Main Building Blocks
Yamba Overview

EA998/MC933
Guido Araujo e Sandro Rigo

Hoje

- Ch1: Android Overview
- Ch2: The Stack
- Ch3: Quick Start
- Ch4: Main Building Blocks
- Ch5: Yamba Project Overview
- Ch6: Android User Interface
- Ch7: Preferences, Project Overview, Filesystems, Menus and Intents
- Ch8: Services
- Ch9: Database
- Ch10: List Adapters
- Ch11: Broadcast receivers

Hoje

- Ch1: Android Overview
- Ch2: The Stack
- Ch3: Quick Start
- Ch4: Main Building Blocks
- Ch5: Yamba Project Overview
- Ch6: Android User Interface
- Ch7: Preferences, Project Overview, Filesystems, Menus and Intents
- Ch8: Services
- Ch9: Database
- Ch10: List Adapters
- Ch11: Broadcast receivers

Main Building Blocks

- Componentes de projeto Android
  - Activities
  - Intents
  - Services
  - Broadcast Receivers
  - Content Providers
- Objetivo da aula
  - Entender os componentes principais de projeto usando do Android
  - Compreender quando usar estes componentes
  - Entender relação destes componentes com o ambiente

Um Exemplo Real

- Estratégia de projeto
  - Top-down
  - Pensar nas telas que o usuário irá ver
  - Quais as caracterísiticas e funcionalidades de cada tela
  - Como as tela interagem entre si.

- Twitter (Yamba)
  - Tela 1: usuário informa suas atualizações
  - Tela 2: usuário é capaz de ver o que os seus amigos estão fazendo
  - Tela 3: usuário precisa entrar nome de usuário de senha

Principais Requisitos

- Precisa ser rápido e independente da rede no ar
- Usar uma cache local que permita trazer dados remotos
- Demanda um banco de dados
- Requer um serviço executando em background
- Serviço em background deve ser iniciado quando dispositivo ligar

Activities

- Características
  - Activity: uma única tela que o usuário ver por vez
  - Equivalente a uma página web
  - Possui também uma *main activity* semelhante a uma *homepage*
  - Navegação semelhante à Web

- Exemplo
  - Está dentro da app Contact e escolhe um amigo para enviar texto
  - Laça uma nova activity para compor uma mensagem de texto na app Messaging

Lançando uma activity

- Muito custoso
- Cria um processo Linux
- Aloca memória para objetos do usuário da UI
- Gera objetos a partir dos seus layouts XML
- Prepara e ajusta toda a tela

Activity Manager (AM)

- Funções principais
  - Gerenciador de “cache” de atividades
  - Criar, destruir e gerenciar activities
  - Projetista tem pouco controle dos recursos (CPU e bateria)
  - No início cria a atividade e coloca-a na tela
  - Quando o usuário troca de activity move a anterior para estado de espera
  - Quando o usuário retornar a atividade pode ser mostrada mais rapidamente
  - Activities mais antigas são destruídas a medida que vão ficando velhas

AM: Máquina de Estados (Life Cycle)

