Aplicativo móvel que auxilia pessoas com deficiência visual no deslocamento de ambientes internos

Aluno(a): Guilherme Barth

Orientador: Dalton Solano dos Reis

FURB
UNIVERSIDADE DE BLUMENAU

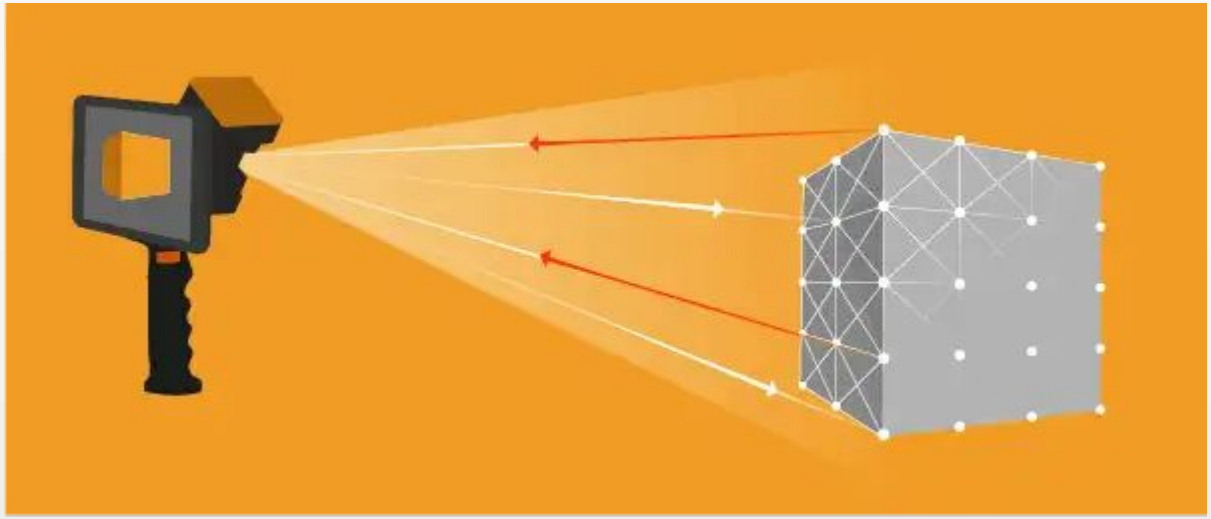
Roteiro

Introdução
Objetivos
Fundamentação teórica
Trabalhos Correlatos
Requisitos
Especificação
Implementação
Análise de Resultados
Conclusões e Sugestões

Introdução

“Para as pessoas sem deficiência, a tecnologia torna as coisas mais fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis.” SILVA, 2019.

- Laser de plataforma: elevada frequência de repetição
- Tempo de retorno do laser: é a distância



Objetivo geral

- Disponibilizar um aplicativo na plataforma iOS que auxilia pessoas com deficiência visual a identificar objetos durante o seu deslocamento utilizando o LiDAR e o ML Kit

Objetivos específicos

- Avaliar se a medida de distância em que o objeto se encontra do dispositivo móvel é coerente com a realidade

- Fazer a interação do usuário com o aplicativo por comandos de voz

Fundamentação Teórica

- Room Plan
- ML Kit
- Speech to Text e Text to Speech

Room Plan

- Fortemente utilizado para ambientes internos
- Possui o auxílio de Machine Learnings como o RealityKit
- Simula e renderiza conteúdo 3D para os apps de realidade aumentada
- Portas, aberturas, janelas, paredes e objetos

ML Kit

- SDK que permite utilizar várias ferramentas
- Reconhecimento de textos
- Reconhecimento facial
- Detecção de malha de rosto
- Reconhecimento de tinta digital
- Detecção e rastreamento de objetos

