

CCT College Dublin

## ARC (Academic Research Collection)

---

ICT

---

Summer 5-2019

### Ecological Expenses Tracker – EcoExT

Asmer Bracho

*CCT College Dublin*

Carina Lins

*CCT College Dublin*

Eduardo Firino

*CCT College Dublin*

Gabriel Oliveira

*CCT College Dublin*

Miguelantonio Guerra

*CCT College Dublin*

Follow this and additional works at: <https://arc.cct.ie/ict>



Part of the [Computer Sciences Commons](#)

---

#### Recommended Citation

Bracho, Asmer; Lins, Carina; Firino, Eduardo; Oliveira, Gabriel; and Guerra, Miguelantonio, "Ecological Expenses Tracker – EcoExT" (2019). *ICT*. 6.

<https://arc.cct.ie/ict/6>

This Undergraduate Project is brought to you for free and open access by ARC (Academic Research Collection). It has been accepted for inclusion in ICT by an authorized administrator of ARC (Academic Research Collection). For more information, please contact [debora@cct.ie](mailto:debora@cct.ie).

## **Ecological Expenses Tracker – EcoExT**



Asmer Bracho - 2016328  
Carina Lins - 2016308  
Eduardo Firino - 2016274  
Gabriel Oliveira - 2016310  
Miguelantonio Guerra - 2016324

Dublin, May 2019

*Dedicated to all the people that we love and miss, this is for all of you!*

# Abstract

*To consume: verb*

1. *Eat, drink, or ingest (food or drink); 2. Buy (goods or services); 3. (of a feeling) completely fill the mind of (someone).*

**Dictionary, C. (2018). CONSUME | meaning in the Cambridge English Dictionary.**

*“People are constantly consuming.”* Nigel Barber a Ph.D. psychologist and author talks about cosmism in *The Human Beast* and why people consume so much nowadays.

Either drinking, eating or buying goods in general on an everyday basis, people tend to consume more than they expect. When it comes to the aspect of buying, every single person in the world has a need to do so. It could be a simple transaction such as having a yummy ice-cream on a summer day, either a large purchase such as buying your first and desired car!

The point is: every single day billions of people around the world are constantly consuming and this number has been increasing continuously.

When talking about consuming and making purchases, it is always expected that there will also be a transaction in this act of trading and in this transaction a paper receipt is included, where the user gets a description of what they have spent and how much was spent.

The current dissertation demonstrates an application solution to reduce the cost of paper receipts that have been used in a daily basis stage and to track the value of expenses and incomes of a user who is consuming and spending their money when purchasing goods.

Additionally, we implement the Ecological Expense Tracker (EcoExT), using Raspberry Pi (RPi) educational computer in order to display the QR Codes that will be generated on the server and the application will be able to scan the code and gather the receipt.

Finally, the prototype created will show the results of the success work and future recommendations in order to improve the current version of the system.

# Table of Contents

<b>Abstract.....</b>	<b>3</b>
<b>Table of figures .....</b>	<b>8</b>
<b>Chapter 1 - Introduction .....</b>	<b>12</b>
1.1 Receipts .....	12
1.1.1 Type of receipts.....	14
1.1.2 Cost of paper receipts .....	16
1.1.3 Usage and Reuse .....	18
1.2 Printers .....	18
1.2.1 Type of printers.....	19
1.2.2 Cost of printers.....	19
1.3 Environmental impact .....	20
1.3.1 Bisphenol A .....	20
1.3.2 Trees and the environment .....	21
1.4 Solution proposed .....	22
1.4.1 The project idea.....	23
1.4.2 Overview .....	23
1.4.2 Steps of adopting the system .....	26
1.4.4 The benefits of adopting EcoExt.....	26
1.5 EcoExt is born .....	29
<b>Chapter 2 - System Analysis and Design .....</b>	<b>31</b>
2.1 Functional Requirements .....	31
2.1.1 Purpose .....	31
2.1.2 Overall description .....	31
2.1.3 Detailed description of the functionality of the proposed system .....	32
2.1.4 Checklist for subsequent stages .....	33
2.2 System design diagrams .....	34
2.2.1 Use Cases.....	34
2.2.2 Use Cases Specifications.....	37
2.2.3 User Stories.....	42

2.2.4 Activity Diagrams .....	43
2.2.5 Overview Diagram .....	57
2.2.6 Sequence Diagram .....	58
2.3 Logical design of the raspberry pi application .....	59
2.3.1 Backend Design .....	59
2.3.2 Frontend Design .....	61
2.4 User interface design .....	62
2.4.1 Wireframes .....	63
2.5 Data design .....	75
2.5.1 Database Requirements .....	75
2.5.2 Data Repository .....	77
2.5.3 Storage of Data .....	77
2.5.4 Entity relationship diagram .....	77
2.5.5 Relational Model .....	80
2.6 Database Normalization .....	84
2.6.1 First Normal Form (1NF) .....	85
2.6.2 Second Normal Form (2NF) .....	85
2.6.3 Third Normal Form (3NF) .....	85
2.6.4 Fourth Normal Form (4NF) .....	86
2.7 Data dictionary for the relations .....	86
2.7 Database physical design .....	89
2.8 Database final conclusions and Database schema .....	93
Chapter 3 - Implementation of the System .....	95
3.1 Technologies used .....	95
3.1.1 Code Hosting/Versioning .....	95
3.1.2 Database Implementation .....	96
3.1.3 Development Environment .....	97
3.1.4 File Storage .....	98
3.1.5 Hardware .....	98
3.1.6 Programming .....	100

3.1.7 Task Manager .....	101
3.1.8 Visual Design.....	101
3.1.9 Quick Response Code (QR code) .....	102
3.2 Developing Raspberry Pi Application .....	102
3.2.1 Listener side of the Raspberry Pi application .....	102
3.2.2 GUI side of the Raspberry Pi application .....	105
3.2.3 Controlling both sides of the Raspberry Pi application.....	106
3.2.4 Running the Raspberry Pi application.....	108
3.3 Problems encountered .....	109
3.3.2 Impeditive factors of android application development .....	110
3.3.3 Problems encountered on database implementation .....	111
3.3.4 Problems encountered when developing the Raspberry Pi application .....	113
Chapter 4 - Testing and Evaluation .....	115
4.1 Test plans and objective .....	115
4.2 Unit and exploratory testing .....	115
4.3 Scenarios to be tested .....	115
4.3.1 Scenario 1: .....	115
4.3.2 Scenario 2: .....	116
4.3.3 Scenario 3: .....	117
4.3.4 Scenario 4: .....	119
4.3.5 Scenario 4: .....	120
4.3.6 Scenario 6: .....	121
.....	122
4.3.7 Scenario 7: .....	122
4.3.8 Scenario 8: .....	123
4.3.9 Scenario 9: .....	124
4.3.10 Scenario 10: .....	125
4.3.11 Scenario 11: .....	127
4.4 Scenarios for Raspberry Pi testing.....	128
4.4.1 Scenario 1: .....	128

4.4.2 Scenario 2: .....	129
4.4.3 Scenario 3: .....	130
4.4.4 Scenario 4: .....	130
4.4.5 Scenario 5: .....	130
4.5 Automated testing .....	131
4.5.1 Writing the testing class .....	132
.....	132
4.5.2 Running Our Tests .....	132
4.5.3 Setting up the testing environment.....	133
4.5.4 Testing Methods.....	134
4.5 Automated testing conclusions .....	138
<b>Chapter 5: Conclusion</b> .....	140
5.1 Project success and results .....	140
5.2 Suggestions for further work.....	140
5.2.1 Cloud storage .....	140
5.2.2 Continuous Integration .....	140
5.3 Features planned to be added in the future.....	141
<b>6. Bibliography</b> .....	142
<b>Appendix A – Android Class Diagram</b> .....	147
<b>Appendix B - Project Planning</b> .....	148
1. Identification and needs in the market .....	148
2. Definition of objectives and priorities .....	149
3. Defining deliverables .....	149
4. Teamwork .....	152
4.1 The Team Members .....	152
5. Methodology .....	152
<b>Appendix C – Self Reports – Group Journal</b> .....	155



