AccessibilityFramework - Documento de referência de uso

O framework de acessibilidade foi desenvolvido com o intuito de facilitar o desenvolvimento de aplicações que usem ou integrem um ou mais recursos de hardware dos dispositivos móveis atuais com Android. Esses recursos compreendem a câmera, o microfone, o gps e demais sensores presentes no dispositivo.

Os arquivos estão estruturados como um projeto executável no ambiente Android Studio 3.0, com sistema de compilação gradle 3.0. Esse projeto executável visa demonstrar o funcionamento dos principais componentes do framework, a saber:

1. aplicativo de câmera personalizada: aplicativo com uma pré-visualização da câmera e um botão de obturação, que captura e salva a imagem na maior resolução do aparelho e gera uma miniatura da foto no canto inferior direito
    - Classe do projeto responsável pelas funcionalidades:
        i. CustomCameraActivity, no diretório view/camera
        ii. CameraPreview, no diretório view/camera
2. aplicativo de câmera padrão: abre o aplicativo padrão do dispositivo para tirar fotos e gravar vídeos. A foto ou vídeo capturados são mostrados em um componente de pré-visualização. O arquivo é salvo no dispositivo pela aplicação padrão
    - Classe do projeto responsável pelas funcionalidades:
        i. DefaultCameraActivity, no diretório view/camera
3. microfone: aplicação para gravação e playback de áudio gravado.
    - Classe do projeto responsável pelas funcionalidades:
        i. MicrophoneActivity, no diretório microphone/audiorecorder
4. aplicativo voz para texto: usa o microfone para reconhecer o áudio falado e o transforma em texto, usando a API do Google. O idioma padrão é o português.
    - Classe do projeto responsável pelas funcionalidades:
        i. SpeechRecognizer, no diretório microphone/speechtotext
5. aplicativo sensores: exibe uma listagem com todos os sensores disponibilizados pelo fabricante do aparelho. Nem todos os sensores listados possuem necessariamente documentação Android de como utilizá-los. Nesses casos, é necessário consultar a própria documentação do fabricante. O aplicativo permite selecionar o sensor, a fim de visualizar maiores informações sobre ele (quando disponível na documentação), como descrição, unidade de medida, consumo de bateria e dados crus (sem processamento) do sensor

    - Classe do projeto responsável pelas funcionalidades:
        i. SensorActivity, no diretório view
6. aplicativo de localização: acessa os recursos de localização do aparelho e identifica onde o dispositivo se encontra, com a maior precisão possível. Para tanto, normalmente associa dados da internet com dados do gps do dispositivo. Exibe a coordenada do dispositivo na forma *(latitude, longitude)* e o respectivo endereço, na forma textual
    - Classe do projeto responsável pelas funcionalidades:
        i. LocationActivity, no diretório positionsensor/gps
7. aplicação de localização contínua: análogo ao anterior, mas busca a coordenada do dispositivo a cada 10 segundos (tempo configurável). Exibe o endereço na forma textual
    - Classe do projeto responsável pelas funcionalidades:
        i. ContinuousLocationActivity, no diretório positionsensor/gps

COMO USAR O FRAMEWORK

O primeiro passo é carregar o framework como um projeto, usando o comando File -> Open e selecionando o projeto. O projeto será compilado e construído automaticamente.

O projeto encontra-se no link abaixo.
https://www.dropbox.com/s/2vh4zymxmjswjpp/AccessibilityFramework.zip?dl=0

A aplicação-exemplo se inicia por meio da classe MainActivity, que está associada à tela com as opções do usuário. Cada opção direciona para uma das activities citadas na seção anterior.

Os códigos-fontes do framework estão dispostos de modo a facilitar que o desenvolvedor crie aplicações seguindo o modelo MVC (Model View Controller). Nesse padrão de projeto, existe separação de responsabilidades entre componentes de interface com o usuário (View ou camada de visualização) e os componentes de armazenamento de dados (Model ou modelo). A comunicação entre a interface e o modelo, bem como o processamento das operações realizadas pelo usuário, é realizada pela camada de controle (Controller). Na aplicação exemplo, não estamos usando as classes de modelo.