Speech to Text
Text to Speech

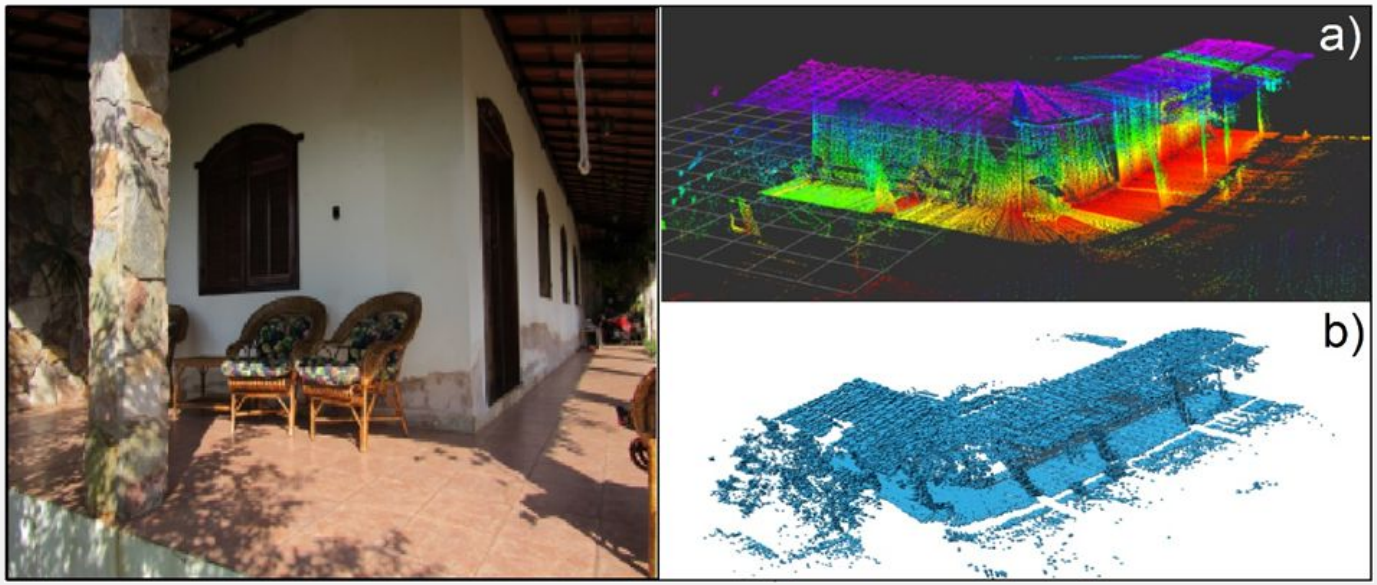
- AVFoundation - TTS
- Speech - STT

```
    to list voice folder
ACCEPTED
any
inside
inside busca
inside busca en
inside busca
inside busca en
inside busca en the shower
inside busca en the shower busca
inside busca in the shower busca
inside busca en isha busca initiate
inside busca en isha busca initiate busca
```

```
33
34      let synthesizer = AVSpeechSynthesizer()
35
36      override func viewDidLoad() {
37
38          let utterance = AVSpeechUtterance(string: "Para começar fale INICIAR BUSCA ou Start Scanning")
39          utterance.voice = AVSpeechSynthesisVoice(language: "pt-BR")
40          utterance.rate = 0.5
41
42          synthesizer.speak(utterance)
43
44          requestPermission()
45          startRecognition()
46
47      }
```

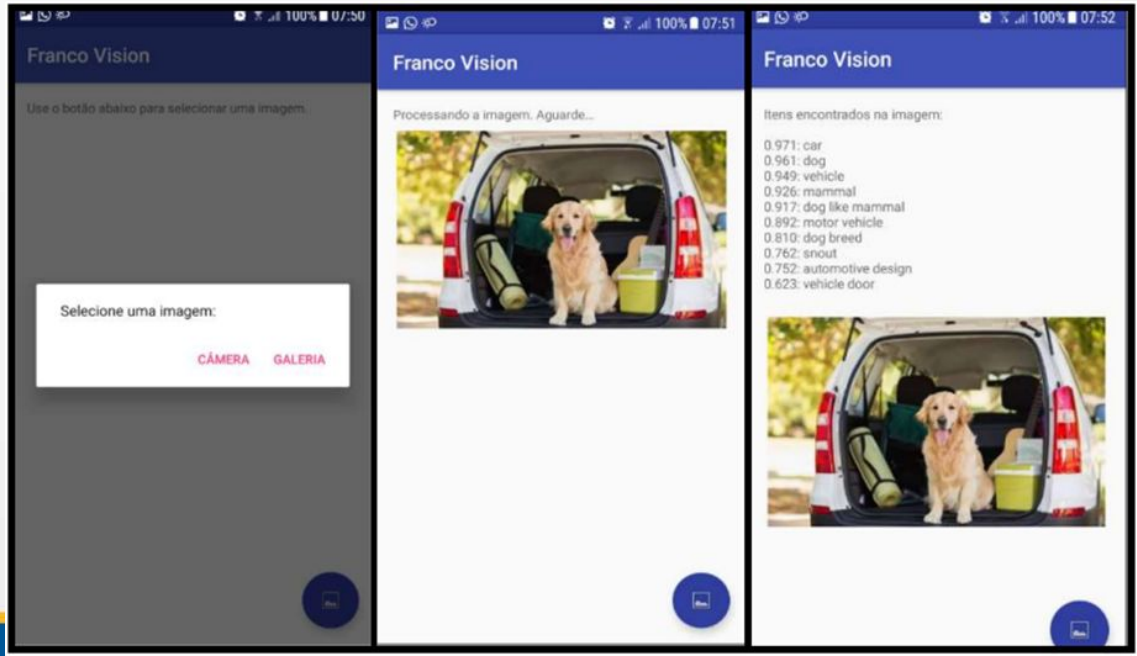
Rossi, Freitas e Reis (2019)