## Table of figures

Figure 1: Result of survey question three .....	12
Figure 2: Result of survey question six. ....	13
Figure 3: Thermal receipt paper. ....	15
Figure 4: Result of survey question five. ....	16
Figure 5: Box of receipt rolls. ....	17
Figure 6: Graph of Average Annual Printing.....	19
Figure 7: Result of survey question seven. ....	21
Figure 8: Reasons to go paperless. ....	22
Figure 9: Full Working Prototype Perspective. ....	24
Figure 10: Full Working Prototype Perspective(changed). ....	24
Figure 11: Result of survey question 8. ....	27
Figure 12: Name chosen to the project. ....	29
Figure 13: Project manager chosen to the team. ....	30
Figure 14: EcoExT logo.....	30
Figure 15: EcoExT Application Use Case Diagram.....	35
Figure 16: EcoExT Raspberry Pi Use Case Diagram.....	36
Figure 17: User Stories Number 1.....	42
Figure 18: User Stories Number 2.....	42
Figure 19: User Stories Number 3.....	43
Figure 20: User Stories Number 4.....	43
Figure 21: Login Activity Diagram. ....	44
Figure 22: Login Activity Diagram. ....	45
Figure 23: Registration Activity Diagram. ....	46
Figure 24: Purse Activity Diagram. ....	47
Figure 25: Transaction Activity Diagram. ....	48
Figure 26: Categories Activity Diagram.....	49
Figure 27: Budget Activity Diagram. ....	50
Figure 28: View Statistics Activity Diagram. ....	51

Figure 29: Setup Profile Activity Diagram. ....	52
Figure 30: Check Notifications Activity Diagram. ....	53
Figure 31: Scan QR Code Activity Diagram. ....	54
Figure 32: Logout Activity Diagram. ....	55
Figure 33: Raspberry Pi - Create Transaction Activity Diagram. ....	56
Figure 34: Overview Diagram of EcoExT. ....	57
Figure 35: EcoExT System Sequence Diagram. ....	59
<b>Figure 36: EcoExT System Sequence Diagram. ....</b>	<b>60</b>
Figure 37: Raspberry Pi Application home window. ....	61
Figure 38: Pi Application QR Code window. ....	61
Figure 39: Raspberry Pi Application successfully scanned window. ....	62
Figure 40: Raspberry Pi Application not scanned window. ....	62
Figure 41: Raspberry Pi Application frontend pseudo code. ....	62
Figure 42: Login wireframe. ....	63
Figure 43: Registration wireframe. ....	64
Figure 44: Sidebar wireframe. ....	65
Figure 45: Records wireframes. ....	66
Figure 46: Transfer funds wireframes. ....	67
Figure 47: Homepage wireframe. ....	68
Figure 48: Notifications wireframe. ....	69
Figure 49: Creating a new purse wireframe. ....	70
Figure 50: Labels wireframes. ....	71
Figure 51: Categories Wireframes. ....	72
Figure 52: Edit Profile wireframe. ....	73
Figure 53: QR code and Receipt Wireframe. ....	74
Figure 54: Entity Relation Diagram using CHEN Notation. ....	78
Figure 55: First version of Entity Relation Diagram. ....	79
Figure 56: Second version of Entity Relation Diagram. ....	79
Figure 57: Mapped Regular Entity types. ....	80
Figure 58: Mapped Weak Entity Types. ....	81

Figure 59: Mapped Binary 1:N Relationship Type. ....	81
Figure 60: Mapped Binary M:N Relationship Types. ....	82
Figure 61: Mapped Multivalued Attributes. ....	83
Figure 62: Relationship Between a Transaction and an Establishment.....	83
Figure 63: Result of the Mapping of Entities, Relationships and Multivalued Attributes. ....	83
Figure 64: Result of the Mapping of Entities, Relationships and Multivalued Attributes. ....	84
Figure 65: Database Final Schema Diagram. ....	94
Figure 66: Screenshot of the GitHub online repositories. ....	96
Figure 67: Google Drive in use as a tool to store files. ....	98
Figure 68: Hardware – Raspberry Pi.....	99
Figure 69: Prototype of EcoExt QR Code .....	102
Figure 70: Pi Class UML Representation. ....	103
Figure 71: Class Diagram of the Listening Side of the Application .....	104
Figure 72: Class Diagram of the GUI Side of the Application. ....	106
Figure 73: Class Diagram of the GUI Side of the Application.....	106
Figure 74: Class Diagram of the Pi Application. ....	107
Figure 75: Running the Pi Application.....	109
Figure 76: @SuppressWarnings (“Valid Fragment”) annotation .....	111
Figure 77: Screenshot of registering with Google account.....	116
Figure 78: Screenshot of logging in to EcoExT application.....	117
Figure 79: Screenshot of creating and adding purse into EcoExT application.....	118
Figure 80: Screenshot of deleting purse on EcoExT application. ....	119
Figure 81: Screenshot of purse details on EcoExT application .....	120
Figure 82: Screenshots of adding a transaction, such as income or expense on EcoExT application. .....	121
Figure 83: Screenshot of transaction added on EcoExT application.....	122
Figure 84: Screenshots of filtering transactions by the date on EcoExT application.....	123
Figure 85: Screenshots of deleting the transaction on EcoExT application .....	124
Figure 86: Screenshots of expenses’ statistics graph on EcoExT application .....	125
Figure 87: Screenshots of Scan QR Code and receipt scanned on EcoExT application. ....	126
Figure 88: Screenshot of the Receipt after the QR Code is scanned.....	127

Figure 89: Screenshot of the application logout .....	128
Figure 90: EcoExT QR Code being displayed on the Raspberry Pi. ....	129
Figure 91: EcoExT error screen being displayed on the Raspberry Pi.....	130
Figure 92: EcoExT success screen being displayed on the Raspberry Pi.....	130
Figure 93: EcoExT home screen being displayed on the Raspberry Pi.....	131
Figure 94: Importing the unittest Module into the testing file. ....	132
Figure 95: Class Heading of the Testing Class .....	132
Figure 96: Main Program of the Testing file .....	132
Figure 97: File Structure of the parts that are used to perform the tests.....	133
Figure 98: GraphQL Mutation showing the format of the transaction data to be able to store it in the database successfully. ....	134
Figure 99: Test 1 - Assert Token Length.....	135
Figure 100: Result of the Test 1 - Assert Token Length .....	135
Figure 101: Test 2 - Assert TypeError Exception .....	136
Figure 102: Result of the Test 2 - Assert TypeError Exception. ....	136
Figure 103: Test 3 - Assert Connection Established. ....	137
Figure 104: Result of the Test 3 - Assert Connection Established.....	137
Figure 105: Result of the Tests Assert Token Length, Assert TypeError Exception, and Assert Connection Established. ....	138
Figure 106: Test 3 - Assert Connection Established. ....	138
Figure 107: Android Class Diagram.....	147
Figure 108: Ethical approval from CCT College to conduct research .....	148
Figure 109: EcoExt project timeline.....	149
Figure 110: EcoExt documentation timeline. ....	150
Figure 111: EcoExt project timeline updated. ....	150
Figure 112: EcoExt documentation timeline updated. ....	151
Figure 113: Team members sprint. From left to right, Asmer, Miguelantonio, Carina, Gabriel and Eduardo.....	152
Figure 114: Trello in use as a tool to manage tasks. ....	153

# Chapter 1 - Introduction

## 1.1 Receipts

The receipts can be defined as a document record which, there is a transfer of ownership over a good or some commercial activity between people and companies. But in general, how important are those receipts?

