

SMARKT - Smart Supermarket

Programming for IOT applications(01QWRBH) – Final Project



Daniel Gaiki (s232546)



SMARTK – General System



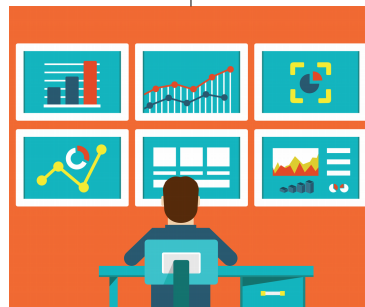
Electronic Trolley:

Raspberry Pi + LCD interface +
weight sensor + barcode Scanner

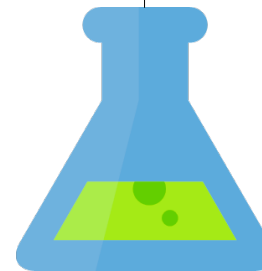


Payment and Checkout control:

Raspberry Pi + LCD interface +
Keyboard + barcode Scanner



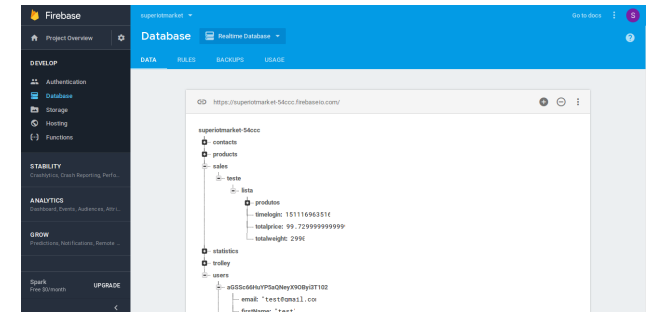
Dashboard- Freeboard



Prediction of future stock or
consumption



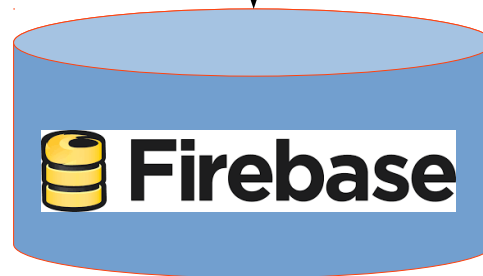
SMARTK – FIREBASE



Control.py

- Time of scanning
- Scanned Prod.
- Total weight
- Time weight meas.
- Trolley #