- Protótipo robótico
- Sensoriamento em 3D: sensor LiDAR
- Mapeamento: ambientes simulados e reais



Franco (2018)

- Instruir usuários com deficiência visual
- Aplicativo de reconhecimento de objetos
- Cloud Text-to-Speech
- API Cloud Vision



Pereira (2022)

- Comparar mapas de suscetibilidade a deslizamentos
- Algoritmo de aprendizado de máquina floresta aleatória
- Random Forest
- LiDAR e VANT

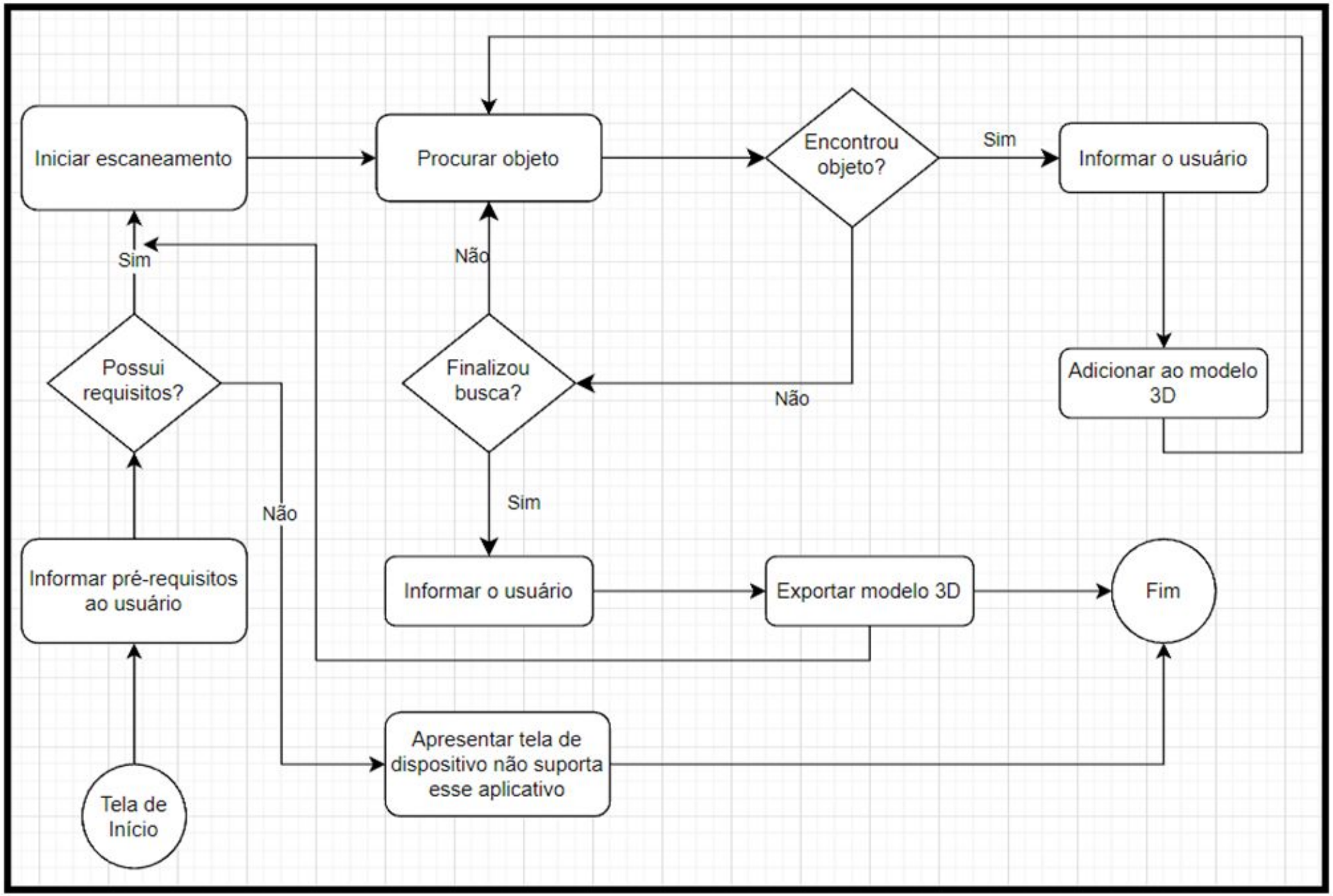