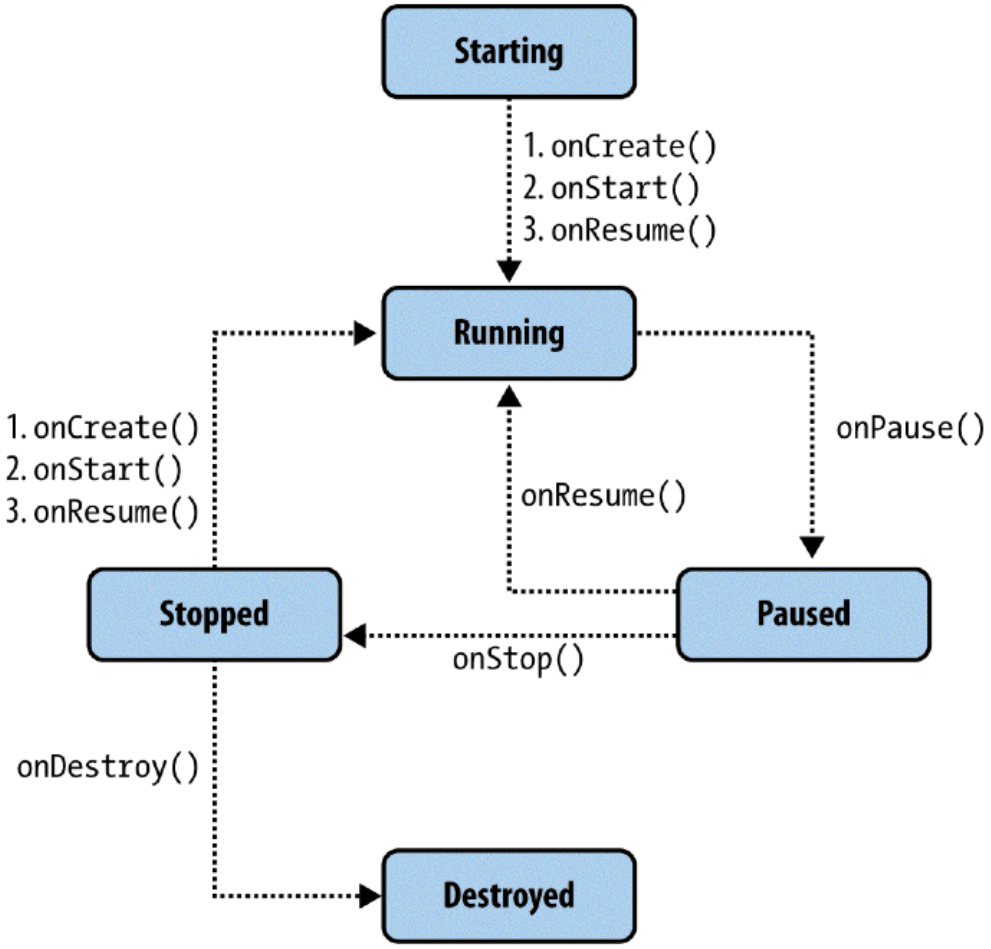


Figure 4-1. Activity life cycle

AM: Starting

Cria e inicializa uma atividade

Muito custosa em energia

Métodos usados para transição

- onCreate()
- onStart()
- onResume()

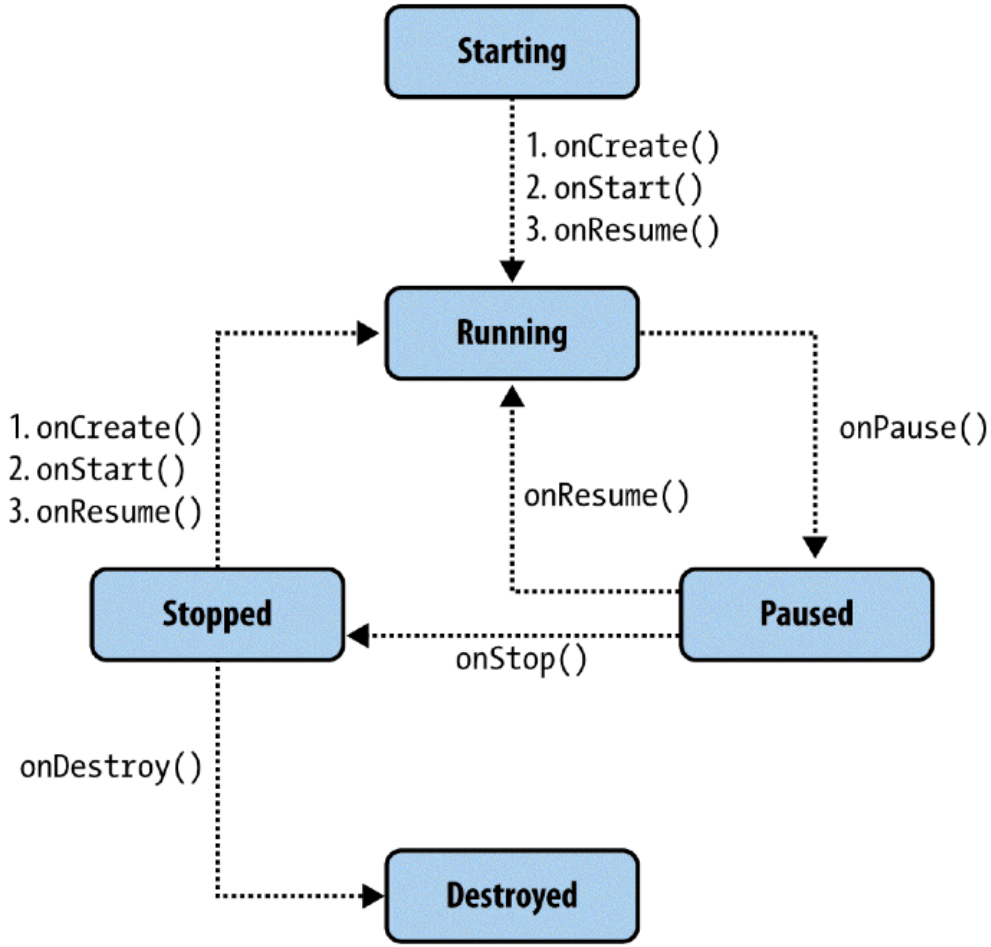


Figure 4-1. Activity life cycle

AM: Running

Activity na tela e interagindo

Activity em "*foco*"

Somente uma exec. por vez

Gerencia tarefas como:

- Digitação
- Toque de telas
- Clique de botões

Prioridade máxima de recursos

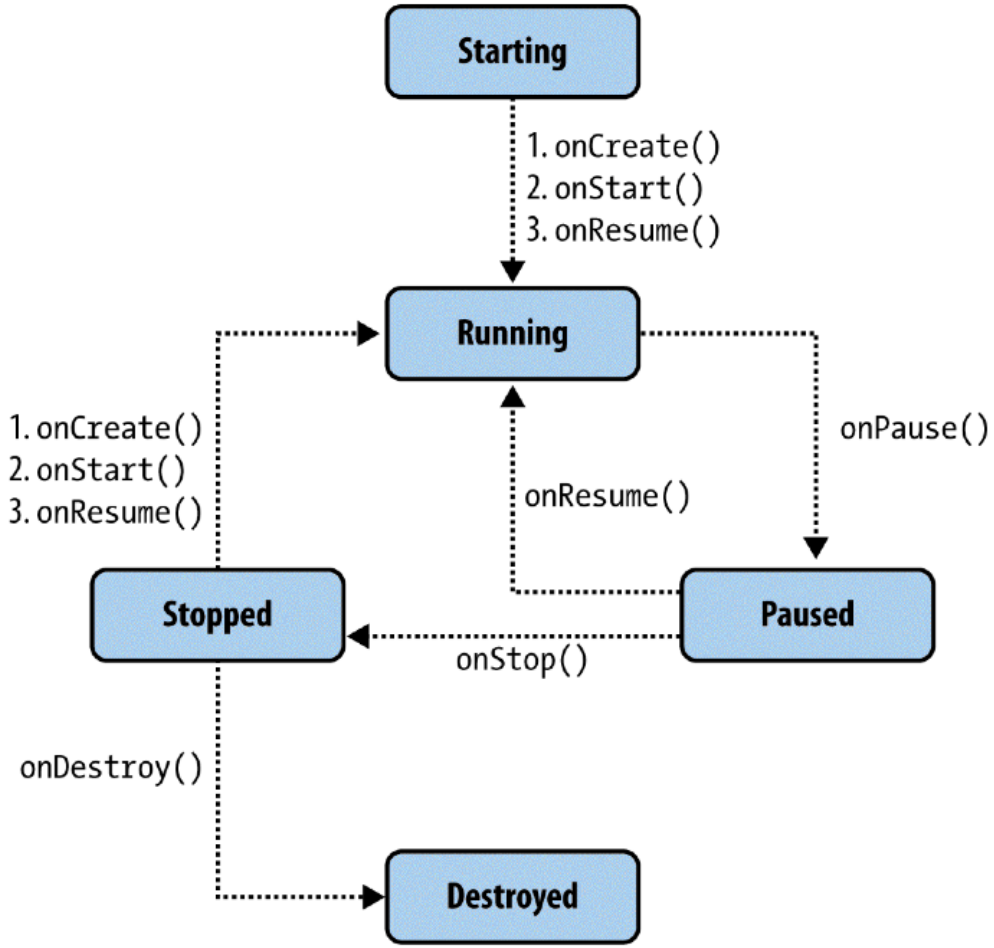


Figure 4-1. Activity life cycle

AM: Pause

Entra quando onPause() executa

Na tela, mas não interagindo

Exemplo: dialog box na frente de uma activity

Podem mover-se para estados:

- Sttoped: via onStop()
- Running: via onResume()

Prioridade máxima de recursos

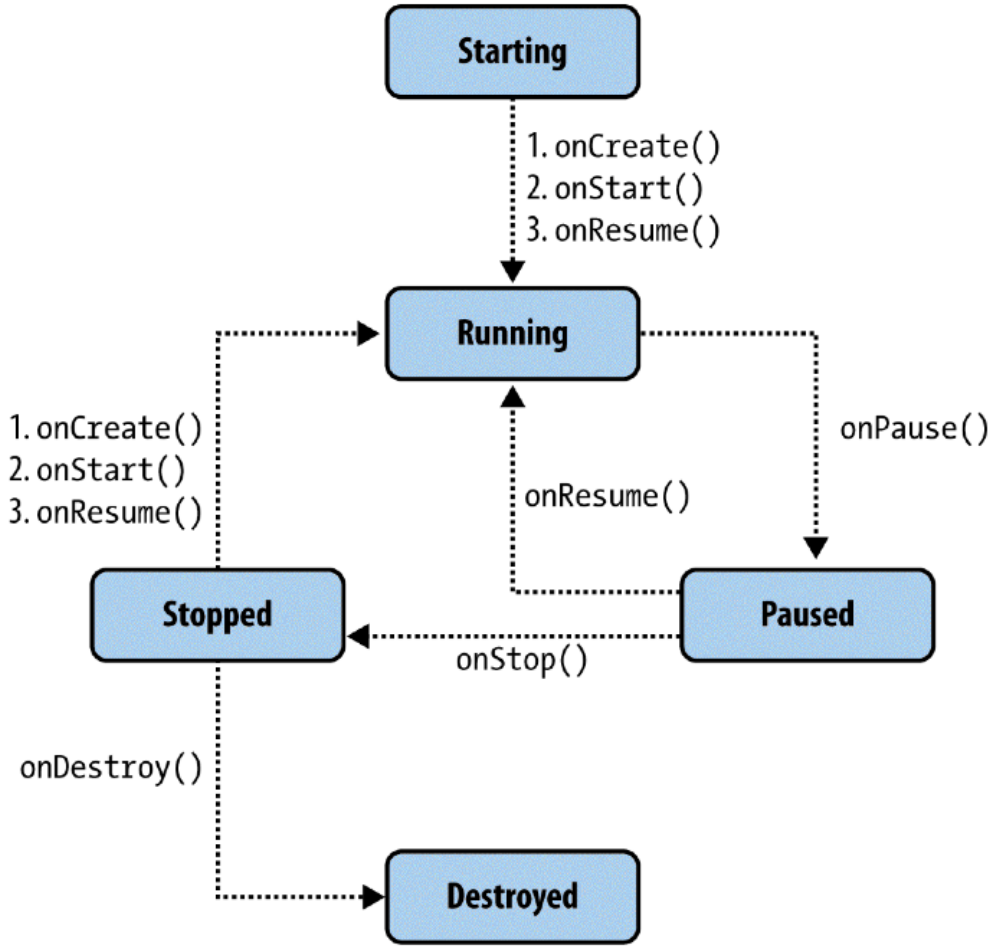


Figure 4-1. Activity life cycle

AM: Stopped

Activity não visível

Ainda na memória

Na tela, mas não interagindo

Mesmo assim pode estar exec.

