IntentLigar.setData(uri.parse("tel:" + aluno.getTelefone()));

E depois, movê-las para o método.

MenuItem itemLigar = menu.add("Ligar");  
itemLigar.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 Intent intentLigar = new Intent(Intent.ACTION\_CALL);  
 IntentLigar.setData(Uri.parse("tel:" + aluno.getTelefone()));  
  
 return false;  
 }  
})

Iremos apagar a setIntent, porque não estamos mais associando Intent.

itemLigar.setIntent(IntentLigar);

Vamos instanciar a Intent e depois, chamamos o startActivity, usando como parâmetro a Intent implícita (intentLigar).

MenuItem itemLigar = menu.add("Ligar");  
itemLigar.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 Intent intentLigar = new Intent(Intent.ACTION\_CALL);  
 IntentLigar.setData(Uri.parse("tel:" + aluno.getTelefone()));  
 startActivity(intentLigar);  
  
 return false;  
 }  
});

Para essa parte do projeto funcionar, você precisa ter certeza de que o Compile Sdk Version está em API 23 Android 6.0 (Marshmallow) Você pode verificar isso no próprio Android Studio em File > Project Structure, selecione seu projeto em Modulesna coluna da esquerda e vá na aba Properties

Você também precisa estar com o Target Sdk Version em API 23 Android 6.0 (Marshmallow) Confira sua versão em File > Project Structure, selecione seu projeto em Modules na coluna da esquerda e vá na aba Flavors

Ainda, verifique se no arquivo build.gradle a dependência está na versão correta:

dependencies {  
 compile fileTree(include: ['\*.jar'], dir: 'libs')  
 androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  
 exclude group: 'com.android.support', module: 'support-annotations'  
})  
compile 'com.android.support:appcompat-v7:23.0.1'  
testCompile 'junit:junit:4.12'  
}

Mas a linha do startActivity ficará sublinhado em vermelho, avisando que a Intent implícita irá precisar de uma checagem de permissão.

Se não tivéssemos já adicionado a permissão no manifest...

//...  
<manifest  
xmls:android="http:schemas.android.com/apk/res/android"  
 package="br.com.alura.agenda" >  
  
 <uses-permission android:name="android.permission.CALL\_PHONE"/>

O Android Studio iria perdir que fizéssemos. Com um ALT + ENTER, ele irá adicionar o código necessário para fazer a checagem de permissão. É bom conhecermos isto, mas nós iremos criar o código.

Primeiramente, iremos verificar se já temos fazer a checagem de permissão e vamos gerar um if. Iremos ver se já temos a permissão com a classe ActivityCompat em que temos o método CheckSelfPermission. Caso exista a permissão do usuário, já podemos usá-la. Adicionaremos também os parâmetros: o contexto ( ListaAlunosActivity.this) e a permissão que será checada ( Manifest.permission.CALL\_PHONE). Com a classe PackageManager e a constante PERMISSION\_GRANTED iremos conferir se a permissão já existe.

Nossa condição ficou assim:

@Override  
public boolean onMenuItemClick(MenuItem item) {  
 if (ActivityCompat.checkSelfPermission(ListaAlunosActivity.this, Manifest.permission.CALL\_PHONE)  
 != PackageManager.PERMISSION\_GRANTED) {  
  
 }  
 \...

Se ainda não foi dada a permissão, teremos que pedi-la. Faremos isto com ActivityCompat com o método requestPermission. Os parâmetros iremos preencher com ListaAlunosActivity e iremos instanciar um *array* de *strings*, porque poderá ser mais de uma.

\...  
itemLigar.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuClick(MenuItem item) {  
 if (ActivityCompat.checkSelfPermission(ListaAlunosActivity.this, Manifest.permission.CALL\_PHONE)   
 != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions();  
  
 }  
  
 Intent intentLigar = new Intent(Intent.ACTION\_CALL);  
 intentLigar.setData(Uri.parse("tel:" + aluno.gettelefone()));  
  
 startActivity(intentLigar);  
  
 return false;  
  
 }  
 });

Dentro da String, quando formos especificar, iremos usar novamente a classe manifest. O último parâmetro que ele irá pedir é o chamado request code. Por enquanto, vamos preencher com 123.