Requisitos Funcionais

- Informar ao usuário quando há um novo obstáculo no visor da câmera
- Permitir interação vocal
- Criar uma planta em 3D do ambiente escaneado
- Permitir exportar o modelo 3D do ambiente

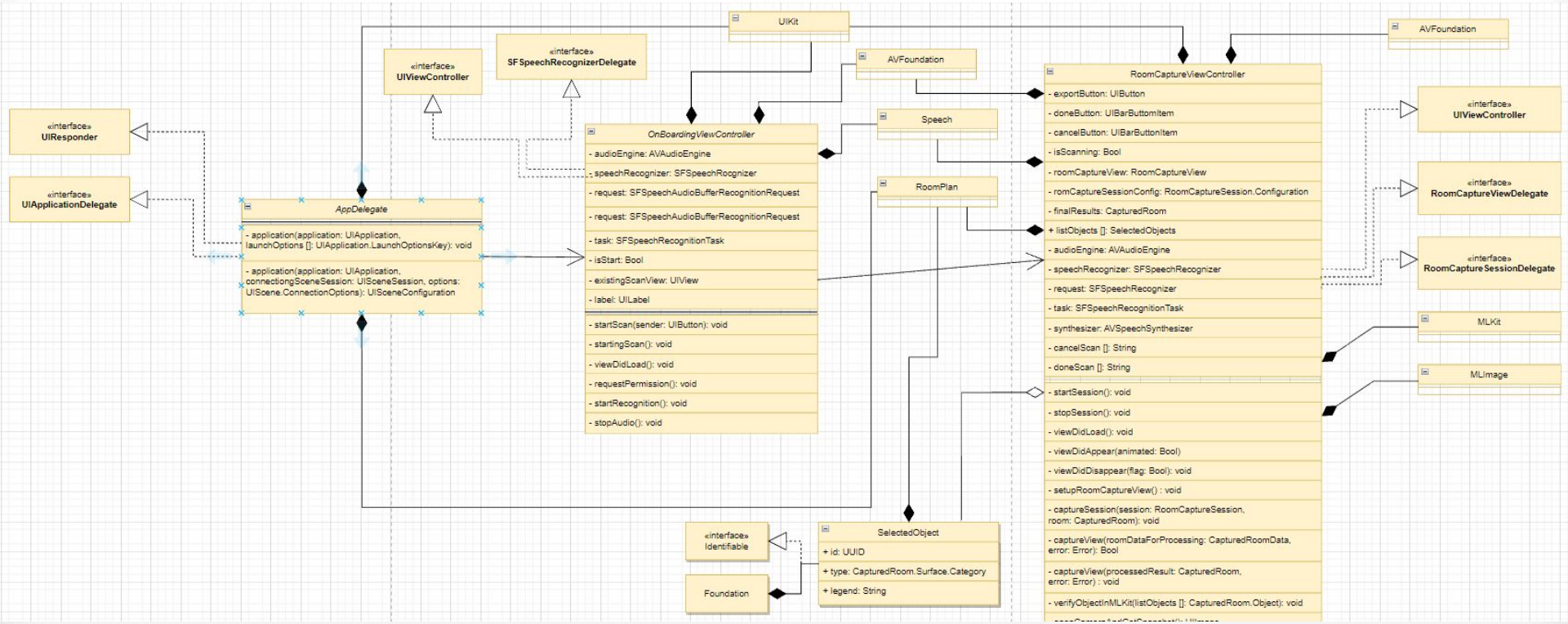
Requisitos Não Funcionais

- A detecção principal de objetos deve ser feita com a *framework* Room Plan, quando não for possível, deve utilizar o ML Kit