Podem mover-se para estados:

- Destroy: via onDestroy()
- Running: via

  onCreate()

  onStart()

  onResume()

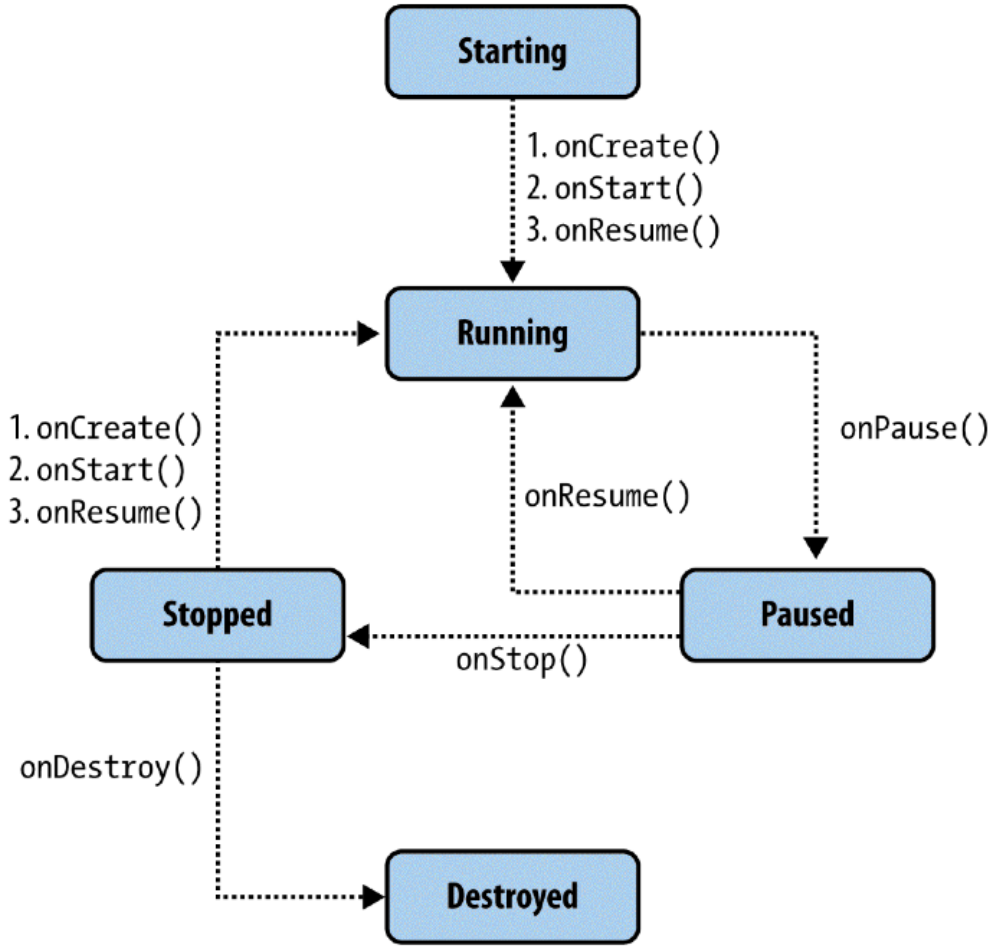


Figure 4-1. Activity life cycle

AM: Destroyed

Activity não é mais necessária

Antes de destruída pode realizar algumas tarefas como salvar dados, etc.