Tendo em vista que, ao contrário do que ocorre com o desenvolvimento de aplicativos para iOS, que é nativamente MVC e gera códigos mais padronizados, o Android deixa o desenvolvedor completamente livre para desenvolver os aplicativos. Portanto, sair da estrutura proposta pelo framework é possível, embora não recomendável. Como as activities são classes que se associam diretamente com definições XML da interface, elas estão associadas à camada de visualização. A Figura 1 fornece uma visão geral do MVC do framework.

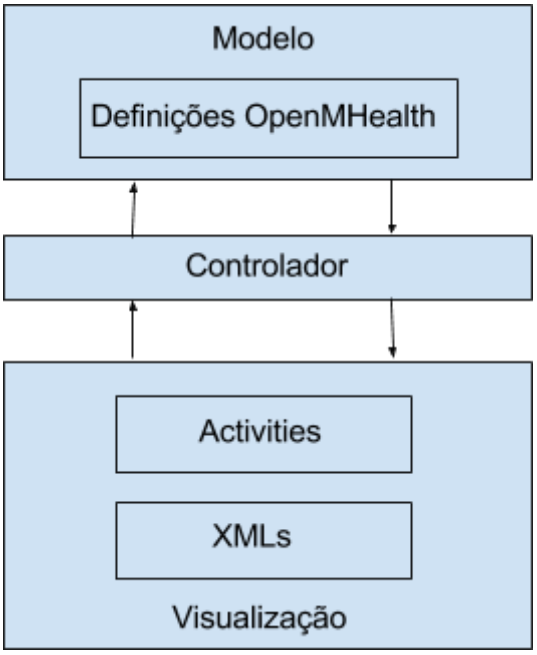

Figura 1. Padrão MVC no Framework

As activities são as principais classes do framework, e contém os atributos e métodos principais para cada componente. Os diretórios e respectivas activities são ilustrados na Figura 2.

Todas as classes do pacote view são activities. Contudo, as únicas que se comunicam com a camada de visualização são as activites do projeto-exemplo. As demais contém o esqueleto básico de aplicações que o desenvolvedor queira implementar.

TRABALHANDO COM SENSORES

O acesso aos sensores em Android é realizado por meio da implementação dos métodos da interface *SensorEventListener*. O método *onSensorChanged()* identifica quando há uma modificação dos valores do sensor, ao passo que o método *onAccuracyChanged()* identifica quando há alterações de precisão nos dados do sensor. Os sensores devem ser registrados

para que possam usados, e precisam ser liberados após o uso, para que não usem desnecessariamente os recursos do sistema, como processamento e bateria.

A classe SensorActivity contém atributos para todos os sensores da documentação oficial Android, exceto o gps. Ela demonstra o uso desses sensores e implementa os métodos necessários da interface. Também possui os registros e liberações do sensor de acordo com o ciclo de vida própria activity. O atributo *timeToUpdate* indica o tempo de atualização dos dados dos sensores, em milissegundos. Use essa classe se você deseja trabalhar com mais de um sensor ao mesmo tempo. Em geral, a única tarefa que o desenvolvedor necessita realizar é implementar o método work() da classe SensorActivity e retirar as referências a sensores que não irá utilizar.

