

Filas de Espera

1st João Regateiro
DCC
FCUP
Porto, Portugal
up202004078@fc.up.pt

2nd João Miguel
DCC
FCUP
Porto, Portugal
up201108870@fc.up.pt

3rd Catarina Fonseca
DCC
FCUP
Porto, Portugal
up201405446@fc.up.pt

4th Tiago Campos
DCC
FCUP
Porto, Portugal
up202009657@fc.up.pt

Abstract—Sempre existiu um processo manual para o atendimento ao público quer seja via tickets, marcações, etc. Com isto em mente este trabalho tem como intuito disponibilizar uma fácil gestão no atendimento de tickets nos guichês. Tem-se 2 postos, um para o talho outro para a peixaria. No guichê vai haver um display para amostrar o número a ser atendido bem como o botão que vai incrementar-lo. Vai existir uma comunicação entre os 3 componentes principais Arduino, Beagleboard e Android. Esta comunicação vai desde Wi-fi até comunicação em porta serie.

Index Terms—Android, Beagleboard, Arduino, Firebase, Display, Wi-fi, API, Guichê

I. INTRODUÇÃO

Neste projeto tratamos de 2 secções com displays, base de dados e smartphones. Estas 2 secções têm colaboradores que primem um botão para incrementar o número corrente a ser atendido, para além disto vai estar disponível um display em cada secção para se visualizar o número.

Para a validação do número vai ser gerado um código numérico na aplicação do smartphone que depois no atendimento deve ser validado (incluindo o número de ordem).

O componente android vai ser usado pelos clientes para pedir a senha e vai servir como meio de comunicação entre o cliente e o nosso sistema usando o Wi-fi da rede local.

O componente Arduino vai servir para amsotrar o número atual a ser atendido bem como o identificador desse mesmo número e vai ter ainda a opção de incrementar a senha por um botão.

O componente Beagleboard vai receber e enviar o fluxo de informação entre os componentes.

Git: <https://github.com/joao2325/SE>

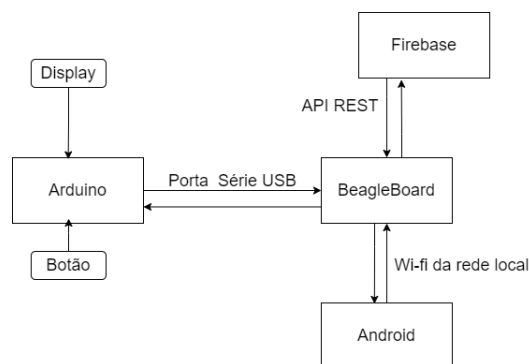


Fig. 1. Diagrama dos componentes

O objetivo final vai ser controlar de forma eficiente todo o atendimento dos guichês.

II. ESPECIFICAÇÃO

Como já foi referido existe 3 componentes principais físicos a serem usados o Android, Beagleboard e Arduino e um componente abstrato de comunicação que vai centrar-se na base de dados Firebase.

A. Android

A aplicação android vai ser usada pelos clientes para pedir a senha e vai servir como meio de comunicação entre o cliente e o nosso sistema usando o Wi-fi da rede local. O smartphone do cliente vai ter de ter essa aplicação android e vai poder optar que secção pretende escolher. Como resultado vai receber uma senha e o identificador dessa mesma senha. Para além disto a aplicação vai disponibilizar o tempo médio de espera a ser atendido.

B. Beagleboard

O Beagleboard é a componente central deste sistema pois vai receber, enviar e gerir toda a informação que existe.

O beagleboard vai comunicar via porta serie com o arduino, via API REST com a firebase e via wi-fi com o android na rede local. Ao receber o pedido de nova senha o beagleboard vai anotar o identificador na base de dados da firebase para depois ser apresentado na parte da verificação da senha. O beagleboard ao receber o pedido de incrementação da senha pelo arduino vai mandar um pedido à firebase e incrementar o número atual para o seguinte e devolver a resposta com o número atual ao arduino. Com isto manda outro pedido à firebase para eliminar o último identificador usado (proveniente da senha anterior).

Este componente vai ser o único a comunicar com a base de dados por motivos de eficiência e segurança.

C. Arduino

O arduino vai ter como periféricos o botão e o display. Vai ser ligado diretamente via porta serie ao beagleboard.

Quando o funcionário premir o botão, o arduino vai comunicar com o beagleboard e vai fazer output no display do número que foi incrementado bem como o identificador desse mesmo número.