Em alguns casos pode ser destruída em Paused

Melhor deixar para salvar dados antes de entrar em Paused vindo de Running

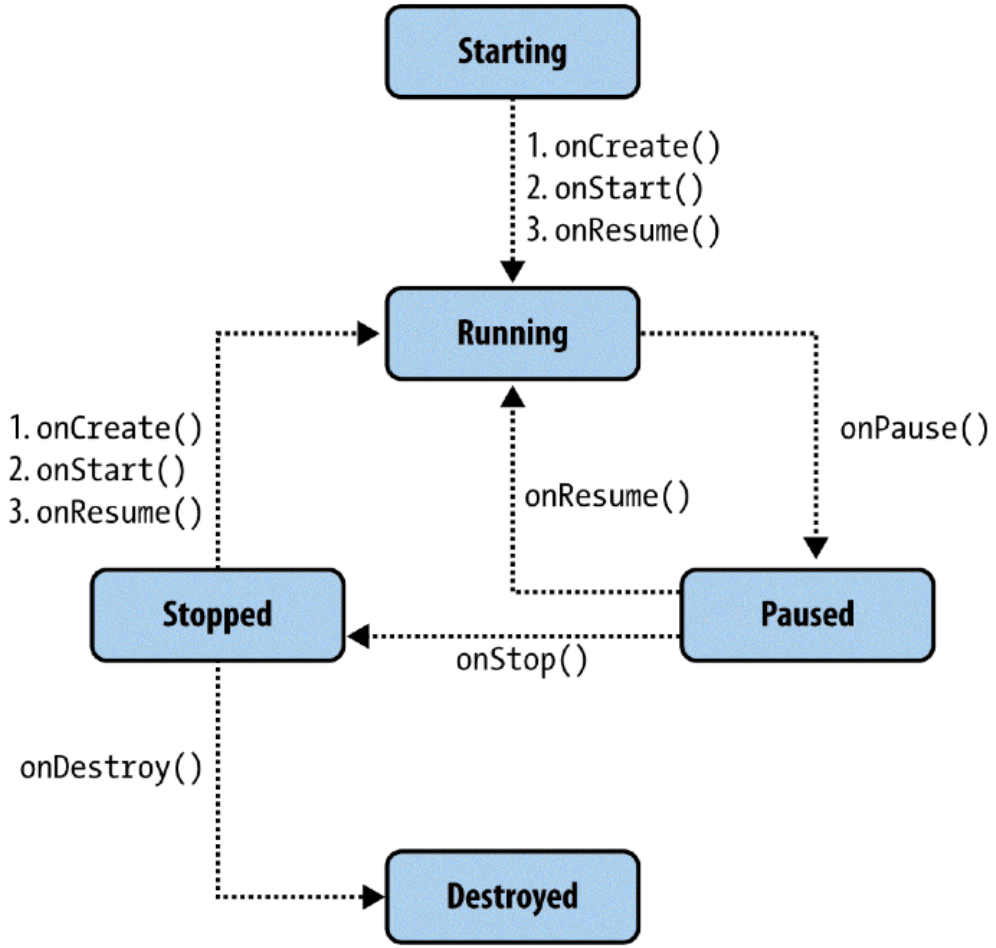


Figure 4-1. Activity life cycle

Intents (IT)

- Características principais
  - Mensagens enviadas entre os blocos da aplicação
  - Disparam uma activity na inicialização
  - Informam o serviço para começar ou terminar
  - Intents são assíncronas, o código que as envia não precisa aguardar que elas concluam

Intents (IT)

- Podem ser explícitas ou implícitas
  - Explícita: especifica que componente deve receber a msg.
  - Implícita: qualquer aplicação que oferece o serviço pode receber
  - Se implícita, o sistema lista e app escolhe. Pode definir uma *default*
  - Permite baixar novas app e trocar serviços usados pela sua app

Intents (IT)

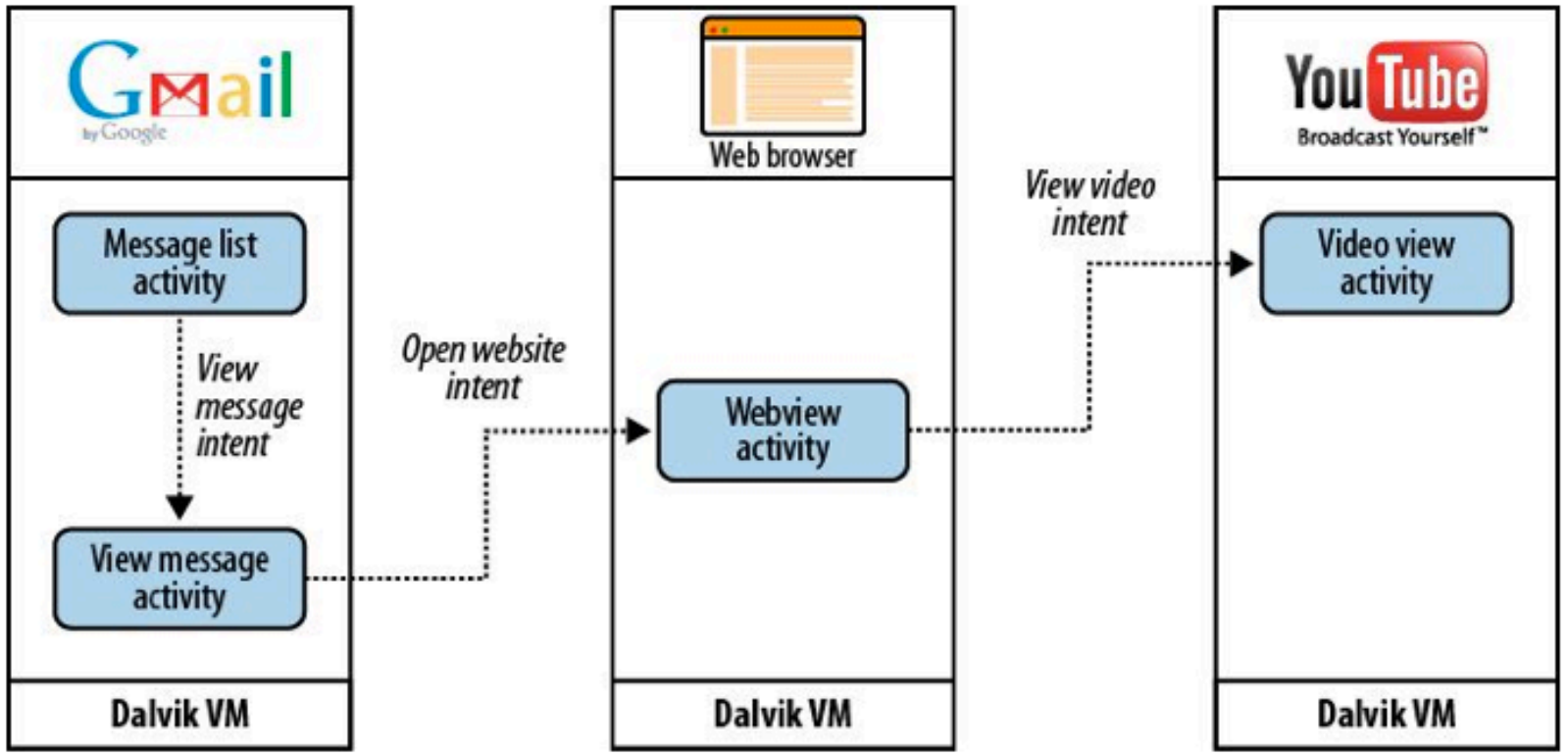


Figure 4-2. Intents

Services (SV)