They are intended to record the transfer of a monetary value between the parties and can be used also in broader contexts and yet may be used to cancel the validity of another already existing receipt.

With the aim of proving that the transaction is really valid, the receipts appeared around the 18th century and until the present days they are evidence that a purchase has been made. It is worth noting that, for each transaction carried out, at least one paper receipt will be issued proving, therefore, that the transaction occurred.

Although, what happens then when we receive this paper receipts?

In general, most people from a user perspective tend to have the habit of keeping track of what was spent and for that reason, they choose to keep the invoice in order to be in control of what was spent on that specific purchase. Some also keep the invoice and yet end up losing it, facing the future with the dilemma of not being able to check what was bought because they have lost the receipts, there is no physical record and there is no possibility of checking the purchases performed. Especially in the case of when the purchase is made with cash and there is no electronic record at all.

Ethical approval was granted and the consent of students at CCT College was obtained so that a survey could be conducted, the aim was to analyse the cost, impact and care of environment on how likely people would keep their paper receipts. The results clearly show that most people who participated in the survey would like to keep their receipts.

See graph below:

### Q3. How likely are you on keeping your receipts?

71 responses

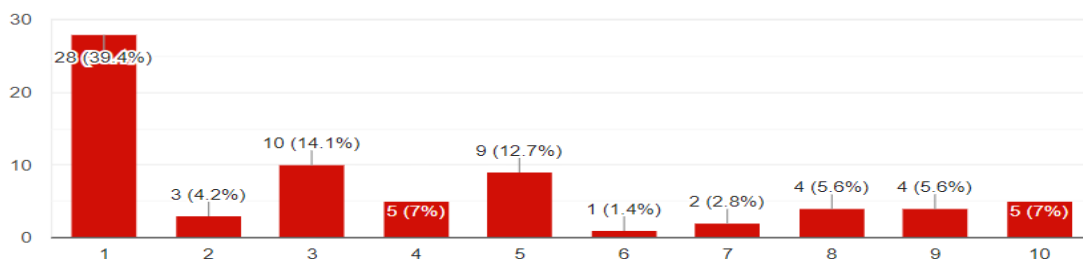


Figure 1: Result of survey question three

On a scale from 1 to 10, 1 meaning very likely and 10 meaning less likely, the graph above shows that the 39.4% of the people who have been interviewed would most likely to keep their receipts.

Another percentage of people choose not to receive that paper receipts and are not too worried about having control of what was spent on that purchase or the day as a whole.

### Q6. If you wouldn't, why?

14 responses

N/A
I think it's important to have the paper receipt
No patient to read
Too much clutter
I dont keep any receipts
Although I like the idea of going peperless, specially when it comes to receipts I probably won't hold on to for very long. I'm not keen on providing my contact information widely.
I forget about them in my emails
It's tedeous to keep receipts on paper
In case i need to give it to my accountant
If I'd get it on my phone it would annoying. But It if it were on screen like ATM I wouldn't mind
Safe time
Not tech savvy

Figure 2: Result of survey question six.

However, there are some laws that require some individual or legal entities to have strict control over the custody of such documents, since there is a requirement of the government of a period that is necessary for them to collect taxes which have not been taken beforehand, also known as the debt collection period. To prove compliance, it will be necessary for the taxpayer to have possession of the document.

In any case, it is clear that there is a proof managerial evidence in the same way of great physical importance to keep the paper receipts. Some companies even rent larger spaces, or even look for a bigger, physical space where they are located, in order to keep them.

In reality, what happens in most cases and which we think may be useful in the future, keeping a piece of paper which would cover us of any obligation or may serve as proof or a guarantee of some purchase that have been made and which requires a proof of purchase. This is in

fact not a mistake since it is really necessary in some cases to present the paper receipts for the purpose of declaring and confirming such an act of purchase. In this way, we can consider that by imposing our mind that we should keep the paper receipts, whenever we make a purchase, we will try to keep some of them with us. Such an act may be considered by some as an excuse to store so many papers that are almost never necessary.

We can highlight that the electronic receipts came up with the intention of changing this reality. Nowadays there is a big environmental impact where many paper receipts are being unnecessary printed and for this reason, digital receipts, unlike paper receipts, have come to reduce paper usage.

There are many reasons that the digital receipts need to be introduced in our lives, the most important reason is that technology is part of our lives these days and we have to adapt ourselves to the new future. Another reason is that considering the business aspect of it, there is also a Business Intelligence perspective where strategies and technologies are used by enterprises with the aim of data analysis and business information. Through the usage of these digital receipts, the companies are able to track the customers' purchases becoming easily to support Business Intelligence in order to adapt themselves to the marketing plan and campaigns towards customers.

### 1.1.1 Type of receipts

Although there are several types of invoices depending on the country and its laws and cultural origins, we will focus on two main types of invoices that are used in Ireland.

**Paper Receipts** - The paper receipts are used to prove that the purchase was made as previously mentioned and thus this enforces the importance of checking if there was any kind of error when the transaction was performed, such as a double charge. Another necessary use for paper receipts is when there is an audit since it facilitates the inspection for documentation of expenses and taxes. When buying a gift there are also gift receipts that can be printed to be delivered to the person who will be receiving the present itself. One more utility would be to confirm that the transaction occurred at a certain time, date and place and for a certain amount.

There are two types of paper receipts, the most common is the Thermal one, although plain paper is still used, thermal paper existed in order to be a thermal paper and it first became available in the early 1990s. Basically, a receipt printed on thermal paper when heated will colour the paper. Bringing the option of printing logos and a more detailed and trend version for new designs.

According to Wikipedia (accessed on 07/11/2018): “*Thermal paper is a special fine paper that is coated with a material formulated to change colour when exposed to heat. It is used in thermal printers and particularly in inexpensive or lightweight devices such as adding machines, cash registers, and credit card terminals.*”

The thermal paper is made up containing four different types of chemicals being leuco dyes, developers, sensitizers and stabilizers. Usually, they are dyes used in papermaking, which are sensitive and often unstable, such dyes return to their crystalline forms when they are not stored in hot and humid conditions.

Another type of paper receipt is the white plain paper receipt, which is generally sold as plain till rolls and does not have the characteristic of changing colour. In this case, it is necessary for the printer to have a cartridge to be able to transfer the ink to this type of paper, so it will depend on the type of printer being used to print the receipts.



Figure 3: Thermal receipt paper.

Credit: [https://en.wikipedia.org/wiki/Thermal\\_paper](https://en.wikipedia.org/wiki/Thermal_paper)

**Electronic Receipts** - According to Collins dictionary, an electronic receipt is: *“noun: (Retail: Payment technology) an electronic receipt is one created in a computerized cash register, or by an online retailer. It will usually show the date and time, how payment is made, and other details of the sale.”*

When dealing with e-receipt, usually customers are provided, at the point of collection of their email addresses, an “opt-out” option from receiving marketing material. Retailers, in fact, should have an electronic record whether a customer has agreed or not to receive marketing emails.

According to the new General Data Protection Regulation (GDPR) rules (accessed on 19/10/2018): *“The Data Protection Commission (DPC) carried out a series of audits in order to assess how organisations gather and process personal data in the course of providing electronic receipts (e-receipts) to customers. In a number of cases, e-mail addresses gathered for the purpose of issuing e-receipts were subsequently used by retailers in order to issue marketing material. Following on from these audits, the DPC has produced guidance around the use of e-receipts to assist retailers to adhere to best practice in this regard.”*

When analysing digital receipts, certain rules must be followed according to the new data protection legislation.

