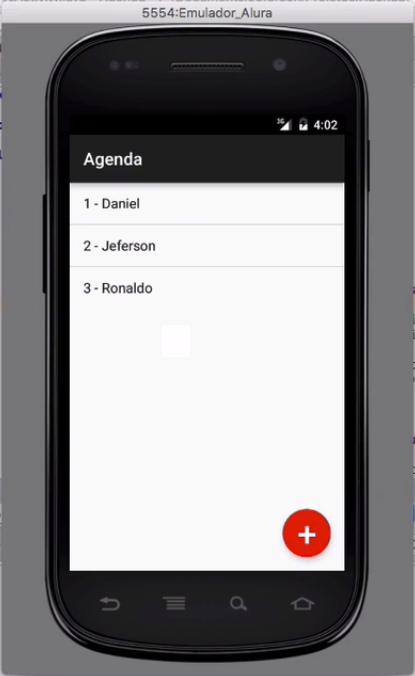
## **Permissões, SMS, mapa e navegador**

Olá, pessoal! Na primeira parte do nosso curso, criamos uma aplicação que basicamente realizava o cadastro de alunos. Fizemos uma agenda simples, mas que envolvia muitos itens do Android, principalmente a parte de banco de dados do SQLite.

Se observarmos a tela do app, temos um agenda, em que conseguíamos cadastrar alunos.

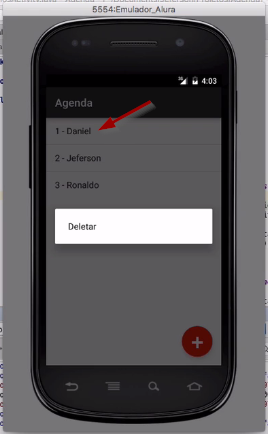


Se clicássemos sobre o nome de algum dos alunos, conseguíamos salvar informações (como site e telefone), além de poder inclusive avaliá-los.



Agora, queremos encontrar um forma de aproveitar estas informações. Por exemplo, seria interessante se pudéssemos clicar no campo em que foi preenchido um site e a partir do endereço poder acessar o navegador, para checar diretamente o site do aluno. Como poderíamos fazer isto?

Quando queríamos deletar o cadastro de um aluno, tínhamos a opção de dar um clique logo e seria aberto um menu de contexto com algumas opções específicas.



Nós poderíamos aproveitar o menu e incluir uma nova opção, como por exemplo, "visitar o site do aluno".

Voltando para o Android Studio, tínhamos o seguinte trecho de código:

Button novoAluno = (Button) findViewById(R.id.novo\_aluno);  
novoAluno.setOn(lickListener((v) -> {  
 Intent intentVaiProFormulario = new Intent(ListaAlunosActivity.this, FormularioActivity.class);  
 startActivity(intenVaiProFormulario);  
});  
  
registerForContextMenu(listaAluno);

Nele, está contido o registerForContextMenu(listaAluno);, que serve justamente para informar ao Android que temos um menu de contexto. Com isto, todas as vezes em que dávamos um clique longo na nossa lista, o Android automaticamente chamava o método CreateContextMenu.

Vamos para este trecho do código:

//...  
  
@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 MenuItem deletar = menu.add("Deletar");  
 deletar.setOnMenuItemClickListener(item) -> {  
 AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;  
 Aluno aluno = (Aluno) listaAlunos.getItemPosition(info.position);  
  
 AlunoDAO dao = new AlunoDAO(ListaAlunoActivity.this);  
 dao.deleta(aluno);  
 dao.close():  
  
 carregaLista();  
 returne false;  
  
  
 });  
}

Era aqui, em que definíamos quais itens iriam aparecer no nosso menu. Então, se queremos incluir um novo item, devemos ir até o método, e vamos chamar novamente o método menu.add - que nos passa um menu de contexto, no qual podemos adicionar os itens que queremos dentro dele. Apenas precisamos criar um comportamento para este item.

Então, para visitar o site do aluno adicionaremos menu.add e depois, o texto que será visualizado (Visitar site):

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 menu.add("Visitar site");  
  
//...

Vamos usar o atalho do Android Studio para ele criar uma variável para mim com ALT + ENTER. Devemos escolher a opção Introduce local variable do menu que irá surgir. Em seguida, daremos um nome descritivo itemSite. Para colocarmos um comportamento nele, usaremos a ideia de listener e vamos adicionar a seguinte linha:

itemSite.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener()

Ao clicarmos ENTER, o Android Studio irá preencher automaticamente.