- Características principais
  - Executam em background
  - Não possuem qualquer componente de UI
  - Executam mesmas ações de activities, mas sem UI
  - Úteis quando desejamos executar por algum tempo, independente do que está na tela
  - NOTA: Não confundir com serviços de baixo nível do Linux
- Exemplo
  - Um tocador de música quando usando outras apps

SV: Life Cycle

Mais simples que o AM

Inicia ou Finaliza:

- Starting
- Running
- Destroyed

Projetista tem mais controle dos recursos (CPU e bateria)

Precisa cuidar do uso adequado

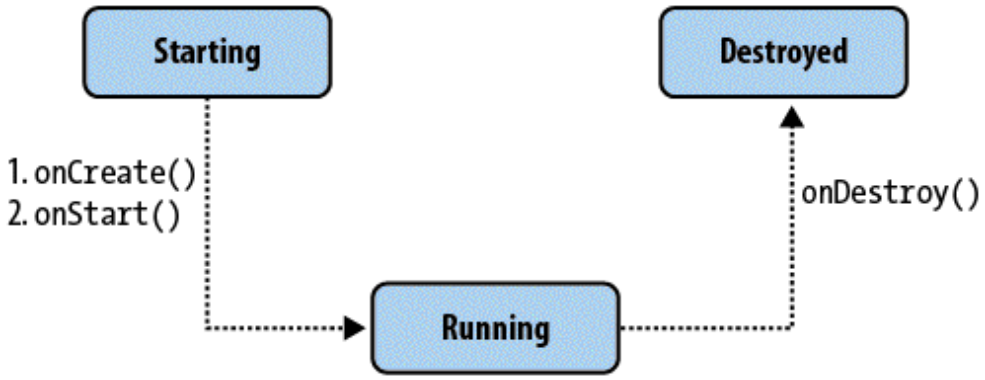


Figure 4-3. Service life cycle

Activities e Services

- Threads
  - Services executam em background mas não necessariamente em uma tthread separada
  - Default: Services e Activities executam na mesma thread principal da aplicação, chamada UI thread
  - Dica importante:
    - Usar thread separada para quaisquer operações potencialmente longas (ex. chamada de rede)
  - De outra forma a UI vai ficar muito lenta

Content Providers (CP)

- Características principais
  - Interfaces para partilhar dados entre aplicações
  - Android usa modelo de sand-box isolando apps
  - CP passa dados persistentes entre aplicações
  - CP API obedece aos princípios de CRUD para uso de dados (Create, Read Update and Delete)
  - Métodos bem simples:

    insert()

    update()

    delete()

    query()

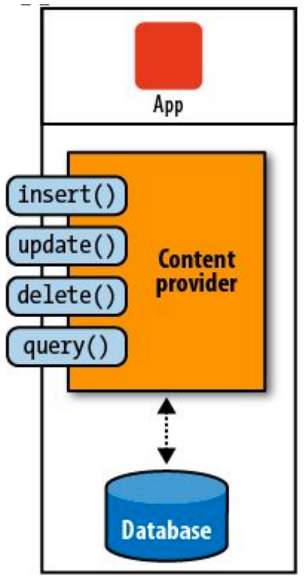


Figure 4-4. Content provider

Content Providers (CP)

- Exemplos
  - Contacts Provider: expõe contatos do usuário a todas apps
  - Settings Provider: expõe configuração do sistema a todas apps
  - Media Store: armazena e expõe fotos, músicas para todas apps

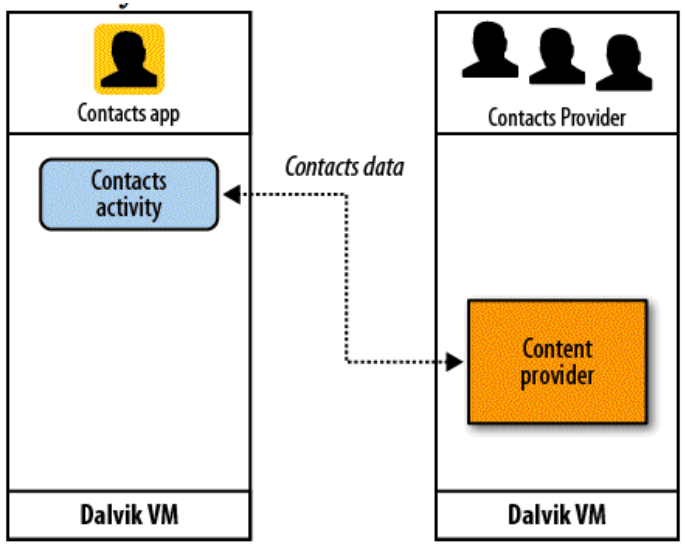


Figure 4-5. Contacts application using Contacts Provider to get the data

Content Providers (CP)

- Separação de papéis entre Activity e CP
  - Activity não possui contatos do usuário
  - CP não possui qualquer interface com usuário
  - Flexibiliza em muito o projeto
- Exemplos:
  - Instalar nova app para livro de endereços, usando os mesmos dados do Contacts Provider
  - Instalar novos *widgets* na tela principal que permite alterar os ajustes do Settings Provider (ex. WiFi on and off).
  - Fabricantes usam muito isto para adicionar suas apps sobre o Android padrão

