Sistema Móvel para Deteção de BLE e navegação indoor

Documento de requisitos do utilizador e do sistema

+

Documento de Arquitetura e Design

Projeto De aplicações Móveis

UFP

Autores: André Costa 35604
Bruno Pereira 38054

Data: 15 01 2022

Docente: Rui S. Moreira UFP

1. Introdução

Este documento descreve o projeto da cadeira de Programação de Aplicações Móveis que foi desenvolvido ao longo do semestre. Este projeto consiste na utilização de beacons(fazer a sua detecção e coletar as suas informações) para saber a localização aproximada de uma pessoa, através de tecnologias Bluetooth e podendo ainda sugerir o

caminho mais rápido para a localização pretendida tal como mostrar a localização atual do utilizador.

1.1. Objetivos

O objetivo desta aplicação é conseguir ter um sistema GPS autônomo indoor,que consiga saber com precisão a localização do seu utilizador bem como fazer o caminho mais curto entre o utilizador e o destino.

2. Descrição do sistema de Detecção de BLE e navegação indoor

Através da utilização de beacons BLE que estarão distribuídos em uma certa localização será possível detectá-los bem como mapeá-los em um grafo.Este contexto será útil, por exemplo, caso algum utilizador esteja a tentar descobrir o caminho mais conveniente(mais rápido) até um certo ponto.

Este projeto é feito recorrendo a uma Aplicação móvel Android que recebe os dados dos beacons através de tecnologias bluetooth, dados estes contendo o ID de cada beacon que será associado a uma localização. Assim, os sensores ao detectar o beacon nas proximidades, envia imediatamente os dados para o Dispositivo Móvel, permitindo saber a localização do utilizador e a hora a que passou nessa localização.

Tendo isto, os dados recebidos são encaminhados para o lado do servidor , onde serão recebidos e armazenados os dados.

Irão então ficar guardados os dados do beacons no servidor tal como os do utilizador onde será possível sincronizar os dados dos beacons entre todos os dispositivos

3. Requisitos do sistema de Deteção de BLE e navegação indoor

3.1. Requisitos Funcionais

Scan beacons

Sincronizar beacons com a API

Adicionar informações sobre um beacon

Adicionar Pontos no Grafo

Fazer ligação de pontos no grafo

Eliminar pontos do grafo

Pedir caminho mais curto entre 2 pontos

Estimar Localização do utilizador

3.2. Requisitos não Funcionais

Interface Responsiva

Interface Intuitiva

largura de banda minima 4 megabyte

escalavel

3.3. Requisitos de Sistema (Software e Hardware)

Mínimo de Ram 122,6 megabytes

Memoria minima 11,75 megabytes

Android

Min SDK 29

Target sdk 31

4. Arquitetura do sistema de Deteção de BLE e navegação indoor

Toda a arquitetura de alto nível do sistema segue um modelo genérico cliente-servidor que podemos ver na Figura 1. À esquerda teremos a própria aplicação móvel e à direita teremos os serviços do servidor.

Figura 1:~modelo genérico cliente-servidor arquitetura de alto nível do sistema de Deteção de BLE e navegação indoor

Em um diagrama mais detalhado da arquitetura podemos ver que se segue um design de Model View Control (MVC)…