- Ao detectar um obstáculo, o aplicativo deve informar qual é o objeto encontrado

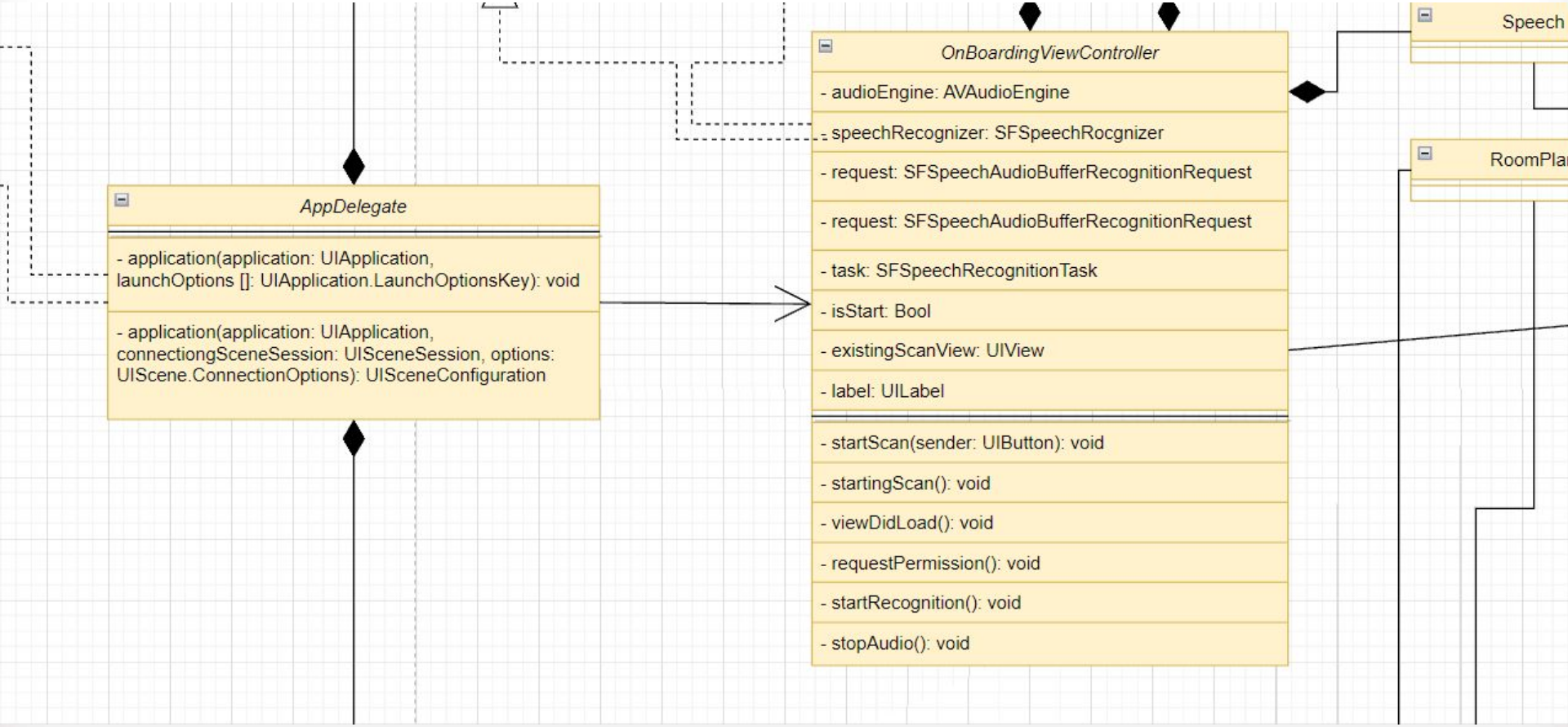

Iniciar escaneamento
Procurar objeto
Encontrou objeto?
Sim
Informar o usuário
Adicionar ao modelo 3D
Não
Finalizou busca?
Não
Sim
Possui requisitos?
Sim
Não
Informar pré-requisitos ao usuário
Informar o usuário
Exportar modelo 3D
Fim
Apresentar tela de dispositivo não suporta esse aplicativo
Tela de Início