Broadcast Receivers (BR)

- Características principais
  - Implementação Android do modelo Observer Pattern
  - BR é um pedaço de código dormente que acorda quando ocorre um evento que ele assinou em um Broadcast
  - Usado para iniciar activities, services, etc.
- Exemplos
  - Muito usado pelo próprio sistema
  - Chegada de um SMS, Bateria ficando baixa, sistema inicializado
  - No caso do Yamba, iniciar o sistema de update quando do boot
  - Subscrever o seu BR ao Broadcast que informa que o sistema completou o boot

Application Context (AC)

- Características principais
  - Ambiente e processo da App
  - Componentes anteriores executam dentro de um AC
  - Permite que eles partilhem dados e recursos
  - Criada quando o primeiro componente da App é iniciado
  - Viva enquanto a App viver independente das activities
- Referência para AC
  - *Context.getApplicationContext()*, ou
  - *Activity.getApplication()*

Aulas

- Ch1: Android Overview
- Ch2: The Stack
- Ch3: Quick Start
- Ch4: Main Building Blocks
- Ch5: Yamba Project Overview
- Ch6: Android User Interface
- Ch7: Preferences, Project Overview, Filesystems, Menus and Intents
- Ch8: Services
- Ch9: Database
- Ch10: List Adapters
- Ch11: Broadcast receivers

Yamba

- Características
  - *Yet Another Micro Blogging App*
  - Semelhante ao Twitter
  - App familiar à maioria dos alunos
  - Permite conectar ao serviço, ver status dos amigos e atualizar o próprio status
  - Cobre a maioria dos blocos vistos anteriormente