Electronic receipts notwithstanding, are still not so common every day, there are many that still are not very practical because the consumer will have to provide their email in order to receive



them. Many consumers do not opt for this option because of many factors to consider, in most cases because they have very busy lives and also because they do not want many emails in their inbox.

Therefore, some establishments have the option to send the receipts by email and also to print the invoice. In most cases, the establishments prefer to keep the consumer's emails so they can send marketing messages and also to allow the tracking of trending products, buying history and even customer movements.

Certainly, electronic receipts can be advertised as an approach of paper saving. They are also convenient and easy to find and if you want to return an item, make a warranty claim or need a receipt for tax or business purposes.

Below is the result of a survey conducted on how many people are willing to use e-receipts and the results show that over fifty (50) per cent of people are looking forward to it:

### Q5. Would you rather have an e-receipt instead of paper receipts?

71 responses

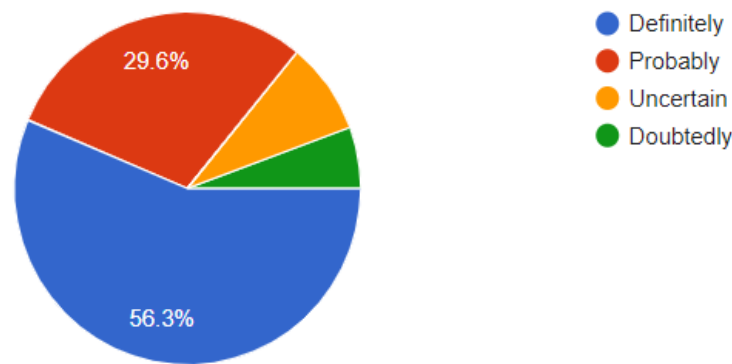


Figure 4: Result of survey question five.

#### 1.1.2 Cost of paper receipts

There are several costs related to the printing of receipts, being they physical cost, labour cost, environmental cost and also the cost with logistic systems and remote connection providers.

When someone buys a receipt printer, the original purchase price is only a small fraction of what they signed up for in the long run. To begin with, there is the cost of the roll of paper, as already mentioned, it could be the thermal paper or the white plain paper, respectively the most recommended and also the cheaper when it comes to cost. When bought in bulk to large quantities

the cost can be reduced and it may not strike the companies as a major expense, although if you consider that the machine will have a lifetime use, how much does paper cost over it?

Investigation and research were done where for each roll of paper would be purchased from some of the suppliers here in Ireland:

- On this website [<https://www.shop4rolls.ie/>] approximately twenty euros (€20) for a box of thermal receipt paper and sixteen euros (€16) for white plain receipt paper.
- On another website [<https://tillroll.needa.ie/>] depending on the size of the rolls, it can vary from eight euros (€8) to sixteen euros (€16) for each roll.
- And also [[https://www.nisbets.ie/tableware-and-bar-supplies/cash-handling/till-rolls-and-ink-rollers/\\_/a33-3](https://www.nisbets.ie/tableware-and-bar-supplies/cash-handling/till-rolls-and-ink-rollers/_/a33-3)] on this website a box would vary from eight euros (€8) to twenty euros (€20).



Figure 5: Box of receipt rolls.

Credit: <https://www.amazon.com/Thermal-Terminal-Receipt-diameter-CORELESS/dp/B076GHYX11>

Assuming that the common and average size per receipt would be nine (9) inches. On a regular roll, you can get approximately two hundred (200) receipts. That means over two million item rated life of a typical receipt printer. Now let's consider that through ten thousand (10,000) rolls of paper, at an approximate cost of four thousand (4,000)! The result of that is certainly many times higher than the cost of the printer itself. Not to mention that every beginning and end of paper rolls are wasted in most cases never used to the end. It seems to be a very little cost per receipt; However, it may not forget about the total cost that floats over an average of two (2) cents per receipt.

Regarding the fact that regular plain white paper is slightly cheaper than thermal paper, yet most bankers and retailers opt for the thermal option because in that they can print their respective

logos and also in the background can have a custom note. In this case, the value of a plain receipt paper can double, and then most opt for the thermal receipt paper option.

We can also mention the cost of inks, every once in a while, the ribbon or inkjet cartridge needs to be replaced, though far less often than the paper. With a cost around three (3) euros, this may vary depending on the establishment. On average, over the lifetime of the device, you'll replace a ribbon two hundred (200) times at a cost of six hundred euros (€600), or an ink cartridge between forty (40) and one hundred (100), at six hundred (€600) to two thousand (€2,000) euros expense.

### **1.1.3 Usage and Reuse**

Accordingly, to a survey study done with some business and I.T students, they tend to not keep their paper receipts and would rather have a way of having all their expenses electronically tracked as shown on the results of the survey on [figure 4](#).

People might think that all sort of paper receipts are useful and have a certain relevance and importance, that they are compulsory by law, when in fact that is not most of the cases as in Ireland there is no legal obligation under consumer protection laws for a business to provide a receipt for the goods that one buys.

Probably and often people do not recall when it was the last time that they have actually made a good use of a paper receipt. Or the last time they have updated their financial spreadsheet with data from it.

The thermal paper receipts, as mentioned before are covered with some substances including the BPA (Bisphenol A), a chemical compound which has been linked to various side effects when on a high level in the bloodstream. By having BPA on it, any kind of paper instantly disqualifies from recycling, impacting then on the reuse and recycling not being able to be done. Approximately ninety-three (93) percent of the disposed of paper receipts contains BPA on it.

Companies have claimed a “*non-BPA*” paper, but it is typically replaced by Bisphenol S (BPS), which is a similar chemical that researches indicate it has similarly detrimental effects as Bisphenol A (BPA) has.

Alternatively, what can be done to reuse receipts would instantly be the digital receipts and card readers and smartphones that could offer paperless transactions with a certain security. Not impacting the environment and helping the consumer to check and store their invoices in an easy and accessible way.

## **1.2 Printers**

We may also consider it necessary that establishments nowadays make use of paper receipts printers. These printers are usually connected to the main system of the establishment and also have costs. The printer is a tool of total necessity and perhaps could not be dispensed.

Although companies are adapting to new technologies, implementing e-receipts, the use of printers has not been ruled out, and there is still a huge cost for each computer that interconnects the system with the receipts to be printed.

### 1.2.1 Type of printers

Nowadays, there are three main printers that allow you to print paper receipts, they are dot matrix, ink printer and thermal printer. Even though the dot matrix is not commonly used anymore, those three are the main ones in the last decade.

One of the earliest known printers, which became quite popular for paper receipt printing, emerged around the 1920s and the printing method became known as a dot matrix printing which was a computer printing process from a collection of dot matrix data.

The dot matrix printer uses ribbons, they have to be replaced frequently and it has to cost which will be added up, in the end, making it more expensive to the companies. This printer has not been used so often anymore and almost no places have it.

Another type of printer would be the inkjet which is also faster and its cartridges last longer than the dot matrix one, although they have a higher cost if you compare to ribbons.

The most used printer today and the thermal printer, which are faster and have a better cost benefit from using thermal paper already mentioned above. This type of printer does not require ink.