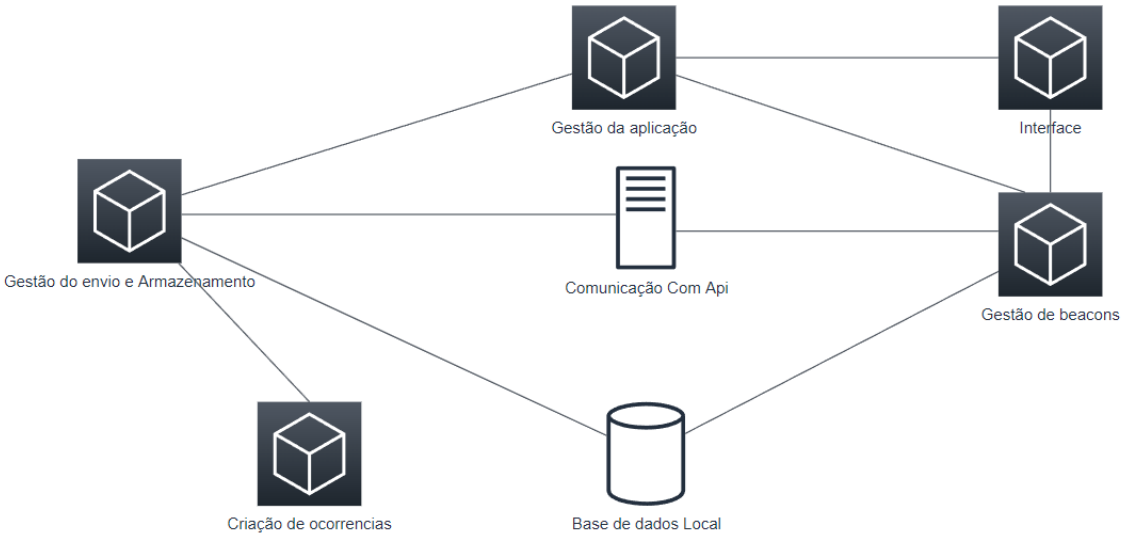


Figure 2: Diagrama mais detalhado arquitetura MVC do sistema de Deteção de BLE e navegação indoor

4.1. Arquitetura da Aplicação Móvel

A figura abaixo mostra uma visão mais detalhada dos componentes da aplicação móvel. Podemos identificar os seguintes componentes…

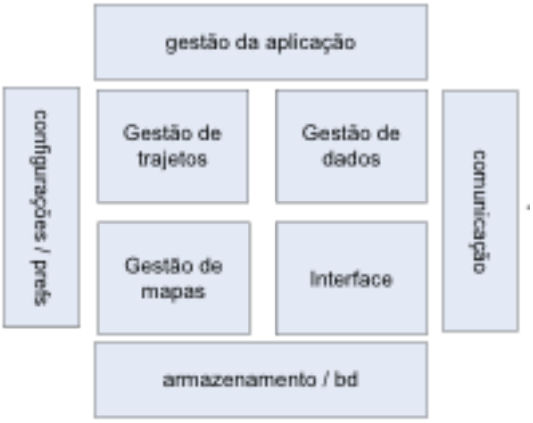
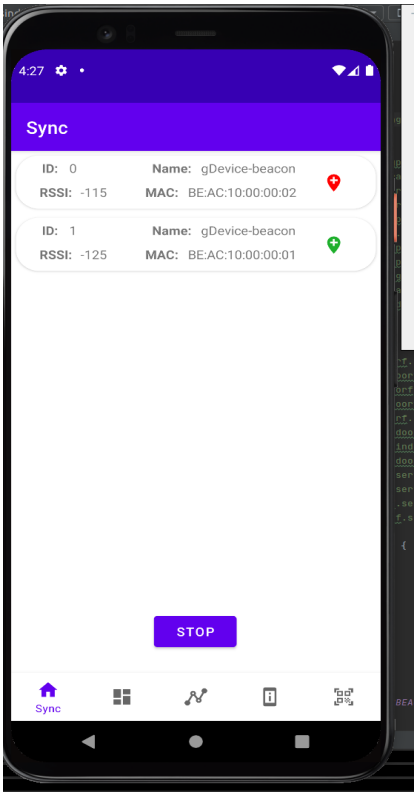
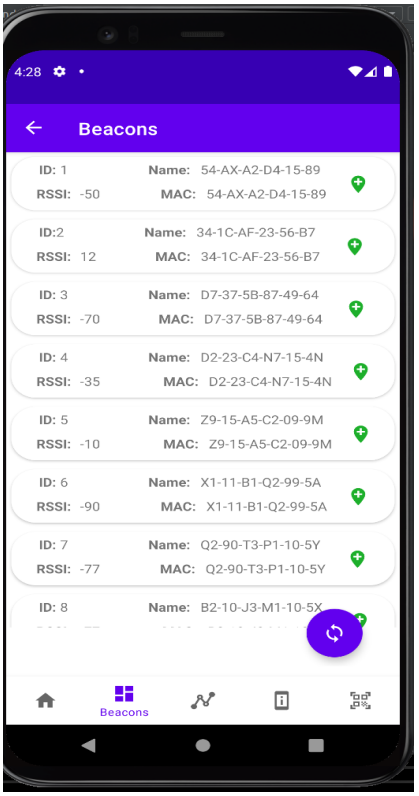
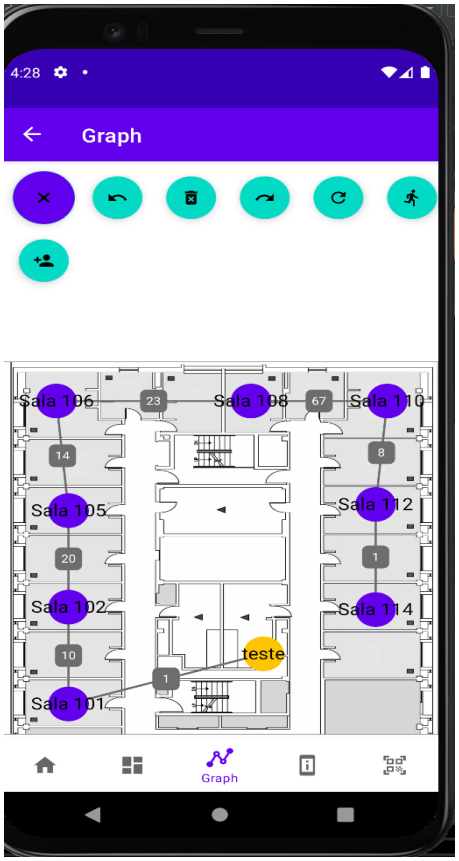