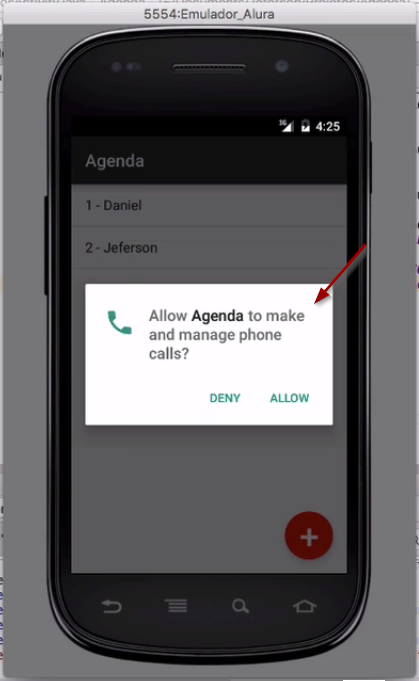
ActivityCompat.requestPermissions(ListaAlunosActivity.this, new String[][Manisfest.permission.CALL\_PHONE], 123);

Se não tivermos a permissão, pediremos para o usuário.

Caso a condição não for verdadeira, cairemos no else, porque já temos a permissão. Então, basta realizar a ação que queríamos: iniciar a chamada telefônica.

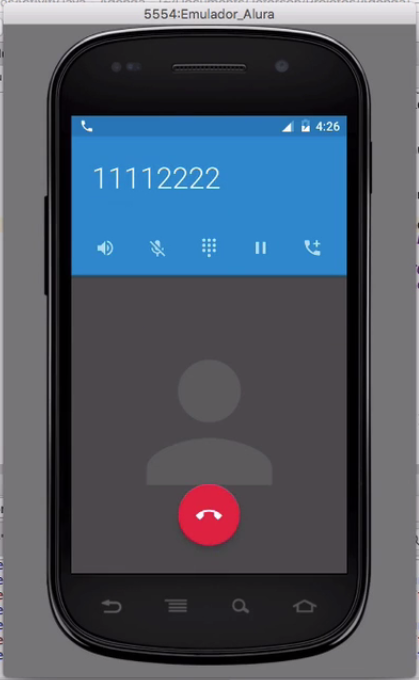
} else {  
 Intent intentLigar = new Intent(Intent.ACTION\_CALL);  
 intentLigar.setData(Uri.parse("tel:" + aluno.getTelefone()));  
 startActivity(intentLigar);  
  
return false;  
}

Vamos verificar se o nosso código já resolve o problema. Rodaremos a aplicação no emulador. Ao clicarmos sobre "Daniel" e selecionarmos a opção "Ligar", irá surgir uma janela de permissão perguntando "você permite que a Agenda faça e gerencie chamadas telefônica".



A mensagem irá aparecer de acordo com o idioma que estiver configurado no seu celular. No emulador, apareceram as duas opções: permitir e negar. Neste caso, nós iremos permitir ("allow").

Agora, a ação de ligar poderá ser realizada. Se clicarmos novamente sobre "Daniel" e em "Ligar", poderemos fazer a ligação.



Ele abriu o activity para ligação. Aparentemente, está tudo funcionando corretamente.

Vamos conversar sobre o request code. Toda vez que precisarmos de uma permissão no Android, teremos que pedir para o usuário com o request permission. Logo após a permissão, podemos querer abrir um popup ou outra ação. Para fazer isto, o Android irá chamar o método onRequestPermissionResult, independentemente se foi dada ou não a permissão. Dentro, iremos informar todas as permissões que foram solicitadas e os resultados. Passaremos dois *arrays*, mas o importante é que ele irá cair dentro e poderemos inserir o código que quisermos.

@Override  
public void onRequestPermissionResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantTesults) {  
 super.onRequestPermissionResult(requestCode, permissions, grantResults);  
  
}

Vamos imaginar que queremos incluir dentro // faz a ligacao.