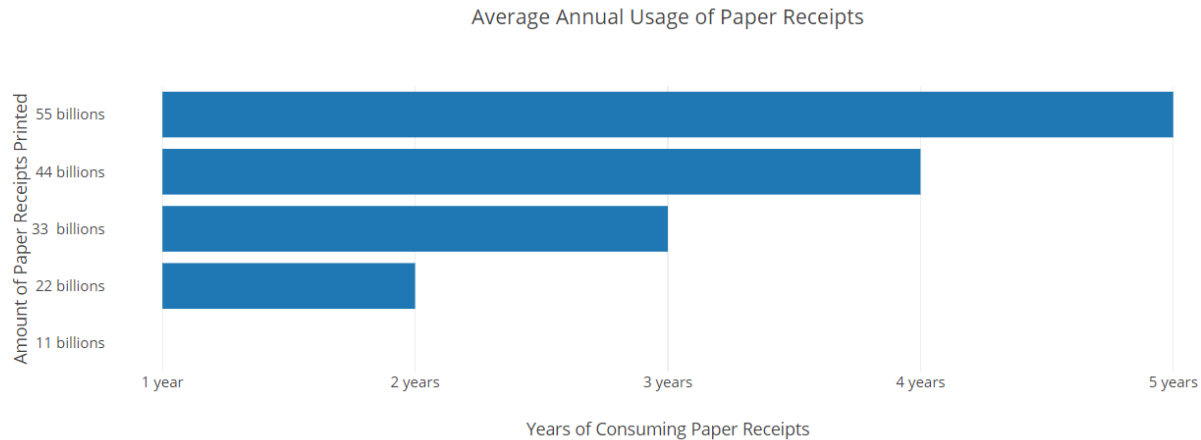
### 1.2.2 Cost of printers

Although each printer has a varying cost and can be sold together or separately, the cost can vary to higher or lower values depending on each customer's needs.

In a market cost research, we can note that printers range from one hundred and fifty (€150) to three hundred (€300) euros.

- On this website: [<https://www.cashdrawers.ie/-Retail-Printer-Barcode-Scanner-Cash-Drawer-Bundle>] the whole cash drawer bundle cost approximately three hundred and seventy five (€375) and it comes all together.
- The most popular one is the Epson Printer which costs around two hundred (€200) depending on where you are purchasing them. [<https://www.cashdrawers.ie/-Epson-TM-T20-USB-Receipt-Printer>]

Printers may vary depending on the port inputs and what type of printing will be performed, which may lead to higher or lower costs. Below I present to you a graph of overhead associated with paper and printers over a period of one (1) to five (5) years:



### 1.3 Environmental impact

The paper receipt is an unsustainable paper that nowadays is causing a huge environmental impact. With paper receipts being unnecessary printed every day and no one seems to really keep them, they become ubiquitous and unnecessary.

Researches show that every year, in order to create receipts, over two hundred and fifty (250) million gallons of oil, ten (10) million trees and one (1) billion gallons of water are consumed. All of that generates one and half (1.5) billion pounds of waste, and this is only in the United States (US). Not mentioning the other continents and that multiplied by the years along. In the United Kingdom (UK), every year more than eleven (11) billion receipts are printed. Could you imagine how much waste does the world in general generates? - All of this clearly creates an extraordinary environment impact.

#### 1.3.1 Bisphenol A

Bisphenol A is an industrial chemical that has been used to make plastics since the early 1960s. It can be found in polycarbonate plastics and epoxy resins. Such chemicals are often used to manufacture containers for food and liquids such as bottles of water.

Bisphenol A (BPA) can make its way into food and liquids, therefore reaching the human blood when those are consumed. Bisphenol A (BPA) can be found nearly everywhere, especially in a big environmental issue: **Paper Receipts**.

Studies and researches have linked Bisphenol A (BPA) with the following problems:

- Cancer
- Infertility and Fetal Development
- Brain Function

- Heart Disease and Diabetes
- Weight Problems

Estimates show that ninety-three (93) percent of paper receipts are coated with Bisphenol-A. Researchers have found that retail workers are at greater risk after a study conducted with a group pregnant cashier. They showed higher levels of Bisphenol A (BPA) than Industrial workers.

### 1.3.2 Trees and the environment

Every year in the United States only, ten (10) million trees are cut down and twenty-one (21) billion gallons of water are used to make paper receipts, generating six hundred and eighty-six (686) million pounds of waste and twelve (12) billion pounds of Carbon Dioxide (CO<sub>2</sub>). On average, two point eight (2.8) pounds of Carbon Dioxide (CO<sub>2</sub>) is produced per receipt issued.

According to some research done, fifty-five (55) thousand trees could save an entire tree. Depending on how long the receipt is measured if they are longer than six (6) inches or duplicated that numbers could change for twice as much. Even so, if that changed, it would save fifty-eight point eight (58.8) million gallons of water and keep seventeen point six (17.6) million pounds of Carbon Dioxide (CO<sub>2</sub>) from entering the atmosphere.

Most people have never sought to know or actually read about the effect that this has on our lives, but they are convinced that there is a certain impact on the environment. We conducted a survey with some students and the result and over eighty-five (85) percent of people agree on the changes that have to be made. See the figure below:

#### Q7. Do you think there is an environmental benefit by going paper-free?

71 responses

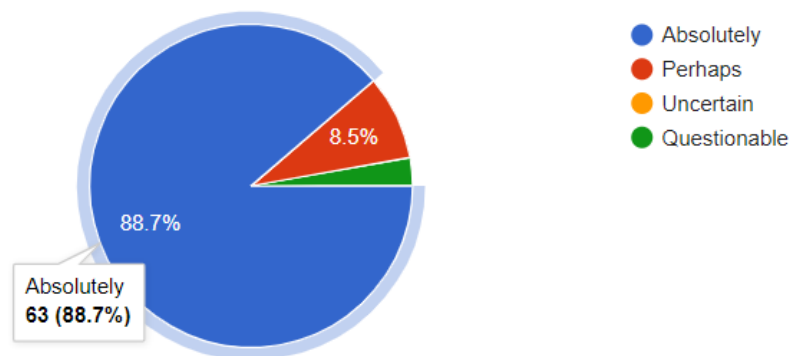


Figure 7: Result of survey question seven.

If we take in consideration and analyse, each human being is responsible for the environment in which we live and because of that, everyone could become aware that there is really a benefit by going paper-free, such benefit can be reflected in the personal and climatic aspects.

### 1.4 Solution proposed

A new software is proposed as an innovation change for going receipt paper free. Adopting a new technology often feel daunting when it comes to running a business. The process could be slow and implementation could take a certain time to be adapted correctly by employees and consumers.

For the past decade, the technology is changing our lives and it also helps business to rely on it adapting themselves to the new challenges and reducing their paper trail needs, accelerating the process as a result.

There are many reasons for going paperless, from consumers to business, that is a need that has been implemented by innovation technologies. It is not only about saving the environment, based on a study people have spent most of their lifetime in a digital environment such as computers, tablets, and smartphones. They crave efficiency and opt to spend their money on customized and good experiences. The last thing any customer wants is another paper receipt cluttering their wallet.

Owing to the entire aspects mentioned above, such as waste of paper receipts, costs and environmental impact, we have taken all aspects into consideration and we are proposing a sustainable solution using our technological know-how to do so.