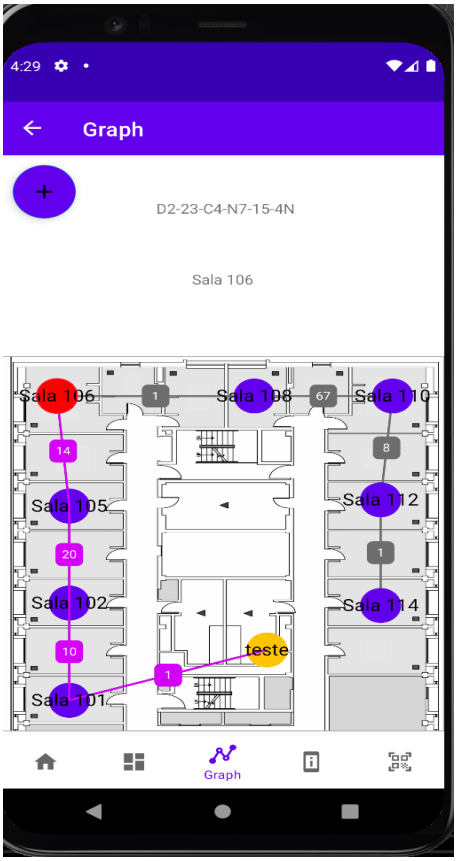
Figure 3: Arquitetura da aplicação móvel do sistema de Deteção de BLE e navegação indoor

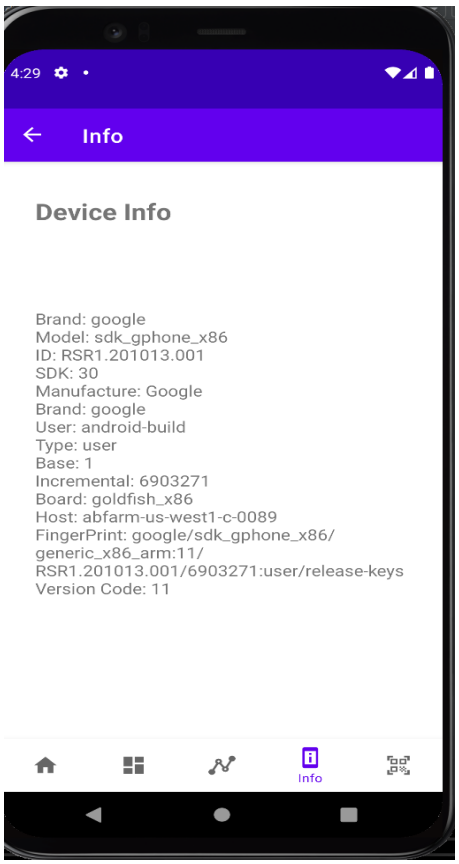
4.1.1. User Interface of the Mobile Application