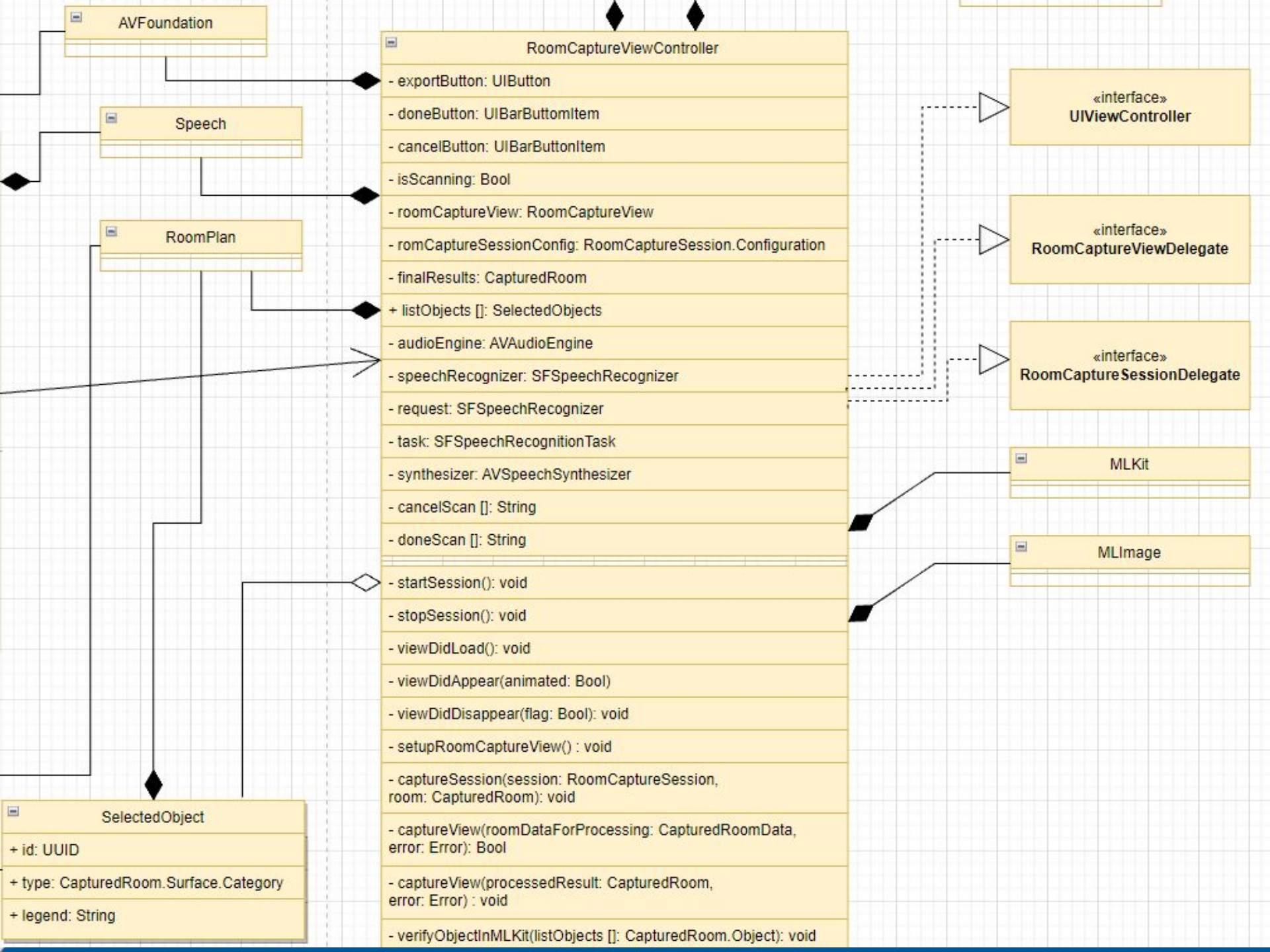
DC - Visão Geral

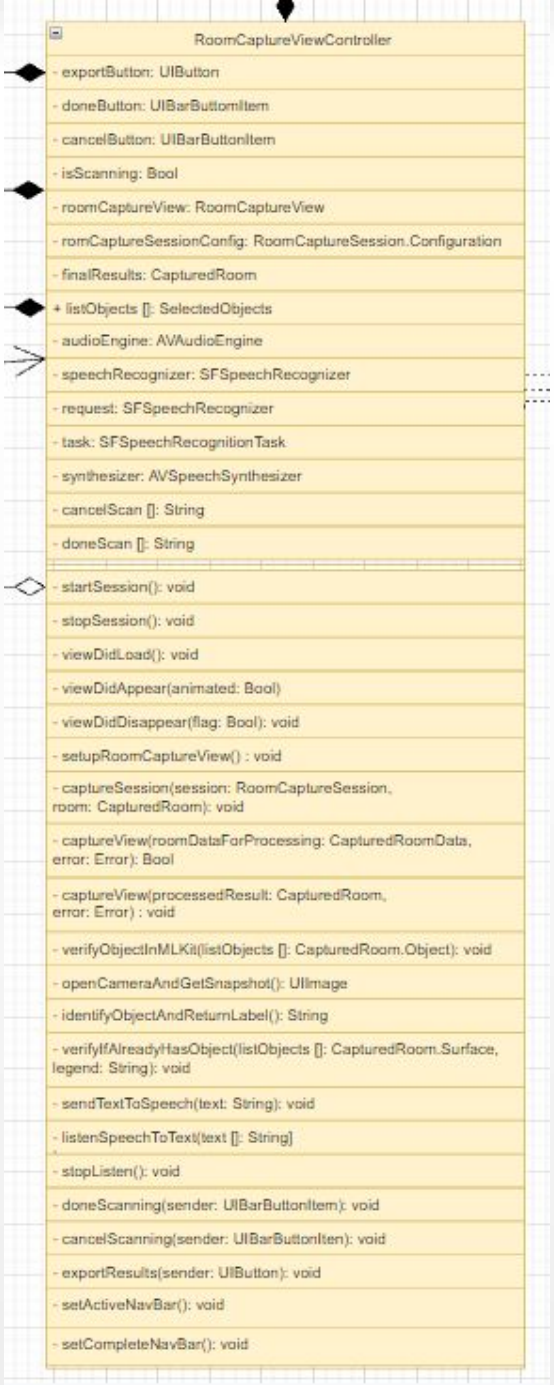


8 - Interfaces
8 - Bibliotecas
4 - Classes

OnBoardingView




AVFoundation
Speech
RoomPlan
SelectedObject
+ id: UUID
+ type: CapturedRoom.Surface.Category
+ legend: String
RoomCaptureViewController
- exportButton: UIButton
- doneButton: UIBarButtomItem
- cancelButton: UIBarButtonItem
- isScanning: Bool
- roomCaptureView: RoomCaptureView
- romCaptureSessionConfig: RoomCaptureSession.Configuration
- finalResults: CapturedRoom
+ listObjects []: SelectedObjects
- audioEngine: AVAudioEngine
- speechRecognizer: SFSpeechRecognizer
- request: SFSpeechRecognizer
- task: SFSpeechRecognitionTask
- synthesizer: AVSpeechSynthesizer
- cancelScan []: String
- doneScan []: String
- startSession(): void
- stopSession(): void
- viewDidLoad(): void
- viewDidAppear(animated: Bool)
- viewDidDisappear(flag: Bool): void
- setupRoomCaptureView() : void
- captureSession(session: RoomCaptureSession, room: CapturedRoom): void
- captureView(roomDataForProcessing: CapturedRoomData, error: Error): Bool
- captureView(processedResult: CapturedRoom, error: Error) : void
- verifyObjectInMLKit(listObjects []: CapturedRoom.Object): void
«interface» UIViewController
«interface» RoomCaptureViewDelegate
«interface» RoomCaptureSessionDelegate
MLKit
MLImage