Below we point out three main reasons to go paperless and the benefit of it:

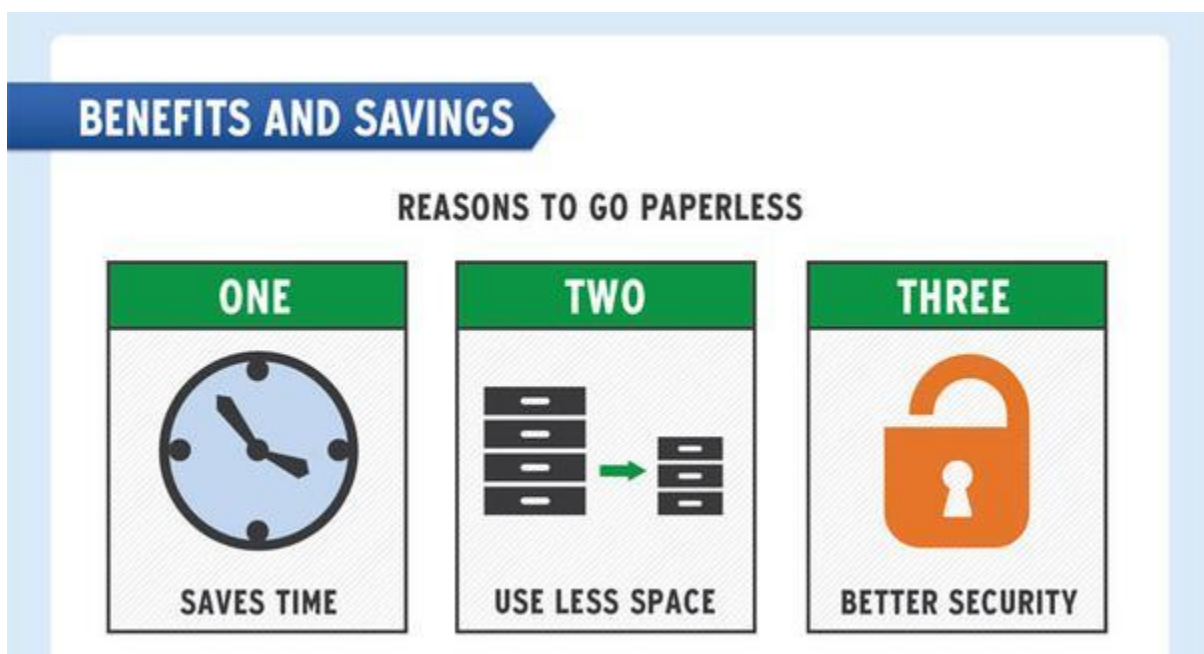


Figure 8: Reasons to go paperless.

### 1.4.1 The project idea

The aim is to eradicate paper receipts starting with small business and grow wider to large business. EcoExt reduces the need of paper receipts on the business by providing them with a solution that allows users to instead scan a Quick Response Code (QR code) and automatically get the receipt into their EcoExt financial application, users will be able to access their record anytime and business will be able to have a digital record from every single purchase made with the application.

EcoExt's strategy is financial tracking and prevention of unnecessary paper receipts being printed - generating a cost and impact - encouraging consumers and retailers to adopt the application as a lower and stable cost solution.

According to The Huffington Post and the Clinton Global Initiative in recognition of the latter's CGI University meeting (April 5-7 at Washington University in St. Louis): *"The benefits of cutting down on receipts extend beyond the environment and our wallets. Deforestation has been linked to ethnic, religious and political instability across the world. Thermal receipt paper also contains a harmful chemical, Bisphenol-A, which has been linked to various forms of cancer and instantly disqualifies receipt paper from recycling."*

EcoExt software, process the data from the issuer and generates a digital receipt for the transaction which will be acquired from the customer by scanning the code through the application. This solution has an objective to avoid either cashier or consumer the need to handle a paper receipt.

The finance application also has features such as observing what has been bought and analyse the personal expenses across different categories like groceries, home expenses, medical expenses and so on. The absolute idea can be explained into more details on the following general overview of how the process happens.

### 1.4.2 Overview

EcoExt App project intends to change the perspective for usability of paper as an unnecessary resource when generating paper base receipts, every time a purchase is generated. However, the process of an application fully working will require implementation from the business side and possible changes in the business model.

When taking into consideration an application that scans a Quick Response Code (QR code) which contains information of a specific transaction, it implies a set of rules and steps that must be followed in order to succeed.

The figure below shows a general understanding of a fully working system prototype and the activity's flow of it.



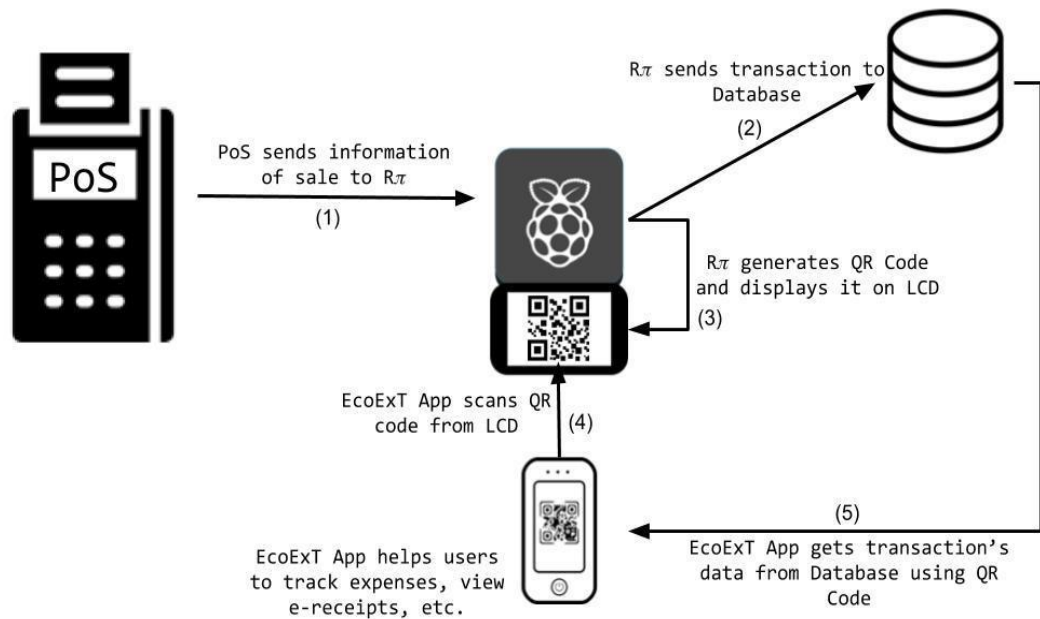


Figure 9: Full Working Prototype Perspective.

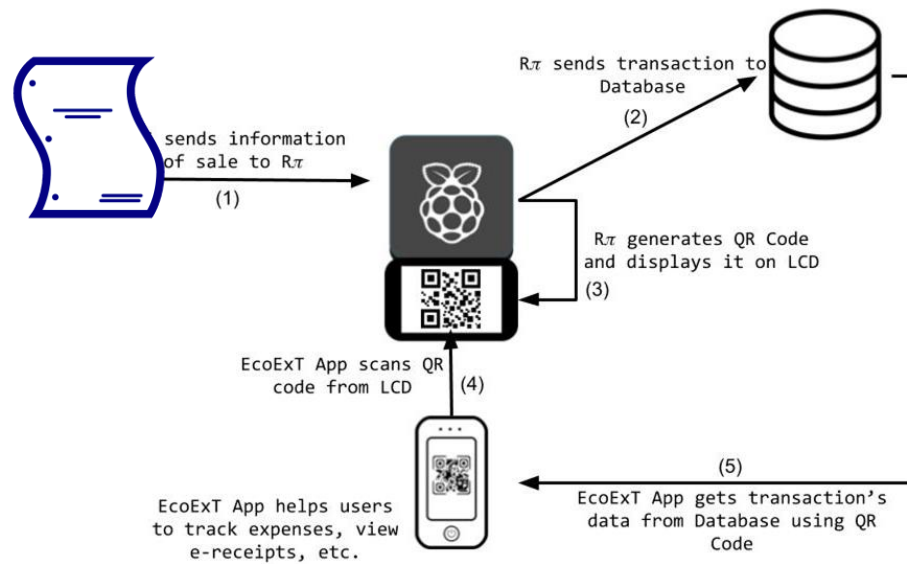
By analysing this prototype, four (4) individual systems that are all connected can be clearly identified as follow.

1. EcoExT Point of Sale (POS)
2. EcoExT Database
3. EcoExT Quick Response Code Generator (Raspberry Pie)
4. EcoExT Application

The proposed recommended solution in order to develop a full working system may be achieved by having a breaking down into two different approaches, which will be called **Versions**.