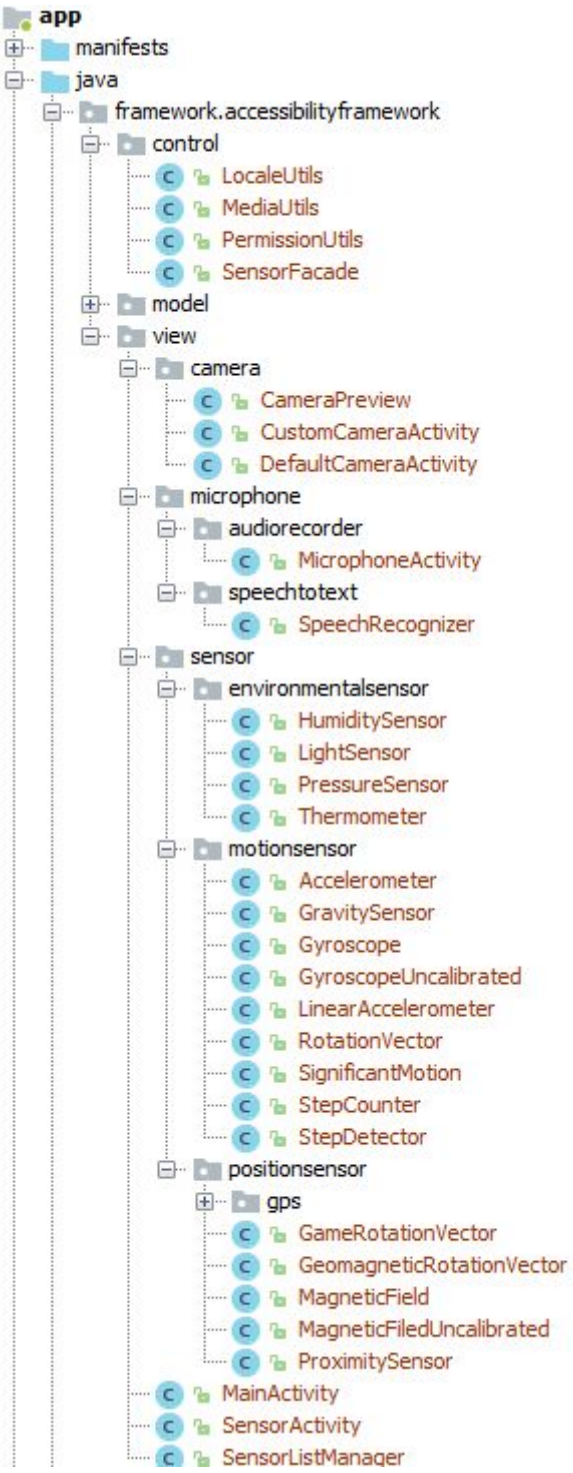

Figura 2. Estrutura de diretórios e classes da camada de visualização

Caso o intuito seja utilizar um único sensor por vez, separadamente, é possível usar a respectiva activity do sensor desejado. Por exemplo, se o objetivo é trabalhar exclusivamente com o sensor de luminosidade, a classe *LightSensor* do pacote *environmentsensor* é a mais adequada.

É recomendável ler os comentários nos cabeçalhos da classe usada, porque eles contém informações básicas sobre o componente e seu uso.

Caso o desenvolvedor queira realizar alguma operação sobre os dados crus dos sensores, ele pode fazê-lo usando funcionalidades da classe *SensorFacade* do pacote control. Além disso, novas funcionalidades podem ser implementadas pelo desenvolvedor nessa classe, sem alterar a arquitetura.

USANDO A CÂMERA

APLICATIVO PADRÃO

Caso o intuito seja tirar fotos ou gravar vídeos utilizando o aplicativo padrão do dispositivo móvel, a classe a ser utilizada é a *DefaultCameraActivity*, do pacote camera. A função takePicture() tira uma nova foto, ao passo que a recordVideo() grava um novo vídeo.

Tanto a função para tirar fotos quanto a de gravar vídeo permite o uso e manipulação da imagem no tamanho original e/ou em miniatura, de tamanho ⅕ do tamanho original da imagem. Esses arquivos são *Bitmaps* da imagem salva no dispositivo móvel. O diretório de salvamento é personalizável e armazenado no atributo *diretorio* da classe.

