Aplicações Móveis

Exercícios de Revisão

Coordenação de Engenharia Informática

Departamento de Engenharias e Tecnologias

Instituto Superior Politécnico de Tecnologias e Ciências

Nome : Marcelo Rocha - 20210032

Android

1. No Android, existe a noção de **API Level**. Diga se concorda com a seguinte definição: "*API Level é um valor inteiro que identifica exclusivamente a revisão da API do framework oferecida por uma versão da plataforma Android*". Indique de que forma isso afecta a portabilidade das aplicações desenvolvidas. Justifique sua resposta.

   R: Sim , esta definição está correcta . O API Level afeta diretamente a portabilidade, pois define quais dispositivos e funcionalidades o aplicativo pode suportar. Quando um aplicativo é desenvolvido, ele pode ser projetado para suportar uma determinada versão mínima de API (especificada pelo minSdkVersion) e uma versão alvo (especificada pelo targetSdkVersion).

   Se o aplicativo usa funcionalidades introduzidas em um nível de API mais alto que a versão mínima suportada, ele pode não funcionar corretamente em dispositivos com versões mais antigas do Android, o que limita sua portabilidade. O desenvolvedor precisa garantir compatibilidade com as APIs mais antigas, seja evitando o uso de novas APIs ou implementando verificações de versão para usar alternativas em dispositivos com versões mais antigas.

   APIs antigas podem ser depreciadas ou removidas em versões mais recentes, afectando a portabilidade exigindo manutenção para manter o aplicativo funcional e atualizado. Se o aplicativo foi desenvolvido para uma versão antiga do API Level e usa APIs obsoletas, ele pode quebrar ou ter funcionalidades limitadas em versões mais novas do Android.

   Versões mais recentes introduzem novas políticas de segurança, afetando o comportamento do aplicativo se ele não aderir às mudanças prejudicando sua execução e distribuição.

2. No Android, as aplicações são construídos em torno de vários **componentes**. Mencione dois desses componentes mais importantes (além do componente "serviço") e forneça seu significado.

   R: No Android, além do componente serviço (service), dois dos componentes mais importantes são:

   - **Atividade (Activity)** : Uma atividade representa uma única tela da interface do usuário. Cada atividade contém os elementos de interação (botões, texto, listas etc.) que permitem que o usuário execute ações no aplicativo. As atividades formam o ponto de entrada para a interação do usuário com o aplicativo e têm um ciclo de vida próprio, gerenciado pelo sistema Android, que controla como elas são criadas, pausadas, retomadas e destruídas.

   - **Broadcast Receiver** : Um Broadcast Receiver permite que o aplicativo reaja a mensagens transmitidas pelo sistema ou outros aplicativos. Ele pode responder a eventos do sistema, como mudanças de conectividade de rede, a recepção de uma mensagem SMS ou o término de um download. Embora os Broadcast Receivers não interajam diretamente com o usuário, eles permitem que o aplicativo fique atento a eventos importantes e execute ações em segundo plano em resposta a esses eventos, como notificações ou sincronização de dados.

3. Considere o **sistema Android**. Porquê que esse sistema é necessário e os sistemas operativos de desktop existentes, como Unix, Windows ou MacOS, não são adequados? Relacione sua resposta à interface programática fornecida aos desenvolvedores de aplicações.

   R: O Android é necessário porque foi projetado especificamente para dispositivos móveis, levando em consideração suas limitações de hardware, como menor poder de processamento e necessidade de gerenciamento eficiente de energia. Diferente de sistemas operativos de desktop (como Unix, Windows ou MacOS), o Android é otimizado para interfaces touch, tem APIs específicas para sensores, geolocalização e comunicação, e oferece um modelo robusto de permissões para maior controle de privacidade. Além disso, ele gerencia o ciclo de vida dos aplicativos de forma eficiente e fornece um ecossistema de distribuição de apps ideal para o ambiente móvel.

4. No Android, considere o componente **serviço**. Este componente tem uma interface com o utilizador? Justifique sua resposta.

   R: No Android, o componente serviço (service) não tem uma interface direta com o utilizador. Os serviços são projetados para realizar tarefas em segundo plano sem interagir diretamente com o usuário, ou seja, eles não apresentam uma interface visual como as atividades (activities). A principal função de um serviço é continuar a executar operações longas ou em segundo plano, como tocar música, baixar arquivos, ou

processar dados, mesmo quando o usuário não está interagindo ativamente com o aplicativo. No entanto, serviços podem, por meio de outros componentes como notificações, informar o usuário sobre o que está acontecendo ou pedir sua atenção, mas as notificações não fazem parte do serviço em si, e sim de outros mecanismos de interação.