#### ***1.4.2.1 First version***

The first version consists in dropping the EcoExt Point of Sale (POS) System as a dependency that could get the information from and, instead, replace it for a simple serialize file that will contain the transaction information generated for a third-party Point of Sale (POS) system. See Figure below:



Having this file as a starting point, will facilitate and simplify the developing process as well as opening an opportunity to integrate third-party Point of Sale (POS) system already prevailing. With the aim to adapt the whole process making the whole adaptation and model's switching an easier process.

In order to clarify better, we can imagine that we want to implement the EcoExT System completely (as initially mentioned on the above overview), then the proposal will be for single establishments to change their current Point of Sale (POS) system and paper receipt system for the EcoExT one. Let's pretend we succeed and get the establishments to implement it, then the same process will have to be followed for every single establishment that is willing to implement the system proposed.

Instead of applying the previous approach, now we can suppose that the proposal goes to a specific Point of Sale (POS) company, and consist of a system that can be integrated to their actual Point of Sale (POS) system by simply making it generates the serialize file where EcoExT will start working from. That would be a more consistent approach and it also would be beneficial to both sides, the company that is adapting as well as EcoExT system.

#### ***1.4.2.2 Second version***

The second version consists of completely developing the below four (4) subsystems:

- EcoExT Point of Sale (POS)
- EcoExT Database
- EcoExT Quick Response Code Generator (Raspberry Pie)
- EcoExT Application

Even though this version implies a much longer development process as well as the integration of a business plan in regard of POS system able to match different markets and types of

establishments, it is also a solution and it needs to be included in case eventualities come across during the implementation of the first version.

Finally we come to a conclusion that any of the versions, first or second, will bring to the market a fully working system that pretends to change the known business model for a much accessible and convenient receipt storage system for customers, it is worth noting that the cost-effectiveness as in investment for establishments and, in fact, an eco-friendly system with environmental benefits.

#### 1.4.2 Steps of adopting the system

Step 1. Download the application

Step 2. Registration/Log In

Step 3. Start keeping track of your finances

Step 4. Access the camera to scan Quick Response code (QR code)

Step 5. Scan Quick Response code (QR code) & Get your receipts

#### 1.4.4 The benefits of adopting EcoExt

The benefits could be divided into two categories such as benefits for the companies and benefits for the users of the system. Stating the benefits for **the company** at first:

##### 1.4.4.1 Increased Efficiency

According to a study done by ILM Corporation, searching through paper can take up as much as thirty (30) percent of an employee's work time.

*“Going paperless allows you to [be more efficient], and efficiency equals money,” says Matt Peterson, president and CEO of eFileCabinet. “Responding to customers quickly is your edge as a small business, and being efficient and paperless gives [you] that edge.”*

In fact, EcoExt proposes a solution which can increase the efficiency for many consumers and they can simply go to the application and search for their invoices. Looking through stacks of paper can be a waste of time and we have all had that experience. Ensuring that the majority of receipts are digital makes retrieving them with less effort as many users have their phone at all times. Another common usage is a cloud-based storage which many smartphones are configured automatically where those documents could be easily approached and benefits to be accessed anywhere and on any device.

##### 1.4.4.2 Money Saved

As discussed before, there are several costs related to printing paper receipts such as costs with till rolls, printers and so on. Adopting EcoExt will be beneficial in a long-term cost-saving and will reduce expenses with unnecessary printing, it might seem that purchasing software will increase the cost, although in a long-term perspective, the companies that adopt the system will no longer need to purchase receipt rolls, reams of paper, printers, inks or containers for file storage.

Another benefit that can occur is that by going paperless and adopting EcoExt technology, the companies can improve their profitability by over fifteen (15) per cent, they will also be able to have a multi-channel & digital platform that interacts with customers in order to deliver a better approach for a efficient customer delivery and then being more profitable.

#### ***1.4.4.3 Advertisement***

The electronic receipts that EcoExt users will generate when purchasing something into their application will also be able to offer targeted ads with what the customer is spending and companies may have that advantage by opting to pay an extra cost and thus adapting the receipts with more ads and so, gathering more customers.

Considering the general benefits for the users additionally:

#### ***1.4.4.4 Financial Tracking***

EcoExt comes with the approach of tracking personal finances where some people usually struggle to do, the app will help users that adapt the system to have a better insight into their spending habits.

We have conducted a survey with some students at CCT College and the results are quite curious, over forty (40) percent of people keep track of their finances. See figure below:

**Q8. Do you keep track of your expenses?**

71 responses

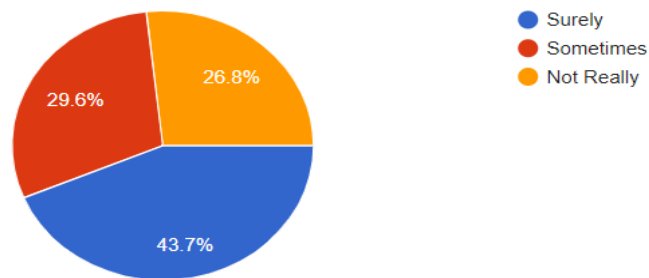


Figure 11: Result of survey question 8.

In order to have a better understanding of such results, we wanted to find out how they are keeping track of their expenses and some answers show that mostly people would use the bank application to do so. Although when you do not have a bank transaction and use cash instead, the record of that specific transaction is not saved on the bank app and there is no option to have a manual input to do such a thing.

EcoExt app comes with a beneficial solution for that by offering to the users the option to manually input such transactions - cash or bank transactions. Another benefit is that EcoExt will explore machine learning as an approach to check what the users are spending their money on and they will be able to have an automatic tracker of what exactly they have been speeding.

In an era of one-click purchasing and automatic billing, EcoExt will make it easier than ever to track the finances and get the receipts by adopting the application. Users will see trends, customised graphs and notice where are their expenses that are misaligned with their values and priorities. They will find themselves naturally spending in ways that don't leave them feeling guilty later, as EcoExt will notify them in which category the money is being spent on as well as using the money that was earmarked for more important uses.

People who track their spending using EcoExt financial application, will know how much they're earning, spending, and saving. Resulting that they will be able to save more and achieve their goals.

#### ***1.4.4.5 Customer Optimization***

Users of EcoExt are able to download the application and then customize it according to their needs, additionally they may use different “purses” such as cash, bank accounts, saving accounts and so on. The benefit of being able to optimize the application is an attractive need that can attract potential users’ attention and also make the application trendier, adapting to the new technologies of the market.

EcoExt users will also have the option to save their invoices and thus have an easy access to them. Facilitating in a way that is best for their needs.

Product classification, where users are able to know where their money is being directed and where it qualifies as in, which category; They will have the option of using a manual input where they will be able to label the category. For example, if you are buying a drink when traveling, you will be able to set that certain amount of money in two different categories - travel and beverage. The product classification fits as another benefit users will have when using the EcoExt app.

Automatic categories - EcoExt users will also benefit from having their invoice evaluated and thus, the application will automatically distribute expenses within categories that the user wishes to be reliable.

Besides, the benefits **for both** - users and companies:

#### ***1.4.4.6 Environment***

In addition to all the environmental benefits already mentioned, the companies and users that adopt EcoExt, can actually benefit the environment as well as the people people who also care about the environment. The companies will benefit from having more customers who care about the case and use EcoExt app. Users therefore, can benefit by not wasting printed receipts on a daily basis.

## 1.5 EcoExt is born

At the beginning of September 2018, five (5) participants and Information Technology Graduate students of CCT College got together with the aim of deciding what end-of-course project would be carried out and conducted by them.

Some innovative ideas were presented to transform the market and build the future with the goal of using modern technologies that could, somehow, be beneficial to a strategic solution, ideas were presented and discussed in which would be the best option, solution that, all team members agreed to develop.