A verificação do identificador da senha e do número é da responsabilidade do funcionario da secção.

D. Firebase

A base de dados usada não é relacional e é simples de estrutura.

Existe 2 tabelas que são as secções (Talho e Peixaria) e estas tabelas são ramificadas individualmente com Id, Senha e Data_atd.

- O Id é o identificador das senhas que existem no qual este vai conter todos os identificadores das senhas pedidas que ainda estão por ser atendidas.
- A Senha vai conter o número da senha atual a ser atendida.
- O Data_atd contém a data e hora na qual a senha atual foi pedida pelo funcionário.

E. Requisitos

1) Não-funcionais:

- Fácil de usar.
- Disponibilidade da Base de dados.
- Acesso no local.
- O sistema deve suportar 2 ou mais guichets simultaneamente.
- O sistema deve suportar 2 ou mais smartphones simultaneamente.
- A atualização de todos as aplicações com o número seguinte deve ser inferior a 3 segundos.
- Após pedido de senha na aplicação móvel, a emissão da mesma deve ser em menos de 2 segundos.

2) Funcionais:

- Apresentação do número atual a ser atendido e tempo estimado de espera na aplicação do smartphone.
- Emissão de um código numérico no smartphone para ser validado a posteriori no atendimento.

F. Material usado

- 2 smartphones xiami.
- 1 Beagleboard.
- 1 Arduino Uno.
- 1 Botão grande Arduino.
- 1 Display LCD 16x2 LED Backlight Azul.
- 1 Cabo USB para USB-B.

III. IMPLEMENTAÇÃO

A. Android

No componente Android começou-se por importar as bibliotecas necessárias e definir as variáveis que vão ser usadas. De seguida instância-se a app e criou-se as funções para os comportamentos que a app vai ter e as funções que vão trabalhar com os requests recebidos e enviados na rede local(Beagleboard). Um exemplo de função usada é `getCurrent()` que tem a finalidade de retornar o ticket a ser tratado no momento.

```
145 // Gets the current ticket being taken care of
146 fun getCurrent() {
147     val cache = DiskBasedCache(cacheDir, 1024 * 1024)
148     val network = BasicNetwork(HurlStack())
149     val requestQueue = RequestQueue(cache, network).apply {
150         start()
151     }
152     val url = "http://192.168.1."+Ip+":8080/actualNum?id=" + currId
153     val stringRequest = StringRequest(Request.Method.GET, url,
154         Response.Listener<String> { response ->
155             actualNum.text = response.toString()
156             if (currId == 1) {
157                 if (response.toInt().compareTo(ticketTalho.toInt() + 1) == 0)
158                     enableGetTicket()
159             }
160             else {
161                 if (response.toInt().compareTo(ticketPeixe.toInt() + 1) == 0)
162                     enableGetTicket()
163             }
164         },
165         Response.ErrorListener {
166         })
167     requestQueue.add(stringRequest)
168 }
```

Fig. 2. Função `getCurrent()`.

As imagens a seguir mostram a execução deste componente.



Fig. 3. Menu inicial.



Fig. 4. Layout para tirar senha.



Fig. 6. Layout 2 para tirar senha.



Fig. 5. Layout com a Senha pedida.

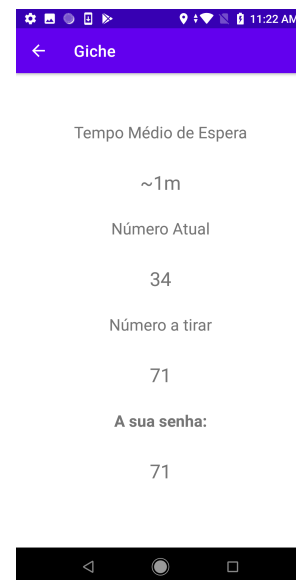


Fig. 7. Layout 2 com a Senha pedida.

B. Arduino

No componente arduino usou-se o Arduino IDE para executar as configurações. Inicialmente começou-se por definir as bibliotecas, de seguida quais os digital pins a serem usados para ligar o arduino ao periférico display e por último as variáveis com tipo inteiro nominadas de inputPin, currentState e num como amostra a figura 8.

```
#include <MD_KeySwitch.h>
#include <LiquidCrystal.h>

// LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int inputPin = 9;

int currentState;
int num = 0;
```

Fig. 8. Estrutura inicial do código arduino antes da função setup().

```
28 void loop() {
29     currentState = digitalRead(8);
30
31
32     if(currentState == HIGH){
33         digitalWrite(inputPin,LOW);
34     }
35     else
36     {
37         digitalWrite(inputPin,HIGH);
38         num++;
39         lcd.setCursor(0,1);
40         String s = String(num);
41         lcd.print(s);
42     }
43     Serial.println(currentState);
44     delay(1000);
45 }
46
```

Fig. 10. Função loop().