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 MenuItem itemSite = menu.add("Visitar site");  
 itemSite.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 return false;  
 }  
});

Agora, dentro deste método vamos colocar o comportamento. Nosso objetivo é sair da aplicação e entrar no navegador. Queremos ir de um activity para outro. Precisaremos criar intent e passar o contexto, além da classe do activity que queremos ir. Como em uma classe anônima não podemos usar o this, mas para o contexto usaremos o nome da classe ListaAlunoActivity.this - e assim, especificar que queremos o this da classe de fora e não da anônima. E depois, definir que queremos ir para o navegador. Para isto, vamos arriscar usar a classe Browser.class. Por enquanto, é irrelevante se esta classe de fato existe. A nova linha ficará assim:

new Intent(ListaAlunosActivity.this, Browser.class);

A linha será inserida no seguinte trecho do código:

@Override  
public boolean onMenuItemClick(MenuItem item) {  
 new Intent(ListaAlunosActivity.this, Browser.class);  
 return false;  
}

Vou criar uma variável local para Intent, com o comando ALT + ENTER e depois clicando em Introduce local variable. Em seguida, irei renomeá-la como IntentSite.

@Override  
public boolean onMenuItemClick(MenuItem item) {  
 Intent intentSite = new Intent(ListaAlunosActivity.this, Browser.class);  
 return false;  
}

Agora que temos a Intent, precisamos informar o sistema operacional. Logo, nós precisaremos chamar o método startActivity() e dentro, passaremos intentSite, que tem as informações para o Android.

@Override  
public boolean onMenuItemClick(MenuItem item) {  
 Intent intentSite = new Intent(ListaAlunosActivity.this, Browser.class);  
 startActivity(intentSite);  
 return false;  
}

Se rodássemos o código, o que aconteceria é: daríamos um clique longo, que abrirá um menu de contexto. Depois, selecionaríamos a opção "Visitar o site" e automaticamente acessaríamos o site.

Nós poderíamos criar a Intent desta forma, fazendo o Listener, adicionando o new Intent e o startActivity. A maneira como fizemos anteriormente:

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 MenuItem itemSite = menu.add("Visitar site");  
 itemSite.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 Intent intentSite = new Intent(ListaAlunosActivity.this, Browser.class);  
 startActivity(intentSite);  
 return false;  
 }  
});  
  
//...

Porém, será um trabalho repetitivo, todo item de menu terá OnMenuItemClickListenere vamos adicionar um comportamento que basicamente é um Intent e um startActivity. Para não repetirmos sempre este código, o Android disponibilizou um atalho.

Então, vamos recortar a seguinte linha e colocá-la mais acima:

Intent intentSite = new Intent(ListaAlunosActivity.this, Browser.class);

Depois, vamos retirar o Listener.

@Override  
public boolean onMenuItemClick(MenuItem item) {  
 Intent intentSite = new Intent(ListaAlunosActivity.this, Browser.class);  
 startActivity(intentSite);  
 return false;  
}

Esse será o nosso código com as alterações:

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 MenuItem itemSite = menu.add("Visitar site");  
 itemSite.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {  
 Intent intentSite = new Intent(ListaAlunosActivity.this, Browser.class);  
 }  
});

Já temos o Item e o Intent, agora, precisamos associar os dois. Faremos isto informando que o itemSite tem uma Intent associada com o método setIntent. Dentro, especificaremos que é a intentSite que está associada.

itemSite.setIntent(intentSite);

Com isto, alcançaremos o mesmo efeito do que fizemos antes. Quando clicarmos em cima do item, automaticamente ele dará um startActivity na Intent criada. Em teoria, ele já abrirá o site do aluno no navegador.

Agora, faltou definir alguns detalhes sobre a Browser.class. No celular, podemos ter diferentes navegadores instalados - uma das grandes vantagens do android é a opção de personalização. Se você não gosta do Browser que vem instalado no aparelho, tem opção de baixar outro e ele funcionará juntamente com o instalado inicialmente.

Imagine que já temos o nosso aplicativo, o site está rodando e queremos acessar a página cadastrada do aluno. Se temos mais de um navegador instalado, o que seria interessante acontecer neste momento? Ao clicar o endereço do site, o app poderia oferecer as opções de navegadores para o usuário selecionar.