5. Considere o ficheiro **manifest.xml** anexado a cada aplicação Android. Este ficheiro especifica os serviços de hardware e software necessários e as bibliotecas externas que precisam ser vinculadas a uma aplicação? Relacione sua resposta com o cenário em que algum serviço de software não é referido nesse ficheiro.

   R: O arquivo AndroidManifest.xml é essencial em aplicações Android, pois especifica os componentes (como atividades e serviços), as permissões necessárias para acessar hardware e software (como câmera, GPS, internet), e as bibliotecas externas que o aplicativo usa. Se um serviço ou permissão não for declarado no manifest.xml, o aplicativo não poderá acessá-lo, resultando em falhas ou mau funcionamento, como a incapacidade de usar GPS ou internet. Um exemplo prático seria se um aplicativo precisar acessar a localização do dispositivo e a permissão não estiver declarada no manifest.xml, o aplicativo pode falhar ao tentar acessar o GPS. Além disso, o usuário não será solicitado a conceder essa permissão, o que pode causar a falha da funcionalidade que depende desse recurso.

6. Considere o mecanismo de **broadcast** no Android que é representado como um objecto Intent. Este mecanismo especifica um determinado objecto Activity que será acionado? O que o objecto Intent contém?.

   R: Não, o mecanismo de broadcast no Android não especifica diretamente uma Activity que será acionada. Em vez disso, ele é usado para enviar uma mensagem global (um broadcast) para o sistema Android ou outros aplicativos. Os receptores dessas mensagens são os Broadcast Receivers, não as atividades (activities). Um Broadcast Receiver pode ser configurado para reagir a essas mensagens e realizar uma ação, como notificar o usuário ou iniciar uma tarefa em segundo plano. O objecto Intent é uma classe que carrega dados sobre a ação a ser realizada ou o evento que está ocorrendo.

7. Uma **actividade** no Android pode iniciar outra actividade que existe em outra aplicação no dispositivo? Responda “sim” ou “não” e justifique com um exemplo em qualquer um dos casos.

   R: Sim, uma atividade no Android pode iniciar outra atividade que existe em outra aplicação no dispositivo. Isso é feito usando um Intent implícito. No caso de um Intent implícito, você não especifica diretamente a atividade de outra aplicação, mas define a ação que deseja realizar, e o Android escolhe qual aplicação ou atividade pode lidar com essa ação. Por exemplo se quisermos abrir uma página da web em um navegador, você pode usar um Intent implícito que pede ao sistema Android para encontrar um aplicativo (como o Chrome) capaz de lidar com a URL:

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("https://www.example.com"));
startActivity(intent);
```

Nesse exemplo, o sistema Android abrirá um navegador web instalado no dispositivo, que pertence a outra aplicação, para exibir a página solicitada.

8. Considere o Android e a seguinte frase: “As actividades são organizadas em uma pilha (**back stack**), na ordem em que cada actividade é aberta com uma lógica de primeiro a entrar, primeiro a sair”. Concorda? Responda “sim” ou “não” e justifique.

   R: Sim, concordo com a frase. No Android, as atividades são organizadas em uma pilha de navegação (back stack), e seguem a lógica de primeiro a entrar, último a sair (LIFO - Last In, First Out), o que significa que a última atividade aberta é a primeira a ser fechada quando o usuário pressiona o botão "voltar". Por exemplo se o usuário abrir a atividade A, ela será colocada na pilha. Abrindo a atividade B, ela é empilhada sobre a atividade A. Pressionando o botão "voltar", a atividade B é removida da pilha e a atividade A é trazida de volta para o primeiro plano.

9. Considerando que no Android uma **tarefa** é uma unidade coesa, actividades de primeiro e segundo plano, o que acontece quando um utilizador inicia uma nova tarefa ou vai para a tela inicial, por meio do botão inicial?

   R: Quando um usuário inicia uma nova tarefa uma nova atividade é criada e colocada na pilha de atividades (back stack). A nova atividade se torna a atividade em primeiro plano, enquanto a atividade anterior é colocada em segundo plano. Dependendo das configurações de tarefas e atividades, a atividade anterior pode ser mantida na pilha ou pode ser finalizada, dependendo da forma como a nova atividade é iniciada (por exemplo, usando Intent com a flag FLAG_ACTIVITY_NEW_TASK).

   Quando o usuário pressiona o botão Início, o sistema Android coloca a atividade atual em segundo plano e leva o usuário à tela inicial. A atividade em primeiro plano é pausada (seu ciclo de vida entra no estado onPause()), mas não é destruída imediatamente. Ela permanece na pilha de atividades, permitindo que o usuário retorne a ela facilmente. Se o usuário voltar à atividade anterior, esta será restaurada a partir do estado em que foi pausada.

10. O Android sugere **recursos** da aplicação, como imagens e strings, para serem externalizados do código? Em caso afirmativo, qual é o motivo?

    R: Sim, o Android recomenda externalizar recursos como imagens, strings, cores e layouts do código. Isso facilita a manutenção, suporte a múltiplos idiomas, organização do projeto, compatibilidade com diferentes dispositivos e reutilização dos recursos. Externalizar recursos permite modificar esses elementos sem alterar o código-fonte, tornando o desenvolvimento mais eficiente e flexível.

11. No Android, quais são os três **estados** em que uma actividade pode existir? Diga a que cada um corresponde.

R:

- **Ativa ou em execução (Running/Resumed) :** Quando a atividade está em primeiro plano, interagindo diretamente com o usuário e visível na tela. O método onResume() foi chamado e a atividade está ativa.

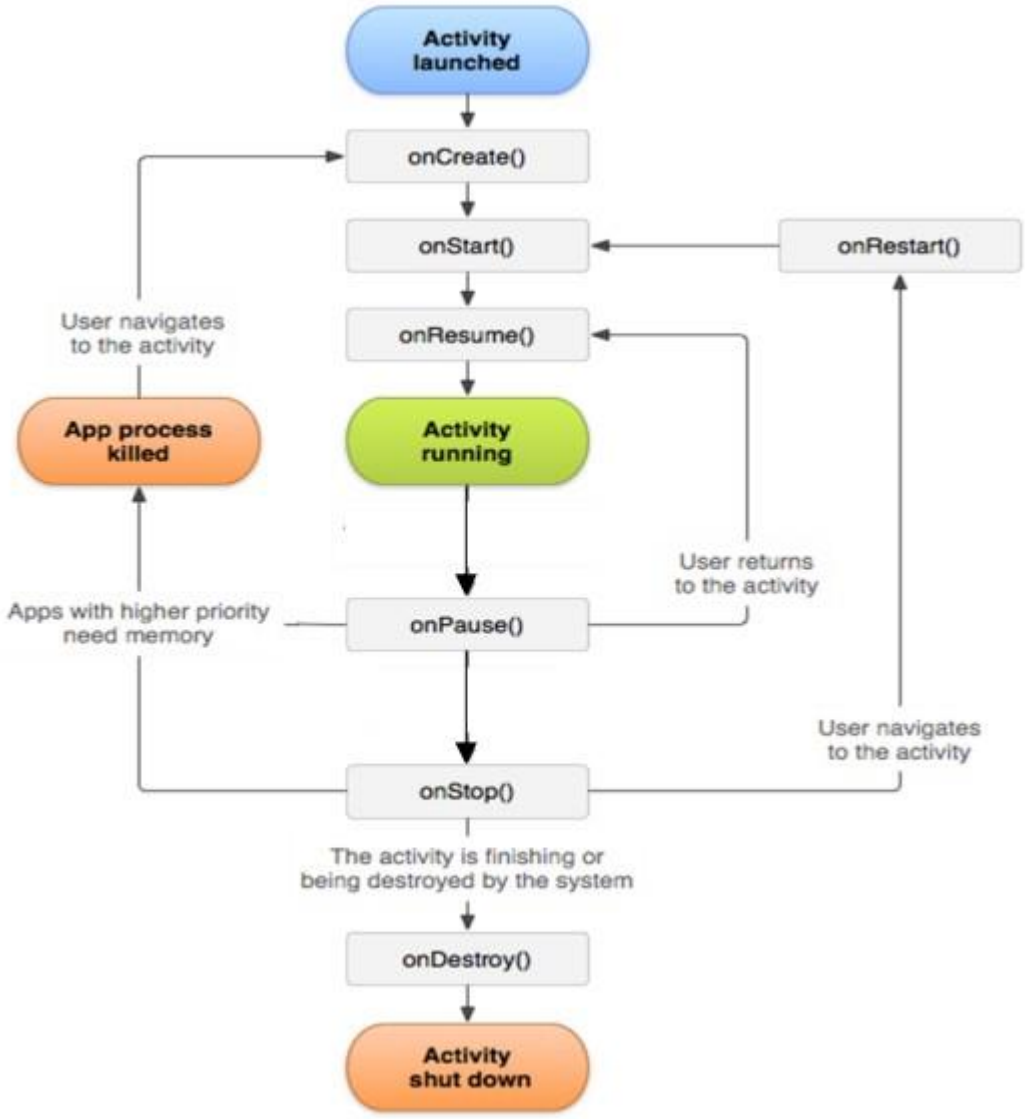
- **Pausada (Paused) :** A atividade ainda está parcialmente visível (por exemplo, uma nova atividade foi iniciada, mas a atual está em segundo plano), mas o usuário não está mais interagindo diretamente com ela. O método onPause() foi chamado. Ela pode ser retomada sem precisar reiniciar, pois permanece na memória.

- **Parada (Stopped):** A atividade não está visível para o usuário (como ao pressionar o botão "Início" para ir para a tela inicial). O método onStop() foi chamado. A atividade ainda está na pilha de atividades, mas pode ser finalizada se o sistema precisar liberar recursos.

12. Considere a figura que descreve o ciclo de vida da actividade. Entre quais chamadas (para quais funções) uma actividade está visível? Entre quais chamadas (para quais funções) uma Activity está em primeiro plano?

R: A atividade está visível entre as chamadas de onStart() e onStop().
Durante este intervalo, a atividade pode estar em primeiro plano (ativa) ou parcialmente visível (pausada).

A atividade está em primeiro plano (ativa e interagindo diretamente com o usuário) entre as chamadas de onResume() e onPause(). Neste estado, o usuário pode interagir completamente com a atividade.



13. O sistema Android suporta o mecanismo designado **AsyncTask** (tarefa assíncrona). Qual é a finalidade desse mecanismo? Em sua resposta, forneça um exemplo de seu uso.

R: O mecanismo AsyncTask no Android é utilizado para realizar operações em segundo plano sem bloquear a interface do usuário (UI). Sua finalidade é executar tarefas demoradas (como operações de rede, acesso ao banco de dados ou processamento de arquivos) de forma assíncrona, deixando a UI responsiva enquanto o processo ocorre em segundo plano.

O AsyncTask permite que você execute três passos principais:

- **onPreExecute():** Executa na thread principal (UI thread) antes de iniciar a tarefa em segundo plano. Serve para inicializar qualquer coisa antes da execução da tarefa (por exemplo, mostrar uma barra de progresso).

- **doInBackground():** Executa a tarefa em uma thread em segundo plano. Aqui você coloca o código que realiza a operação que pode demorar (como baixar dados da internet). Esse método não pode manipular a interface do usuário diretamente.

- **onPostExecute():** Executa na thread principal (UI thread) após a conclusão da tarefa. Aqui você pode atualizar a interface do usuário com o resultado da operação.