4:28
Graph
Sala 106
23
Sala 108
67
Sala 110
14
8
Sala 105
Sala 112
20
1
Sala 102
Sala 114
10
teste
1
Sala 101
Graph


4:29
Graph
D2-23-C4-N7-15-4N
Sala 106
Sala 106
1
Sala 108
67
Sala 110
14
8
Sala 105
Sala 112
20
1
Sala 102
Sala 114
10
teste
1
Sala 101
Graph


4:29
Info
Device Info
Brand: google
Model: sdk_gphone_x86
ID: RSR1.201013.001
SDK: 30
Manufacture: Google
Brand: google
User: android-build
Type: user
Base: 1
Incremental: 6903271
Board: goldfish_x86
Host: abfarm-us-west1-c-0089
FingerPrint: google/sdk_gphone_x86/
generic_x86_arm:11/
RSR1.201013.001/6903271:user/release-keys
Version Code: 11
Info


4:29
fragment_share
QR CODE
Share

4.2. Arquitetura do Serviço web

Figure 4: Arquitetura web do sistema de Deteção de BLE e navegação indoor

5. Selection of Software Tools and Frameworks

Identify and document all the tools and components to be used.

Android Studio

Layout editor(XML)

APK Analyzer

Fast Emulator

Intelligent code editor

Flexible Build System

Realtime profilers

6. Adopted Guidelines and Standards

6.1. Coding and naming conventions

camelCase

7. Implementação do sistema de Detecção de BLE e navegação indoor

7.1.1. Navegação de interface do utilizador

É usada uma barra de navegação para o utilizador poder viajar entre fragmentos como podemos ver na imagem abaixo:

8. Teste do sistema de Detecção de BLE e navegação indoor

8.1. Testes da aplicação móvel

```
@RunWith(AndroidJUnit4ClassRunner::class)
class SyncFragmentTest{



    @Test
    fun buttonDisplay(){
        var frag= launchFragmentInContainer<SyncFragment>()
        Espresso.onView(ViewMatchers.withId(R.id.btScan))
            .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))
    }

    fun buttonDisplayText(){
        var frag= launchFragmentInContainer<SyncFragment>()
        Espresso.onView(ViewMatchers.withId(R.id.btScan))
            .check(ViewAssertions.matches(ViewMatchers.withText("Scan")))
    }
}
```

```
@RunWith(AndroidJUnit4ClassRunner::class)
class MainActivityTest{



    @Test
    fun layoutDisplay(){
        val activityScenario= ActivityScenario.launch(MainActivity::class.java)
        onView(withId(R.id.container)).check(matches(isDisplayed()))
    }

}
```

```
@RunWith(AndroidJUnit4ClassRunner::class)
class MainActivityTest{




    @Test
    fun layoutDisplay(){
        val activityScenario= ActivityScenario.launch(MainActivity::class.java)
        onView(withId(R.id.container)).check(matches(isDisplayed()))
    }

}
```

```
import ...

@RunWith(AndroidJUnit4ClassRunner::class)
class NavigationTest {

    @Test
    fun testFragmentNavigation() {
        val activityScenario= ActivityScenario.launch(MainActivity::class.java)
        //nav to another fragment
        onView(withId(R.id.navigation_graph)).perform(click())
        //verify
        onView(withId(R.id.btRun)).check(matches(isDisplayed()))


    }
}
```