Então, em vez de definirmos no código qual Browser queremos abrir, nós vamos dizer para o Android que queremos abrir o site. O responsável por escolher qual será o aplicativo que abrirá o site, será o usuário. Uma janela choose diolog será aberta e ele poderá escolher entre as opções de navegadores.

Para diferenciarmos, a Intent que explicita qual é a activity que seremos direcionados, chamamos de **Intent Explícita**.

Intent intentSite = new Intent(ListaAlunosActivity.this, Browser.class);

Quando dizemos que o usuários irá escolher e nós só sabemos qual ação queremos realizar, nós iremos utilizar uma **Intent Implícita**. Você informa para o Android 'quero abrir um site' e ele fará o possível para executar a ação.

Vamos ver como fazemos isto no código. Se observarmos a linha da Intent, em vez de passarmos os dois parâmetros, definiremos a ação (action) de visualizar algo.

Intent intentSite = new Intent(Intent.ACTION\_VIEW);

Mas se deixássemos apenas assim, o Android iria ficar se perguntando "visualizar o quê?"

A ação de visualizar pode ser usada para diversas coisas: com imagens, sites, ou outro recurso. Precisamos especificá-lo. E o que podemos usar para descrever qual site queremos ir? O endereço, a url.

Vamos na Intent, depois usaremos um método para dados **obrigatórios**, setData, que serve para passar um parâmetro.

intentSite.setData();

Este irá solicitar uma uri (Identificador uniforme de recursos), uma forma de identificar qualquer tipo de recurso no sistema operacional. Uma uri bastante conhecida é a url.

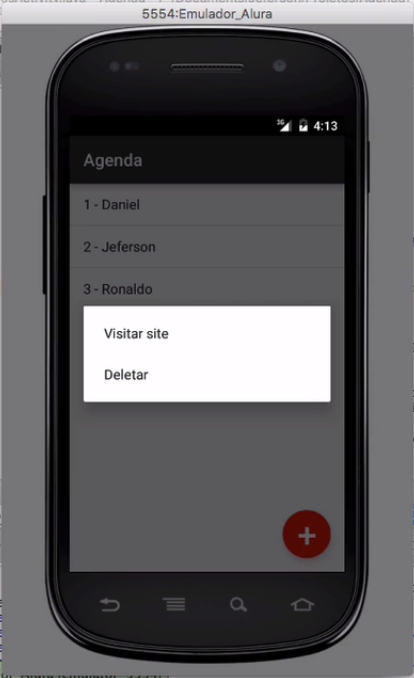
Poderíamos adicionar por exemplo o endereço http://www.google.com. Porém, se fizermos isto, o endereço ficará com um sublinhado vermelho, porque o que ele espera é um objeto do tipo uri e não, uma *String*. Sem problemas. Temos uma forma de gerar uma uri a partir de uma *String* : usando a classe Uri (com apenas a letra Umaiúscula) juntamente com o método parse.

intentSite.setData(Uri.parse("http://www.google.com"));

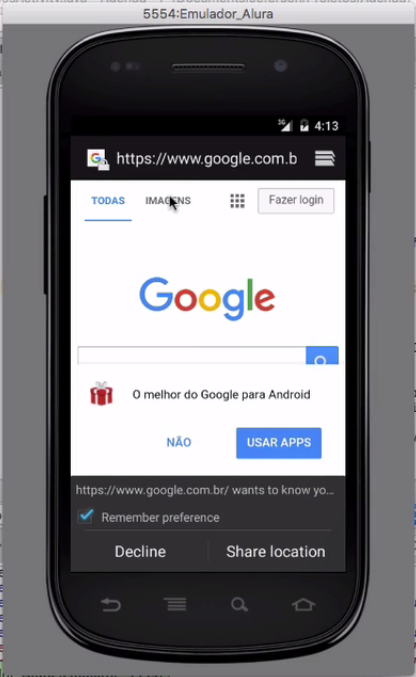
Vou fechar a classe com ALT + ENTER. Vamos ver como ficou nosso código:

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 MenuItem itemSite = menu.add("Visitar site");  
 Intent intentSite = new Intent(Intent.ACTION\_VIEW);  
 intentSite.setData(Uri.parse("http://www.google.com"));  
 ItemSite.setIntent(itentSite);  
});

Iremos testar o nosso app no emulador e vamos rodar a agenda. Após o aplicativo ser aberto, vamos dar um clique logo sobre a opção Daniel. Automaticamente, ele irá abrir um menu de contexto.