```
private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    ImageView imageView;

    public DownloadImageTask(ImageView imageView) {
        this.imageView = imageView;
    }

    // Executa antes da tarefa em segundo plano
    @Override
    protected void onPreExecute() {
        // Pode-se colocar um código para mostrar uma barra de progresso, por exemplo
    }

    // Tarefa em segundo plano: baixar a imagem
    @Override
    protected Bitmap doInBackground(String... urls) {
        String url = urls[0];
        Bitmap bitmap = null;
        try {
            InputStream input = new java.net.URL(url).openStream();
            bitmap = BitmapFactory.decodeStream(input);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bitmap;
    }

    // Executa após a conclusão da tarefa e atualiza a interface
    @Override
    protected void onPostExecute(Bitmap result) {
        // Define a imagem baixada no ImageView
        imageView.setImageBitmap(result);
    }
}

// Uso da AsyncTask para baixar uma imagem
new DownloadImageTask(imageView).execute("https://example.com/image.png");
```

A AsyncTask permite fazer o download da imagem em segundo plano sem travar a interface do usuário, garantindo uma experiência fluida para o usuário final.

OBS : ele foi depreciado nas versões mais recentes do Android (a partir do Android 11), e o uso de Executors ou Coroutines com o LiveData/ViewModel agora é recomendado para melhor desempenho e controle.

14. Considere o sistema Android. O Android UI toolkit é thread-safe? Esclareça sua resposta indicando se pode manipular a interface do utilizador de um thread de trabalho.

R: Não, o Android UI toolkit não é thread-safe. Isso significa que a interface do usuário (UI) do Android só pode ser manipulada diretamente pela thread principal (também chamada de UI thread ou main thread). Manipular a UI a partir de uma thread de trabalho (ou qualquer thread que não seja a principal) pode causar erros imprevisíveis e falhas no aplicativo.

A manipulação da interface gráfica envolve acesso a recursos compartilhados, como widgets, layouts e elementos visuais, que não são projetados para serem acessados por múltiplas threads simultaneamente. Se várias threads tentarem modificar a UI ao mesmo tempo, podem ocorrer condições de corrida e corrupção dos elementos gráficos. Para atualizar a UI a partir de outras threads, é necessário utilizar mecanismos específicos que garantem que as modificações ocorram na thread principal, preservando a integridade da UI.