As permissões de câmera e salvamento no cartão SD estão implementadas, de modo que usuários com Android 6.0 ou superior poderão utilizar os recursos da classe normalmente.

APLICATIVO PERSONALIZADO

A classe responsável pela câmera personalizada é a *CustomCameraActivity*, do pacote camera. Essa classe implementa uma pré-visualização da câmera por meio de funcionalidades e eventos da classe *CameraPreview*. Assim como no caso do aplicativo padrão, há o atributo *directory* para o local de gravação do arquivo.

A função *setCameraDisplayOrientation()* posiciona a imagem na orientação correta, após identificar a orientação da câmera. Essa funcionalidade é importante porque, ao tirar uma foto e girar o dispositivo, a orientação da foto não muda, o que faz com que ela seja posteriormente armazenada sem rotação adequada (de lado ou de cabeça para baixo, por exemplo).

A função *rotate()* gira a foto usando matrizes de rotação. Usada após o a função *setCameraDisplayOrientation()*, posiciona a imagem na orientação correta para salvamento.

A função *mirror()* realiza o espelhamento da imagem, para o caso de fotos que, após tiradas, aparecem espelhadas. Esse é um problema reportado por vários desenvolvedores de aplicativos Android, e aparentemente depende do fabricante.

A função *createMiniature()* gera uma miniatura da foto, de acordo com as novas proporções passadas por parâmetro.

Use essa classe caso o intuito seja usar a câmera como plano de fundo de uma aplicação. Nesse caso, a activity deve ser transformada em um serviço, conforme consta na Seção CONSIDERAÇÔES FINAIS deste documento. Contudo, esteja ciente de que seu uso consome a bateria do dispositivo e que normalmente a gravação estará ocorrendo de forma não-transparente ao usuário, que dificilmente possui ciência das funcionalidades sendo realizadas em segundo plano.

As permissões de câmera e salvamento no cartão SD estão implementadas, de modo que usuários com Android 6.0 ou superior poderão utilizar os recursos da classe normalmente.

USANDO O MICROFONE

PARA GRAVAÇÃO E PLAYBACK

O acesso ao microfone é feito por meio da classe *MicrophoneActivity.* As funcionalidades do microfone são executadas de acordo com as funcionalidades da classe MediaRecorder, que possui seu próprio ciclo de vida.

As funções de playback são executadas pelas funções *startPlaying()* e *stopPlaying()*, que inciam o playback e param o playback, respectivamente.

As funções de gravação se iniciam com *startRecording()* e param com *stopRecording().*

É possível transformar essa classe como um serviço, mas isso não é recomendado pela documentação do Android, por conta de falta de visibilidade para o usuário e potencial alto consumo de bateria.

As permissões de gravação de áudio e salvamento no cartão SD estão implementadas, de modo que usuários com Android 6.0 ou superior poderão utilizar os recursos da classe normalmente.

PARA RECONHECIMENTO DE FALA

A funcionalidade de reconhecimento de fala e posterior transformação em texto é feita por meio do uso da API de speech-to-text disponibilizada pelo Google, e está implementada na classe *SpeechRecognizer*..

O atributo *recognizedSentences* contém uma lista com todas as sentenças reconhecidas, após o usuário dizer uma sentença qualquer. As sentenças da lista estão ordenadas da mais parecida com o que o usuário disse para a menos parecida, ou seja, a posição 0 contém a frase mais próxima ao que foi dito.

As permissões de gravação de áudio e acesso à internet estão implementadas, de modo que usuários com Android 6.0 ou superior poderão utilizar os recursos da classe normalmente. O reconhecimento por meio da API é feito online.

USANDO RECURSOS DE LOCALIZAÇÃO

O Android disponibiliza diferentes abordagens para a obtenção de localização do usuário, de acordo com duas possíveis permissões. O desenvolvedor deve escolher a permissão mais adequada de acordo com a necessidade da aplicação sendo implementada. As permissões possíveis são as seguintes:

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

A permissão acima autoriza o sistema Android a usar informações de localização sem uso do gps, ou seja, usando apenas informações online, por exemplo. A precisão é de 100m, sendo utilizada quando se deseja um aplicativo com precisão de um quarteirão de uma cidade.

