João Victor Sá de Araújo - 202304587167
Estácio São Luís Polo Cohama
Interação Com Sensores de Smartphones e Wearebles *– 2026.1*

Lidando com sensores em dispositivos móveis

1. Objetivo da Prática

O objetivo desta missão prática foi explorar o desenvolvimento de aplicações para Wear OS utilizando o framework Flutter, com foco em acessibilidade e comunicação interna para a empresa Doma. A atividade visou a implementação de funcionalidades de áudio em tempo real, como alertas de segurança e assistência para funcionários com necessidades especiais. Ao final, implementou-se uma interface otimizada para smartwatches, demonstrando a capacidade de integrar código nativo Kotlin (para gerenciamento de hardware de áudio) com a camada de interface Dart, garantindo uma solução inclusiva e funcional.

2. Análise Crítica e Procedimentos

A implementação seguiu uma abordagem de integração híbrida, unindo a agilidade do Flutter para a interface com a precisão do código nativo Android para o controle de dispositivos de saída de áudio.

Passos da Implementação:

1. Configuração da Estrutura Base e Wearable: Utilizou-se o MaterialApp com o tema Brightness.dark para otimizar o consumo de bateria em telas OLED e garantir o contraste necessário para usuários com baixa visão. O

layout foi centralizado para se adaptar a visores circulares de smartwatches.

2. Implementação de Saídas de Áudio (Nativo): Através de um MethodChannel, integrou-se código em Kotlin para acessar o AudioManager. Foi desenvolvida a lógica para detectar dinamicamente se o dispositivo possui alto-falantes integrados (TYPE_BUILTIN_SPEAKER) ou se há fones Bluetooth conectados (TYPE_BLUETOOTH_A2DP).

3. Detecção Dinâmica e Callbacks: Implementou-se o registerAudioDeviceCallback no lado nativo para monitorar em tempo real a conexão ou desconexão de periféricos de áudio, permitindo que o app reaja instantaneamente a mudanças no hardware.

4. Facilitação de Conectividade: Para melhorar a experiência do usuário, configurou-se uma Intent com ACTION_BLUETOOTH_SETTINGS. Isso permite que, caso uma saída de áudio não seja detectada, o funcionário seja direcionado automaticamente para as configurações de pareamento, evitando mensagens de erro complexas.

5. Reprodução de Áudio e Acessibilidade: No Flutter, utilizou-se o pacote audioplayers para a execução de alertas de segurança e instruções de treinamento. A estrutura de pastas foi organizada no pubspec.yaml para incluir recursos sonoros locais, garantindo que as notificações funcionem mesmo sem conexão constante com a internet.

6. Otimização de Interface (UI): Substituiu-se botões padrão por elementos de toque ampliado (GestureDetector e círculos concêntricos), facilitando a interação em telas reduzidas e garantindo que comandos de voz e alertas possam ser acionados com precisão por qualquer colaborador.