Depois, clicamos sobre Visitar site e ele irá abrir o navegador.



Ele abriu corretamente o site do Google. No entanto, ainda não alcançamos nosso objetivo. Nosso objetivo é clicarmos no site específico que está cadastrado no perfil do aluno. Se clicarmos sobre o nome do Jeferson, no app, veremos que está cadastrado o endereço www.alura.com.br.

Iremos alterar o nosso código. Em vez de colocarmos a *string* com o endereço, queremos adicionar o site do aluno aluno.getSite(). Porém, ainda não temos o aluno neste ponto.

Quando estamos fazendo o clickListener, nós nos associávamos ao aluno no seguinte trecho:

//...  
  
 MenuItem deletar = menu.add("Deletar");  
 deletar.setOnMenuItemClickListener(new MenuItem.OnMenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;  
 Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(info.position);  
 }  
});

Vamos mover duas linhas do Listener.

AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;  
Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(info.position);

Para o CreateContextMenu...

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;  
 Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(info.position);  
});

Atenção, que quando temos a referência de uma variável, dentro de uma classe anônima, e ela declarada fora, precisamos declarar esta variável como constante. Assim evitamos confusão no Java, ao alterarmos uma classe e correr o risco de alterar a outra.

Por isso, a exigência de que as variáveis que acessamos em uma classe anônima, sejam constantes no lugar onde foram declaradas. Para isto, vamos adicionar final antes de Aluno.

final Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(info.position);

Nós já declaramos o aluno, veremos como ficou o nosso código até aqui:

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, final ContextMenu.ContextMenuInfo menuInfo) {  
 AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;  
 final Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(info.position);  
  
 MenuItem itemSite = menu.add("Visitar site");  
 Intent intentSite = new Intent(Intent.ACTION\_VIEW);  
 intentSite.setData(Uri.parse(aluno.getSite()));  
 itemSite.setIntent(intentSite);  
  
 MenuItem deletar = menu.add("Deletar");  
 deletar.setOnMenuItemClickListener(new MenuItem.OnMenuItem.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 AlunoDAO dao = new AlunoDAO(ListaAlunosActivity.this);  
 dao.deleta(aluno);  
 dao.close();  
  
 carregaLista();  
 return false;  
  
 }  
 });  
  
}

No entanto, se voltarmos no emulador, veremos que os endereços estão cadastrados apenas com www no inicio. Isto significa, que está faltando https://, ou seja, o protocolo - justamente o que define para o Android se você está acessando uma imagem ou um site. Então, se chamamos www.caelum.com.br, o Android irá se perder. Para evitarmos este problema e para não necessitarmos que o usuário adicione o https://, vamos fazer um tratamento antes de colocarmos na Intent. Criaremos a variável site, em que vamos puxar o site do aluno, com aluno.getSite(). Com o método starsWith, iremos verificar se o endereço começa com https://. Se (if) ele não começar, teremos que concatenar o endereço. Logo, site será igual a https:// + o que o site tinha antes.

Em vez de pegarmos o site direto de aluno.getSite(), vamos chamá-lo de site.

MenuItem itemSite = menu.add("Visitar site");  
Intent intentSite = new Intent(Intent.ACTION\_VIEW);  
  
String site = aluno.getSite();  
if (!site.startsWith("https://")) {  
 site = "https://" + site;  
}   
  
intentSite.setData(Uri.parse(site));  
itemSite.setIntent(intentSite);

Vamos rodar o aplicativo para ver o que acontece. Com as atualizações, se clicarmos novamente no endereço cadastrado no perfil de Jeferson, (www.alura.com.br), ele irá abrir a página.



Se testarmos com outros alunos, veremos que somos direcionados corretamente para o endereços cadastrados.

Então, até aqui aprendemos como criar uma **Intent Implícita**, que é quando queremos trocar de tela na nossa aplicação e queremos aproveitar algo que já está pronto no celular e não ter que criar um Browser do zero, mas não sabemos qual classe irá tratar desta ação. Então, deixamos para o Android a tarefa e ele irá descobrir qual activitytem disponível para realizá-la. Nós instanciavámos a Intent e especificávamos qual ação queríamos realizar.