HTTP post/get



HTTP post/get

payment_check.py

- Payment verification.
- Time of payment
- User
- Mode of payment

HTTP post/get

Display HTML + AngularJS

- Login
- Register new user
- View shop list
- Confirm shop list

dashboard prediction.py

statistics about purchases
generate a image

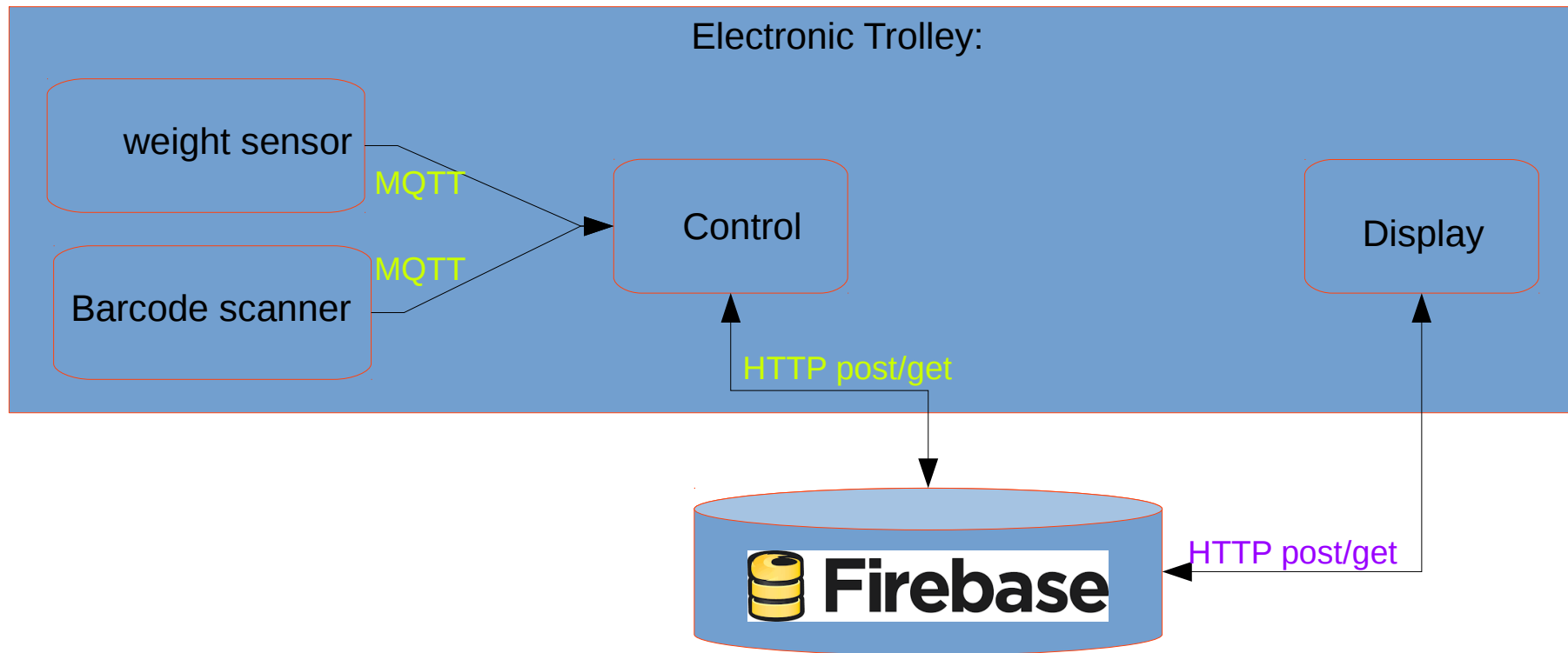
HTTP post/get

calc_listas.py

- Total Price
- Total weight
- Create historic info user- Statistics



SMART – Electronic Trolley





SMARTK – FreeBoard



HTTP post/get

Dashboard FREEBOARD

- Statistics about quantity of purchased products
- Graphic generation
- Prediction of future consumption



SMARTK – Payment and Checkout



People passage control

- Open close
- Light indicator of free passage
- Alarm

MQTT

Payment and checkout

- Payment OK
- Send message to passage control
- Scan trolley
- Send mode of payment
- Send time of payment

HTTP post/get





SMARTK – Trolley shop interface

1- REGISTER NEW USER

Register ✕

First Name:

Last Name:

Email:

Password:

Confirm Password:

Close Register

HTTP/POST

Firestore

Identifier	Providers	Created	Signed In	User UID ↑
teste2@email.com	📧	Nov 20, 2017	Nov 20, 2017	1GkptqWoAR0RtIKJTxCi5sU6L...
teste3@email.com	📧	Nov 20, 2017	Nov 20, 2017	4TISwXtB9uOwZoiIyxaRBilP8wi1
teste4@email.com	📧	Nov 20, 2017	Nov 20, 2017	6t6GnKU80kVrOm5gXwnv0LB4...
teste9@email.com	📧	Nov 20, 2017	Nov 20, 2017	GFzZlF5oVMNzTmZSpu9NfH4d...
teste5@email.com	📧	Nov 20, 2017	Nov 20, 2017	KMRrz2X0v4aqIcJgxcyuH51tm...
teste1@email.com	📧	Nov 20, 2017	Nov 20, 2017	KZxF0uOwUcWktil6OemZonWo...
teste10@email.com	📧	Nov 20, 2017	Nov 20, 2017	Ss679NRAnuWC4PN5fCmjxmrzKH2
teste6@email.com	📧	Nov 20, 2017	Nov 20, 2017	UfgPEPWChEUtu781n2pK5Nrlq...
teste7@email.com	📧	Nov 20, 2017	Nov 20, 2017	cQGVVeewXKdhFZ26sOTTaA72ENJ2
teste8@email.com	📧	Nov 20, 2017	Nov 20, 2017	iwDmh8JLnWhpEJFet6A68Mv9ZBP2

```
users
├── 1GkptqWoAR0RtIKJTxCi5sU6Low2
├── 4TISwXtB9uOwZoiIyxaRBilP8wi1
├── 6t6GnKU80kVrOm5gXwnv0LB4FFT2
├── GFzZlF5oVMNzTmZSpu9NfH4dAOP2
├── KMRrz2X0v4aqIcJgxcyuH51tmMQ2
├── KZxF0uOwUcWktil6OemZonWoQBq2
├── Ss679NRAnuWC4PN5fCmjxmrzKH2
├── UfgPEPWChEUtu781n2pK5Nrlqxa2
├── cQGVVeewXKdhFZ26sOTTaA72ENJ2
└── iwDmh8JLnWhpEJFet6A68Mv9ZBP2
    ├── email: "teste8@email.com"
    ├── firstName: "teste8"
    └── lastName: "teste"
```