Friends Timeline

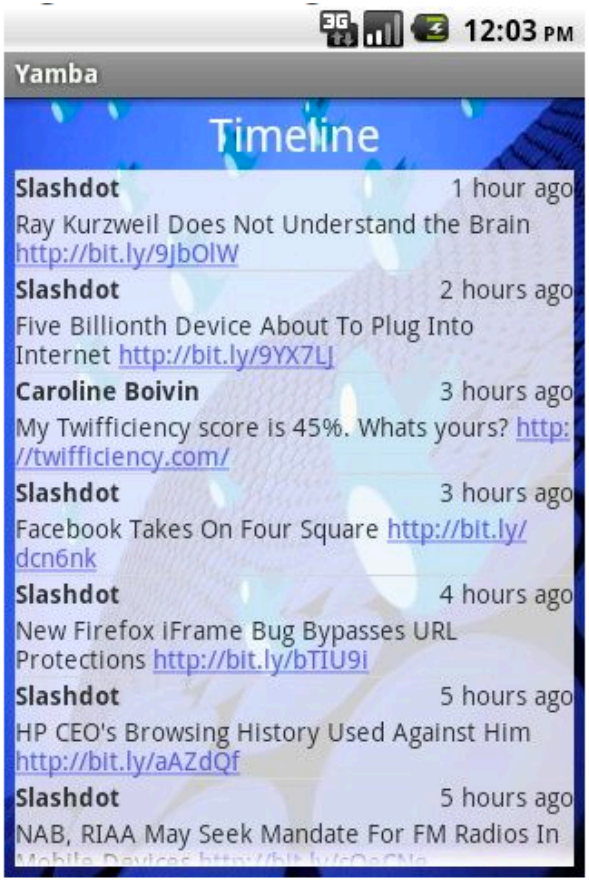


Figure 5-1. List of status messages from other people, called a timeline

User Status Update

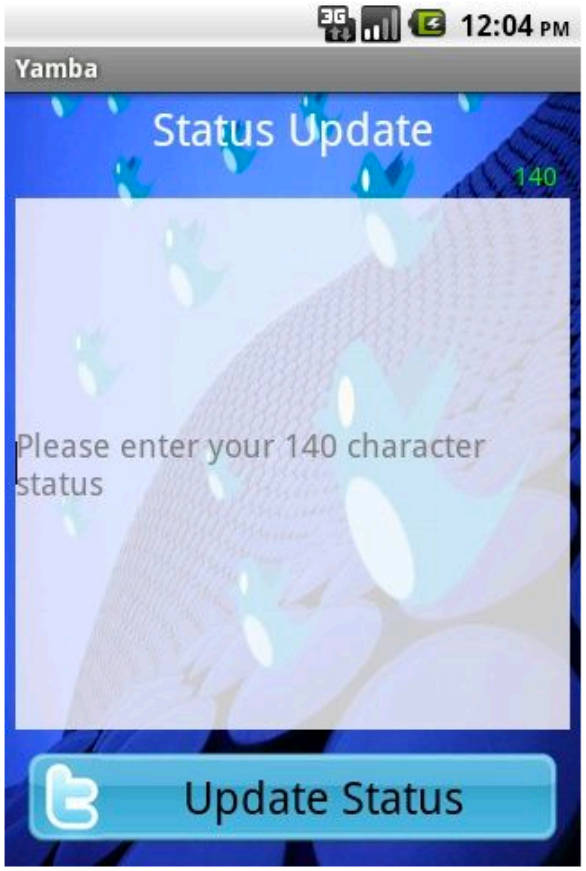


Figure 5-2. Screen where the user can enter a status message

User Preferences

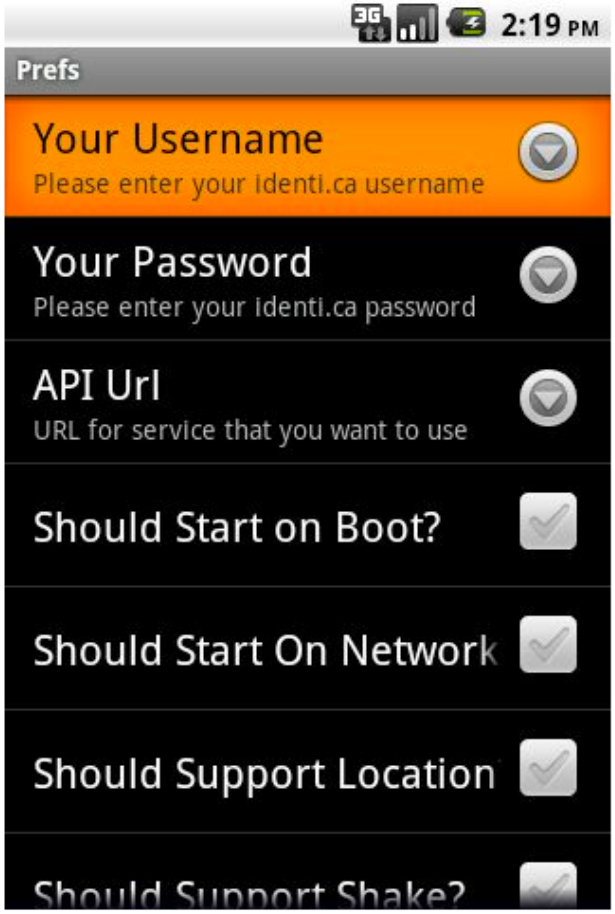
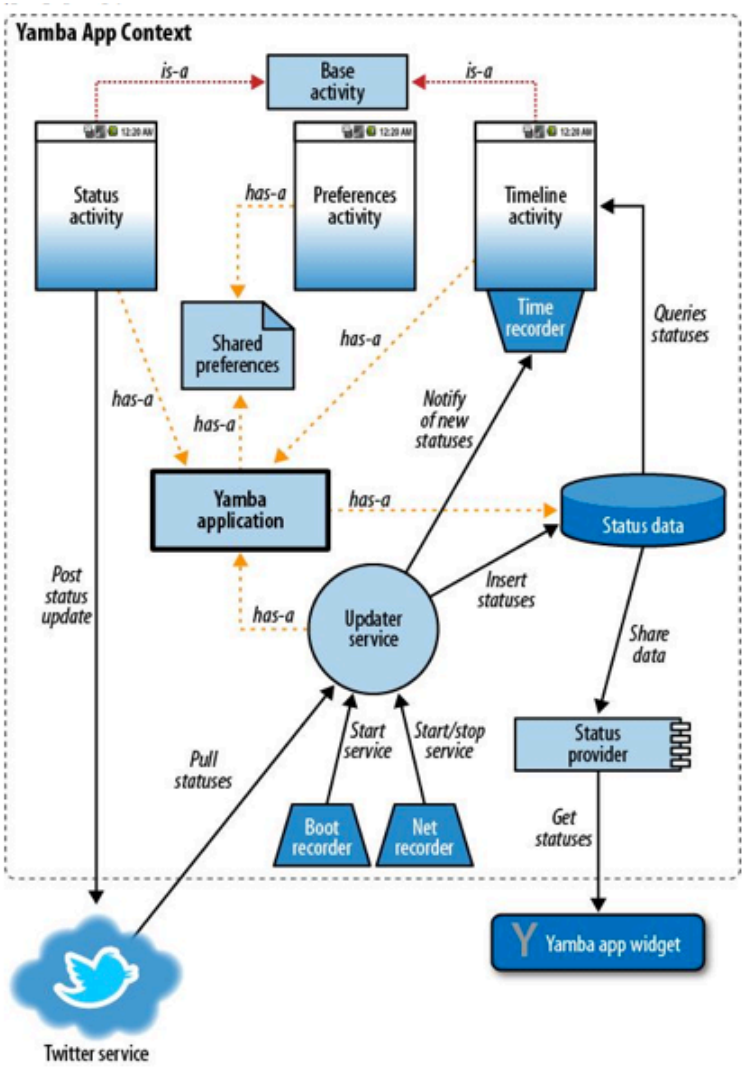


Figure 5-3. User preferences

Yamba: Filosofia do Projeto

- Incremental
  - Vamos acompanhando em lab. o desenvolvimento
  - Gradualmente crescendo em funcionalidade e complexidade
- Sempre completo
  - Projetando em pedaços (activities, services, broadcast receivers)
  - App precisa sempre funcionar, mesmos em etapas intermediárias do projeto.
- Refatoração de código
  - Enfatizar reuso sempre
  - Livro usa refino do projeto de uma solução para outra melhor

Yamba Project Overview



Aulas

- Ch1: Android Overview
- Ch2: The Stack
- Ch3: Quick Start
- Ch4: Main Building Blocks
- Ch5: Yamba Project Overview
- Ch6: Android User Interface
- Ch7: Preferences, Project Overview, Filesystems, Menus and Intents
- Ch8: Services
- Ch9: Database
- Ch10: List Adapters
- Ch11: Broadcast receivers