Hardware

- iPhone 13 Pro
- MacBook Pro
- i3 Intel 2.33 Ghz

FURB
UNIVERSIDADE DE BLUMENAU

```swift
func captureSession(_ session: RoomCaptureSession, didAdd room: CapturedRoom) {

    verifyIfAlreadyHasObject(listObjects: room.doors, legend: "porta")
    verifyIfAlreadyHasObject(listObjects: room.walls, legend: "parede")
    verifyIfAlreadyHasObject(listObjects: room.openings, legend: "passagem")
    verifyIfAlreadyHasObject(listObjects: room.windows, legend: "janela")

    if room.objects.count > 0 {
        verifyObjectInMLKit(listObjects: room.objects)
    }
}

func verifyObjectInMLKit(listObjects: [CapturedRoom.Object]) {
    var alreadyHasObj = false

    for obj in listObjects
    {
        for verifyObj in self.listObjects {
            if obj.identifier == verifyObj.id {
                alreadyHasObj = true
            }
        }

        if !alreadyHasObj {
            let legend = self.identifyObjectAndReturnLabel()
            let find = "Tem um \(legend) na sua frente."
            self.sendTextToSpeech(texto: find)
            self.listObjects.append(SelectedObject(id: obj.identifier, type: nil, legend: legend))
        }
        alreadyHasObj = false
    }
}
```

```swift
func verifyObject(image: UIImage) -> String {
    var imageLabel = "objeto"

    let options: CommonImageLabelerOptions! = ImageLabelerOptions()
    options.confidenceThreshold = NSNumber(floatLiteral: Constants.labelConfidenceThreshold)

    let onDeviceLabeler = ImageLabeler.imageLabeler(options: options)
    let visionImage = VisionImage(image: image)

    visionImage.orientation = image.imageOrientation
    onDeviceLabeler.process(visionImage) { labels, error in
        guard error == nil, let labels = labels, !labels.isEmpty else {
            let errorString = error?.localizedDescription ?? Constants.detectionNoResultsMessage
            print("Erro durante a detecção da legenda: \(errorString)")
            return
        }

        let labelsInImage = labels.map { label -> String in
            return "Label: \(label.text), Confidence: \(label.confidence)"
        }.joined(separator: "\n")

        if !labelsInImage.isEmpty {
            imageLabel = labels.first?.text ?? "objeto"
        }
    }
    return imageLabel
}
```

SceneDelegate

- Criado extensão: wear var shared
- Evento de scene: willConnectTo

```
29 extension SceneDelegate {
30     static weak var shared: SceneDelegate?
31
32     func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions:
           UIScene.ConnectionOptions) {
33         Self.shared = self
34
35         guard let scene = scene as? UIWindowScene else {
36             return
37         }
38         // Save the reference when the scene is born.
39         currentScene = scene
40     }
41 }
```

```
144
145         if let sceneDelegate = UIApplication.shared.connectedScenes.first?.delegate as? SceneDelegate {
146             let snap = sceneDelegate.window?.snapshotView(afterScreenUpdates: false)
147
148             view.addSubview(snap!)
149             UIGraphicsBeginImageContextWithOptions(view!.frame.size, true, 0.0)
150
151             if let context = UIGraphicsGetCurrentContext() { view!.layer.render(in: context) }
152             screenshotImage = UIGraphicsGetImageFromCurrentImageContext()
153             UIGraphicsEndImageContext()
154         }
155
156         return screenshotImage
157     }
158
```

```
145
146         let snap = SceneDelegate.shared?.window?.snapshotView(afterScreenUpdates: false)
147
148         view.addSubview(snap!)
149         UIGraphicsBeginImageContextWithOptions(snap!.frame.size, true, 0.0)
150
151         if let context = UIGraphicsGetCurrentContext() { snap!.layer.render(in: context) }
152         screenshotImage = UIGraphicsGetImageFromCurrentImageContext()
153         UIGraphicsEndImageContext()
154
155
156         return screenshotImage
```

Captura de tela

- Scene Delegate
- Window
- UIView
- UIImage

```
97
98      func captureSession(_ session: RoomCaptureSession, didAdd room: CapturedRoom) {
99
100         verifyIfAlreadyHasObject(listObjects: room.doors, legend: "porta")
101         verifyIfAlreadyHasObject(listObjects: room.walls, legend: "parede")
102         verifyIfAlreadyHasObject(listObjects: room.openings, legend: "passagem")
103         verifyIfAlreadyHasObject(listObjects: room.windows, legend: "janela")
104
105         if room.objects.count > 0 {
106             DispatchQueue.main.async {
107                 self.verifyObjectInMLKit(listObjects: room.objects)
108             }
109         }
110     }
111
```

Análise dos Resultados

- Objetos: escaneados com sucesso
- Semelhante ao API Cloud Vision - Franco (2018)
- Diferente de Rossi, Freitas e Reis (2019)

Conclusões

- Informar o usuário que há um obstáculo no seu campo de visão e que ele está próximo
- Comunicação do aplicativo com o usuário por comandos de voz
- Determinar a distância do objeto

Sugestões

- Novas funcionalidades para o Modelo 3D: localização indoor utilizando Beacons
- Melhorar a utilização do LiDAR: trazer a localização exata do objeto
- Adaptação do reconhecimento de voz: permitir comunicação nativa

Apresentação da Implementação