Após esta configuração inicial passou-se à função Setup(). Instância-se a função Serial.begin() que passa o valor 9600 para o parâmetro de velocidade. Isto diz ao Arduino que vai-se trocar mensagens com o Monitor a uma taxa de dados de 9600 bits por segundo. Recebe Input de dados pelo Pin digital número 8 e definimos o tamanho que o display tem, que é 16 por 2 usando lcd.begin(16, 2). Por fim manda-se escrever "Fila de espera:" na primeira linha do display e o número que está a ser atendido na 2ª linha do display.

```
12 void setup() {
13
14     Serial.begin(9600);
15
16     pinMode(8, INPUT);
17
18     pinMode(inputPin, OUTPUT);
19     //digitalWrite(inputPin,HIGH);
20     lcd.begin(16, 2);
21     lcd.setCursor(0,0);
22     lcd.write("Fila de espera:");
23     lcd.setCursor(0,1);
24     String s = String(num);
25     lcd.print(s);
26 }
```

Fig. 9. Função setup().

Finalmente definiu-se a função loop() dizendo que quando o botão for premido, o inputPin recebe essa informação ficando como "HIGH" e incrementa o número da variável "num", fazendo OUTPUT para o visor do display. Isto sempre com um delay 1 em 1 segundo.

Na imagem a seguir vemos o exemplo da execução deste componente.

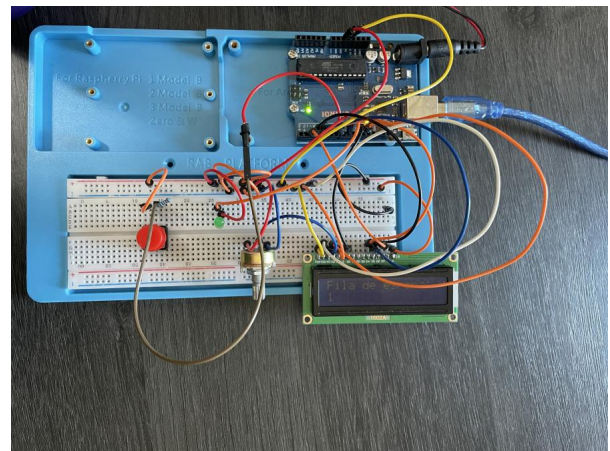


Fig. 11. Arduino e Display imagem 1.

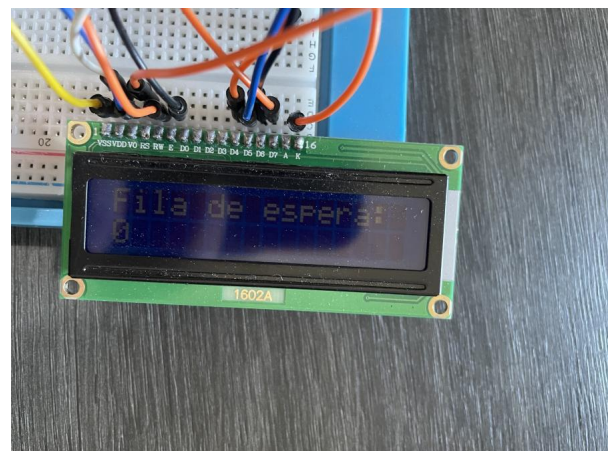


Fig. 12. Display com a senha nº0.

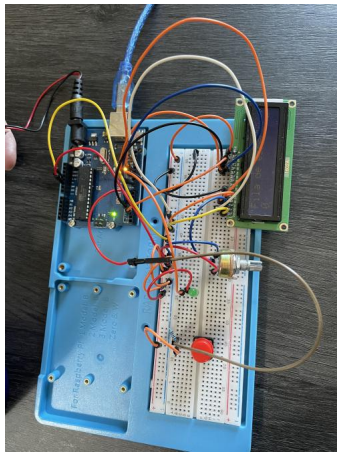


Fig. 13. Arduino e Display imagem 2.

