MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAIANO – CAMPUS GUANAMBI

THREADS E UI THREAD

GUANAMBI - BA

2022

CLOVES DE BRITO RODRIGUES JÚNIOR

LEONARDO RODRIGUES SANTANA

PAULO SÉRGIO SANTOS DE OLIVEIRA

THREADS E UI THREAD

Projeto apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Baiano – *Campus* Guanambi como parte dos requisitos da disciplina Laboratório de Programação para Dispositivos Móveis e Sem Fio.

Prof. Joao Paulo Barbosa Gloria

GUANAMBI - BA

2022

SUMÁRIO

1. INTRODUÇÃO

Quando um componente de aplicativo é iniciado, e não há outro componente em execução, o sistema Android inicia um novo processo no Linux para o aplicativo com um único thread de execução. Por padrão, todos os componentes do mesmo aplicativo são executados no mesmo processo e thread (chamado de thread “principal”).

2. THREADS E UI THREAD

Quando um aplicativo é executado, o sistema cria um thread de execução para ele, que é chamado de “principal”. Esse thread é muito importante porque ele é encarregado de despachar eventos para os widgets adequados da interface do usuário, incluindo eventos de desenho.

Quase sempre, é também o thread em que o aplicativo interage com componentes do kit de ferramentas da IU do Android (componentes dos pacotes android.widget e android.view). Portanto, o thread principal também pode se chamar thread de IU.

No entanto, em circunstâncias especiais, o thread principal de um aplicativo pode não ser o thread de IU.

O sistema não cria um thread separado para cada instância de um componente. Todos os componentes executados no mesmo processo são instanciados no thread de IU, e as chamadas do sistema para cada componente são despachadas a partir deste thread. Consequentemente, métodos que respondam aos callbacks do sistema (como onKeyDown() para informar ações de usuário ou um método callback do ciclo de vida) sempre serão executados no thread de IU do processo.

Por exemplo, quando o usuário toca em um botão na tela, o thread de IU do aplicativo despacha o evento de toque para o widget que, em troca, ativa o estado

pressionado e publica uma solicitação de invalidação à fila do evento. O thread de IU retira a solicitação da fila e notifica o widget de que ele deve desenhar novamente por conta própria.

Quando o aplicativo realiza trabalho intenso em resposta à interação do usuário, esse modelo de um thread pode produzir desempenho inferior, a não ser que você implemente o aplicativo adequadamente.

Especificamente, se tudo estiver acontecendo no thread de IU, realizar operações longas, como acesso à rede ou consultas a banco de dados, bloqueará toda a IU. Quando o thread é bloqueado, nenhum evento pode ser despachado, incluindo eventos de desenho. Da perspectiva do usuário, o aplicativo parece travar. Pior ainda, se o thread de IU for bloqueado por mais do que alguns segundos (cerca de 5 segundos atualmente), o usuário receberá a vergonhosa mensagem “aplicativo não respondendo” (ANR). O usuário talvez decida fechar o aplicativo e desinstalá-lo se estiver descontente.

Além disso, o kit de ferramentas de IU do Android não é seguro para threads. Portanto, você não pode gerenciar a IU de um thread de trabalho. Gerencie a interface do usuário no thread de IU. A partir disso, há duas regras simples para o modelo de thread único do Android:

Não bloquear o encadeamento da IU

Não acessar o kit de ferramentas de IU do Android fora do encadeamento da IU

3. MÉTODOS SEGUROS PARA THREADS

Em algumas situações, os métodos implementados podem ser chamados a partir de mais de um thread. Por isso, eles precisam ser programados para serem seguros para threads.

Isso se aplica especialmente para métodos que podem ser chamados remotamente, como métodos em um serviço vinculado. Quando uma chamada em um método implementado em um IBinder tem origem no mesmo processo de execução de IBinder, o método é executado no thread do autor da chamada. No entanto, quando a chamada tiver origem em outro processo, o método será executado em um thread escolhido a partir de uma série de threads que o sistema mantém no mesmo processo que IBinder (ele não será executado no thread de IU do processo).