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

A permissão acima usa todos os recursos possíveis no dispositivo, a fim de fornecer a localização mais precisa possível. Quando o dispositivo contém gps, ele é utilizado. A precisão varia de acordo com o gps, mas costuma ser em torno de 20m.

No framework de desenvolvimento, a permissão utilizada é a ACCESS_FINE_LOCATION, a qual pode ser alterada pelo desenvolvedor.

CAPTURANDO A LOCALIZAÇÃO ATUAL

Caso o intuito seja identificar a posição atual do usuário, a classe *LocationActivity* pode ser utilizada. Os atributos *latitude* e *longitude* contém a latitude e a longitude do usuário e, juntos, formam uma coordenada que pode ser transformada em um endereço. O endereço, por sua vez, está armazenado no atributo *address* da mesma classe.

CAPTURANDO A LOCALIZAÇÃO DE TEMPOS EM TEMPOS

Caso o usuário tenha que ser monitorado por um período de tempo, a classe *ContinuousLocationActivity* pode ser utilizada. A cada verificação, o atributo *address* é atualizado. O desenvolvedor necessita, portanto, salvar o registro dos endereços que deseja em alguma lista ou outra estrutura de dados da sua escolha.

O atributo *timeBetweenUpdates* armazena o tempo em milissegundos entre cada verificação de localização do usuário, e pode ser alterado oportunamente.

A atualização da posição do usuário inicializa com o início da classe e só para quando se chama a função *stopLocationUpdates()*. O framework chama a função de parada quando a classe não está mais em uso, por exemplo quando o usuário muda de tela na aplicação.

As funcionalidades de localização podem ser realizadas em plano de fundo por meio de serviços.

As permissões de localização e internet estão implementadas, de modo que usuários com Android 6.0 ou superior poderão utilizar os recursos da classe normalmente. O reconhecimento por meio da API é feito online.

CONSIDERAÇÔES FINAIS

Idealmente, o desenvolvedor interessado em utilizar o framework construirá sua aplicação a partir do próprio framework. Contudo, para projetos já iniciados, é possível usar apenas as classes necessárias para cada projeto, o que, evidentemente, altera a arquitetura do projeto, copiando individualmente os componentes. Outra alternativa é usar o framework todo como um módulo do projeto, o que permite acessar os componentes do projeto separadamente, como se ele fosse um projeto à parte, um conjunto de bibliotecas externas. Mais detalhes sobre essa possibilidade são explicadas pela documentação oficial do sistema operacional, disponível em https://developer.android.com/studio/projects/android-library.html?hl=pt-br.

As activities são classes que se associam a interfaces de sistemas, mas muitas vezes o desenvolvedor deseja trabalhar com interfaces por meio de fragments ou simplesmente não precisa de uma interface explícita, deixando as funcionalidades como parte de um serviço Android.

Para transformar as classes em fragments, basta mudar a extensão de AppCompatActivity para Fragment (ou uma de suas classes filhas), alterando os métodos do ciclo de vida de acordo com as necessidades do fragmento desejado.

Para transformar as classes activities em serviços, basta mudar a extensão de AppCompatActivity para Service, alterando o método *onCreate()* para *onBind()*.

A camada de modelo do framework contém diversos esquemas para salvamento de informações médicas de usuários/ pacientes. Essas informações seguem o padrão proposto pelos profissionais da saúde da plataforma OpenMHealth. Portanto, são uma recomendação de como salvar os dados para cada informação, como por exemplo frequência cardíaca e pressão arterial.

Ainda faltam operações a serem implementadas no framework: 1) integração com a infraestrutura do ESPIM, para permitir comunicação com o especialista de pacientes; 2) inclusão de comunicação por bluetooth.