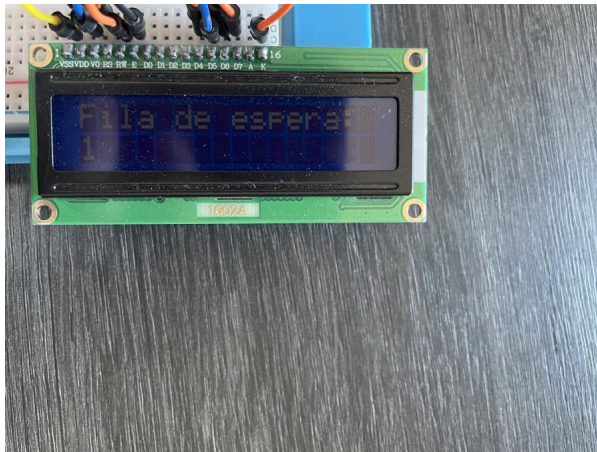


Fig. 14. Display com a senha nº1.

C. Firebase

Na implementação do Firebase criou-se o projeto no website respetivo.

Definimos na secção de Autenticação como método o "Email/senha" e criou-se um utilizador com email e password. Na Base de dados estruturou-se 2 tabelas, Talho e Peixaria e ramificou-se cada uma com Data_atd que tem o propósito de gravar a data e hora no momento que a senha atual é incrementada, Id que contém todos os identificadores das senhas que ainda estão em espera e a senha que está a ser atendida como amostra a Figura a seguir, definiu-se as regras para autenticação de escrita e leitura só para quem está autenticado.

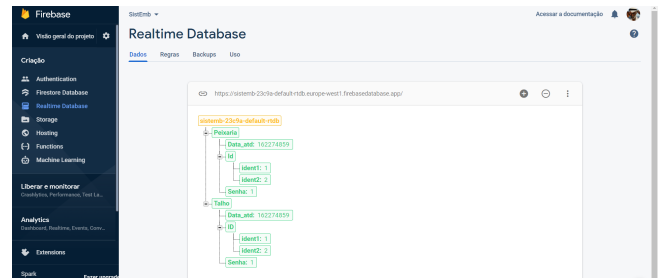


Fig. 15. Estrutura da Base de Dados.

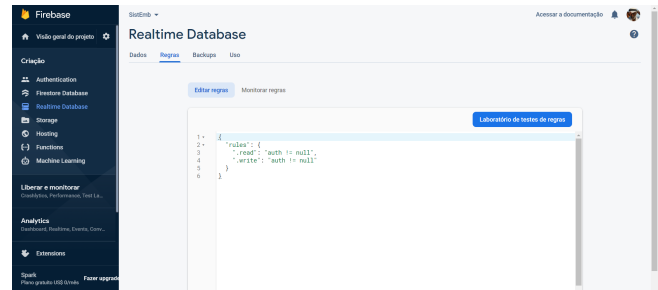


Fig. 16. Regras para a autenticação.

Para se testar a ligação via API REST usou-se código python e ficheiro json com a informação da chave privada do administrador(adquirido pelo website da firebase).

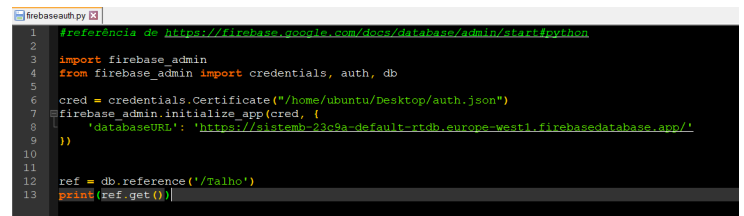


Fig. 17. Código python para autenticação.

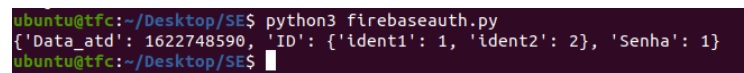


Fig. 18. Resultado do pedido de autenticação com a amostra do conteúdo da tabela Talho.

D. Beagleboard

IV. CONCLUSÃO

Conclui-se que mesmo este trabalho não estar 100% operacional maior parte dos componentes estão a funcionar da forma esperada. Sendo que este projeto está dividido em componentes conseguimos testar cada um individualmente. Houve dificuldade para adotar um método de autenticação no firebase que a posteriori conseguiu-se ultrapassar, houve também dificuldades no arduino para conseguir ter um método que estabeleça uma comunicação entre este e o beagleboard.