@Override  
public void onRequestPermissionResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantTesults) {  
 super.onRequestPermissionResult(requestCode, permissions, grantResults);  
  
 // faz a ligacao  
}

Agora, imagine que precisassemos de permissão para o envio de SMS. No trecho referente, precisaríamos adicionar também um Listener para pedir a permissão e então, faríamos o request permission. Porém, este chamaria o onRequestPermissionResult, que no nosso caso, fará uma ligação. E não é o que queríamos...

Precisamos criar uma forma de distinguir qual a permissão que está voltando, quem fez o pedido, e o que queremos fazer de fato. É onde entra o *request code*. Na ligação, o parâmetro 123 é o *request code*.

ActivityCompat.requestPermissions(ListaAlunosActivity.this, new String[][Manisfest.permission.CALL\_PHONE], 123);

Para termos certeza que a ligação é a ação desejada, podemos incluir um requestCode = 123.

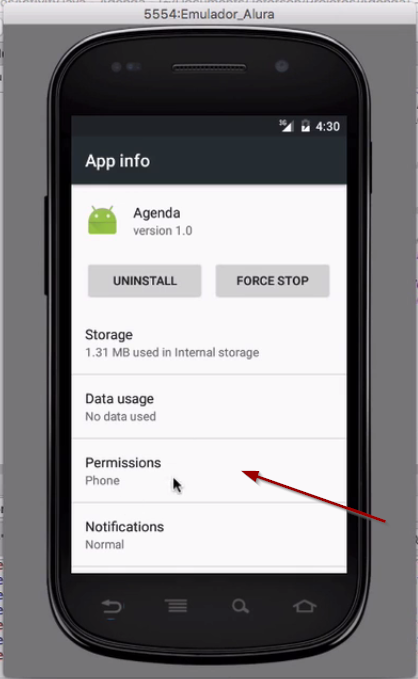
@Override  
public void onRequestPermissionResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantTesults) {  
 super.onRequestPermissionResult(requestCode, permissions, grantResults);  
  
 if (requestCode == 123) {  
 // faz a ligacao  
 }  
}

Se o código do request do SMS fosse 124, poderíamos inserir else if que // faz o envio do SMS.

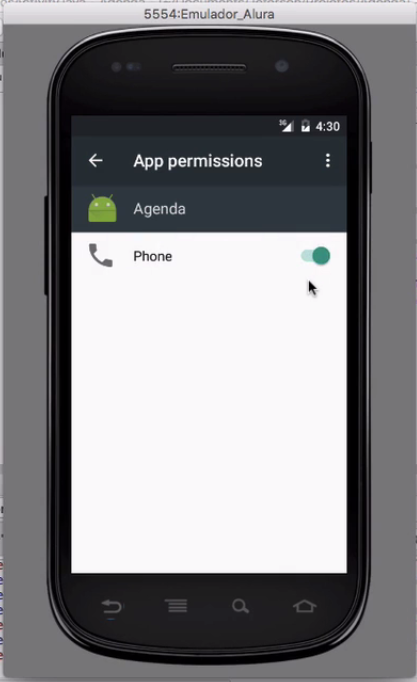
if (requestCode == 123) {  
 // faz a ligacao  
} else if (requestCode == 124) {  
 // faz o envio do SMS  
  
 }

Nós não vamos utilizar este método, mas é interessante conhecê-lo. Iremos retirar as linhas de acima do nosso código. E nosso app faz as chamadas telefônicas.

Além de criarmos as janelas de aplicação, o Android 6 permite ao usuário mexer nas permissões das aplicações. Então, se abrimos o menu das apps e encontramos o nosso. Veremos a seção de Permissions.



Ao clicarmos na seção, encontraremos todas as permissões que incluímos no manifest. E o usuário tem a opção de ativá-las ou desativá-las.



Tem o mesmo efeito de permitir ou não diretamente na aplicação. Por exemplo, vamos desativar a permissão no menu de "Permissions" da agenda. Com isto, quando tentamos fazer uma nova ligação na Agenda, será solicitado novamente a permissão do usuário. Caso seja negada, não será feita a chamada.

Precisamos estar preparados para este tipo de necessidade do Android 6. Nas versões anteriores, incluímos as permissões no manifest e estará resolvido.

Quando incluirmos recursos de segurança ou que gere custos para o usuário, o Android também irá pedir permissões.