SMARTK – Trolley shop interface

2- LOGIN

Login ×

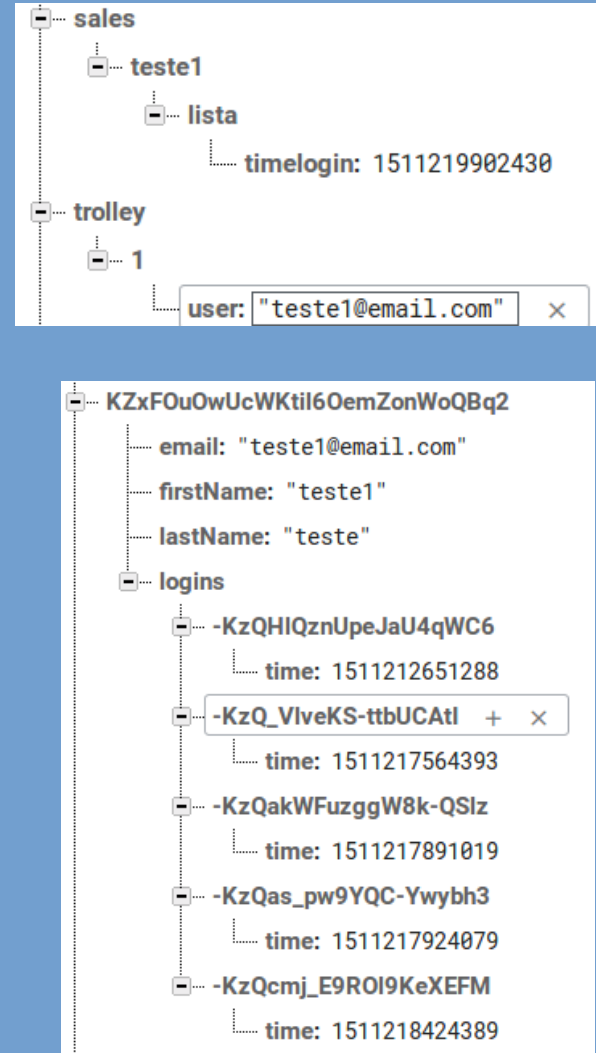
Email:

Password:

Close Login

HTTP/POST

Firestore



SMARTK – Scanner Barcode

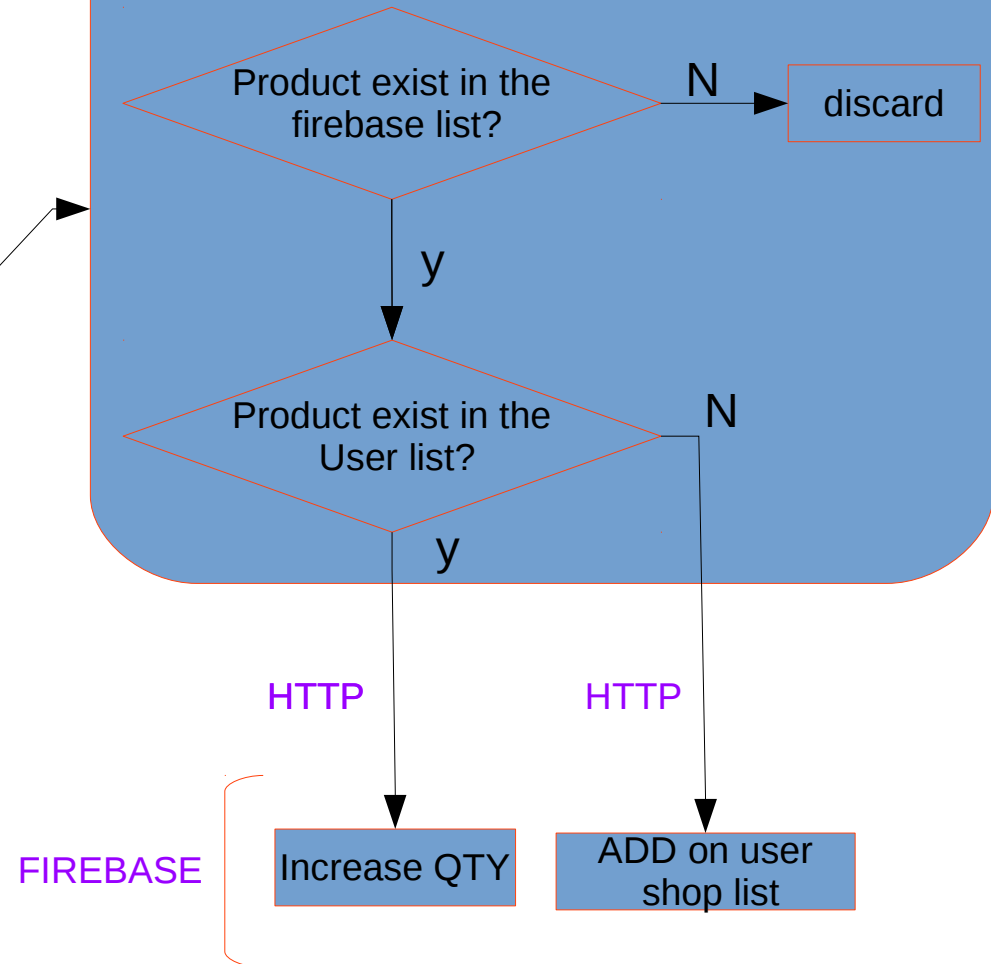
3- Scan Barcode

```
import cv2.cv as cv #Use OpenCV-2.4.3
import zbar
```



MQTT

Control.py





SMARTK – Final Shop accept list

3- Verify final shop list

```
sales
├── teste1
│   └── lista
│       └── produtos
│           ├── 1511254993
│           │   ├── prod: "Baking cake"
│           │   ├── qty: 2
│           │   └── total_price: 22.28
│           └── 1511255029
│               ├── prod: "Raw legums"
│               ├── qty: 2
│               └── total_price: 34.22
└── time: "2017-11-21 10:03:08.569439"
    timelgin: 1511254974257
    totalweight: "1200"
```

Shoplist confirmation

SMARTK - Smart Supermarket

Your Shop List

Products

Product	QTY	Price
Vegan food	3	88.98
Raw legums	3	51.33

[FINISH and ACCEPT](#)

Generation of timeconf on firebase

```
sales
├── teste1
│   └── lista
│       └── produtos
│           ├── 1511254993
│           │   ├── prod: "Baking cake"
│           │   ├── qty: 2
│           │   └── total_price: 22.28
│           └── 1511255029
│               ├── prod: "Raw legums"
│               ├── qty: 2
│               └── total_price: 34.22
└── time: "2017-11-21 10:03:08.569439"
    timeconf: 1511255109968
    timelgin: 1511254974257
    totalweight: "1200"
```



SMARTK – Final Calculation on shop list

4- Final shop list calculation on FIREBASE

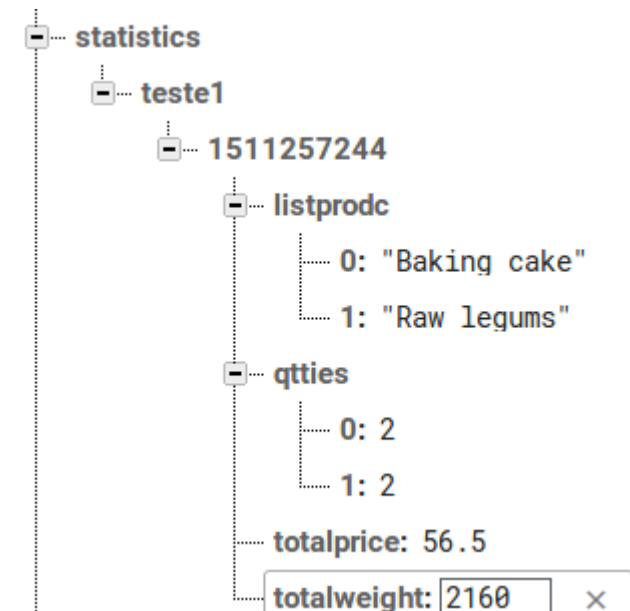
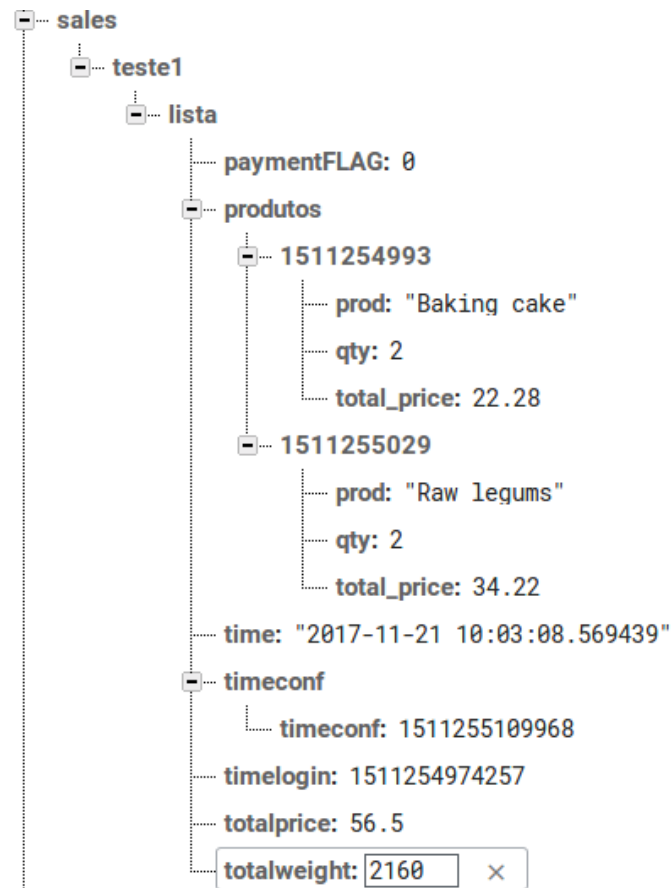
Generation of:

Total Price

Total Weight

Payment FLAG

Statistics





SMART – Entire system codes

Interface

Index.html

<u>views</u>	<u>js</u>	<u>controllers</u>
<u>add.html</u>	<u>app.js</u>	<u>addcontroller.js</u>
<u>edit.html</u>	<u>script.js</u>	<u>editcontroller.js</u>
<u>list.html</u>	<u>security.js</u>	<u>listcontroller.js</u>

Python

Client

Scanner.py
Control.py
weight.py

Firebase management

calc_listas.py

Supermarket payment control

payment_check.py

dashboard

prediction.py



SMARTK – Entire system codes

Interface explanation

Index.html General Vision of the screen, Register and Login screen

add.html Vision of purchased items by the user.

addcontroller.js Javascript connection between Firebase and add.html code, responsible to get purchased items on the user list on firebase.

Python explanation

Client

Scanner.py Using OpenCV library, get image from webcam and remove from a barcode image the code, send the message by MQTT to control.py
Message Topic: j_temp = '{"id":"trolley#","sensor":
"scanner","reading":""+str(codigo)+", "time":""+hora_s+"}'

weight.py Measure the weight of total amount of purchased products of the client, send message to control.py

Message Topic: j_temp = '{"id":"trolley#","sensor":
"weight","reading":""+str(ins)+", "time":""+hora_s+"}'

Control.py Receive message from Scanner and weight sensor and send to Firebase database, control the login of user on the trolley, remember, the trolley has a fixed number.



SMARTK – Entire system codes

Python explanation

Supermarket
payment control

payment_check.py

Scan the trolley, verify if the weight measured by trolley sensor has the same value of the database value, if the user pay the purchase , send a message to control passage authorizing the passage to the user.

Firestore
management

calc_listas.py

Control all information in the firestore, calc the total amount of each user, and the total weight of each purchase.

dashboard

prediction.py

Verify the statistics about purchases of the products on the history, quantity of each product buyied in specific week day, play a regression and predict the future purchase, generate a image and sent to the firestore.

SMARKT - Smart Supermarket



We are hiring too!

:)

Thanks for watching!