Gabriel Oliveira, our backend developer introduced the idea of developing an application that could help the environment but which also does not exist in the market, so that we could innovate with a justified solution. He comments that: "Every time I enter a shop, I always look at the garbage and realise the absurd amount of paper waste, I thought to myself, there must be some way to stop it."

Every single participant in the team agreed on the innovative idea and started to think about ways to adjust and improve it. We then, started holding weekly meetings so we could organize and plan the project with a better approach and at the first meeting we had to choose what name the new project would be given. Some solutions were presented and a vote was taken using an online tool called Easy Polls. See figure below:

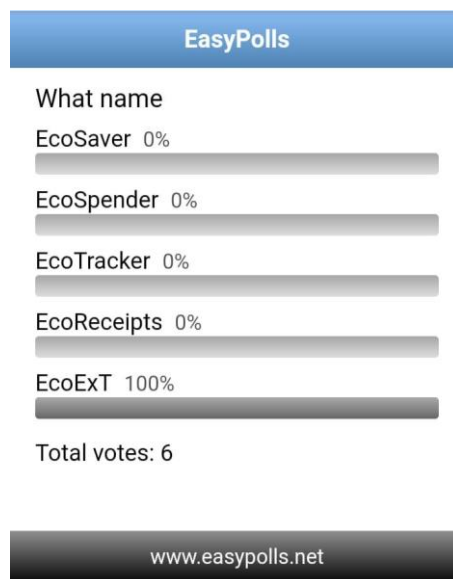


Figure 12: Name chosen to the project.

EcoExt is born with a hundred (100) percent of votes and then on the same meeting we decided who was going to manage the team of EcoExt, another Easy Poll was made in order to do so and Carina Lins was chosen to be the project manager of the team. See figure below:



Figure 13: Project manager chosen to the team.

As every good project, EcoExt also needed a logo and we had the idea to create something that was related to the environment to be used in different needs. Asmer Bracho, our frontend developer came up with an amazing visual solution to it. See figure below:

### **EcoExt Stands for - Ecological Expenses Tracker.**



Figure 14: EcoExT logo.

## **Chapter 2 - System Analysis and Design**

The objective of this chapter will be focused on what EcoExT will have to offer the user, consisting in an analysis of what will be implemented in the system and what functional requirements are necessary for the production of such. Also, how the software analysis was created and how it will be implemented.

When analysing what a system will have to offer, some requirements are necessary and essential for a project to be driven to success. Generally, these requirements are a formal agreement between a client and a provider, both of whom are working together to achieve the same goal. The quality of the product is measured by the details of the analysis made and also by the risks that may arise and be mitigated, those risks could be financial risks, as well as the deadline for delivery of the product and cases that arise during the implementation process.

### **2.1 Functional Requirements**

In the next scenarios, we investigate what EcoExt has to offer to its users and a functional analysis is made to reflect the potential of the application.

#### **2.1.1 Purpose**

As mentioned earlier in the first chapter, the main purpose of EcoExT is to reduce the paper receipts and track what has been consumed along with the purchases done by the end users. An overview of the system is that, with this software application working, users will not need to have a physical invoice and yes, they can have access to a digital invoice within the application that will be interconnected with a scanner to scan the QR code which will generate the invoice that will be stored inside the application itself. The end users therefore will always be available to access and track their purchases.

#### **2.1.2 Overall description**

The vision of the product and to provide the customer with the availability and easy access of scanning tokens to be able to retrieve the paper receipts, the vision also includes contributing to the environment, reducing the use of paper and innovating with a product that does not exist in the market. Another vision is also to create an application that makes a difference, using new technologies and building a future so that people can facilitate their day to day use of our application.

Having an application, the tracks your speeding and scans the QR codes in order to gather the digital receipt allow the users of EcoExt to reduce the need for paper receipts and additionally have the track of what they have purchased themselves. Whether it is purchasing eleven at a time, they will scan and get the token from the raspberry LCD. All EcoExT users are able to get the receipt and the raspberry is able to generate different authentication tokens for different transactions so they do not get the same receipt of a different user.



The product's vision is to provide the customer the availability and easy access of scanning tokens with the aim of users being able to retrieve the digital receipts, the vision also includes contributing to the environment by reducing the use of paper and with an innovating product in the market that still does not exist. One more vision to be mentioned is that EcoExT aims to bring a differential product application that inspire on building the future so that people can have an understating of the environment issue and with a plus of facilitating their daily activities by using EcoExT application.

Having an application that tracks your speeding and scans the QR codes in order to gather the digital receipt allow the users of EcoExt to reduce the need of paper receipts and additionally have the track of what they have purchased for themselves. Whether it is purchasing once at a time where they will scan and get the token from the raspberry pi LCD. Every EcoExT users are able to get the receipt and the Raspberry Pi is able to generate different authentication tokens for different transactions so they do not get the same receipt of a different user.

### **2.1.3 Detailed description of the functionality of the proposed system**

1. If the user of EcoExt does not have an account, they are not able to log in. They must create an account in order to use the application.
2. Once they are logged in, they are able to see the home page and records pages, they will also be able to scan a QR code ad also create a new record.
3. The user is able to see a sidebar with some information of Home, Settings, Reports, Notifications. They are also able to share information and log out.
4. The records are stored in the application with the information of the product that was purchased, which purse it has been stored and the amount spent on that specific transaction. Under the records, there is also a filter tab and the users are able to filter anything that the user types and contains data into it.
5. The receipt gathered is reported with the Establishment Name, address and further information along with the amount spent on the transaction and the number of the receipt. Also, the products are described and a subtotal amount of what was spent on that transaction.
6. The user is able to create new records, the inputs can be manually added manually with inputs of amount of transactions.
7. The users are able to create different purses and add and delete transactions.
8. The users are able to filter by the date on the calendar and the dates of purchases.
9. The raspberry pi aims to be integrated with the POS system and the POS sends information to the raspberry Pi.

10. The raspberry pi sends the transaction to the server, the server stores the transaction and generates a token that represents a transaction back to the raspberry pi to be displayed on the LCD.
11. The mobile application sends a request to the server in order to validate the token that has been generated. If the token is authenticated, then the user gets a confirmation with the transaction and the server sends a confirmation to the raspberry pi. If the token is not authenticated, then the user gets an Error message which says that the QR code scanned is not an EcoExT QR Code.
12. The QR Code is able to be authenticated only by EcoExT users and can only be scanned once at a time.

EcoExt include some boundaries and restrictions of data access and versioning of the application, the boundaries are show in the next project specifications. The value of creating those specifications is in the optimization of the development process and the Agile methodology which contains the testing and implementation along with the development process. All the functionalities of EcoExT is to be approved by the users and ensures that the developers are building what the customers wants.

#### **2.1.4 Checklist for subsequent stages**

In the future staging, EcoExT aims to have more features and also the users are going to be able to have a premium account where they can have more access to premium features.

1. Reports
2. Notifications
3. Geolocation - as a subsequent stage, the users will able to use the application in different geolocations.
4. Budgets - as a subsequent stage, the users will be able to set budgets of how to spend their money.
5. Planned Payments - The users will able to plan an specific payment such as payment of bills to a certain time.
6. Debts - the users will be able to check their debts.
7. Loyalty Card - if the user has a loyalty card of an establishment, they will be able to attach the card to the application so that they can have more discounts.
8. FAQ - A help guide - The users will be able to have a read through a help guide and frequently asked questions.

The team aimed to develop a consistent system which is able to attend many users in different places and different types of transactions. The Proposed system is not able to be fully consistent since the first version is only a prototype, so future features is aimed as subsequent stages of the applications.

## 2.2 System design diagrams

On the next following pages, the application design that has been done with the aim of planning the system to be implemented and utilizing then, the tools needed in order to better design the proposed system.

### 2.2.1 Use Cases

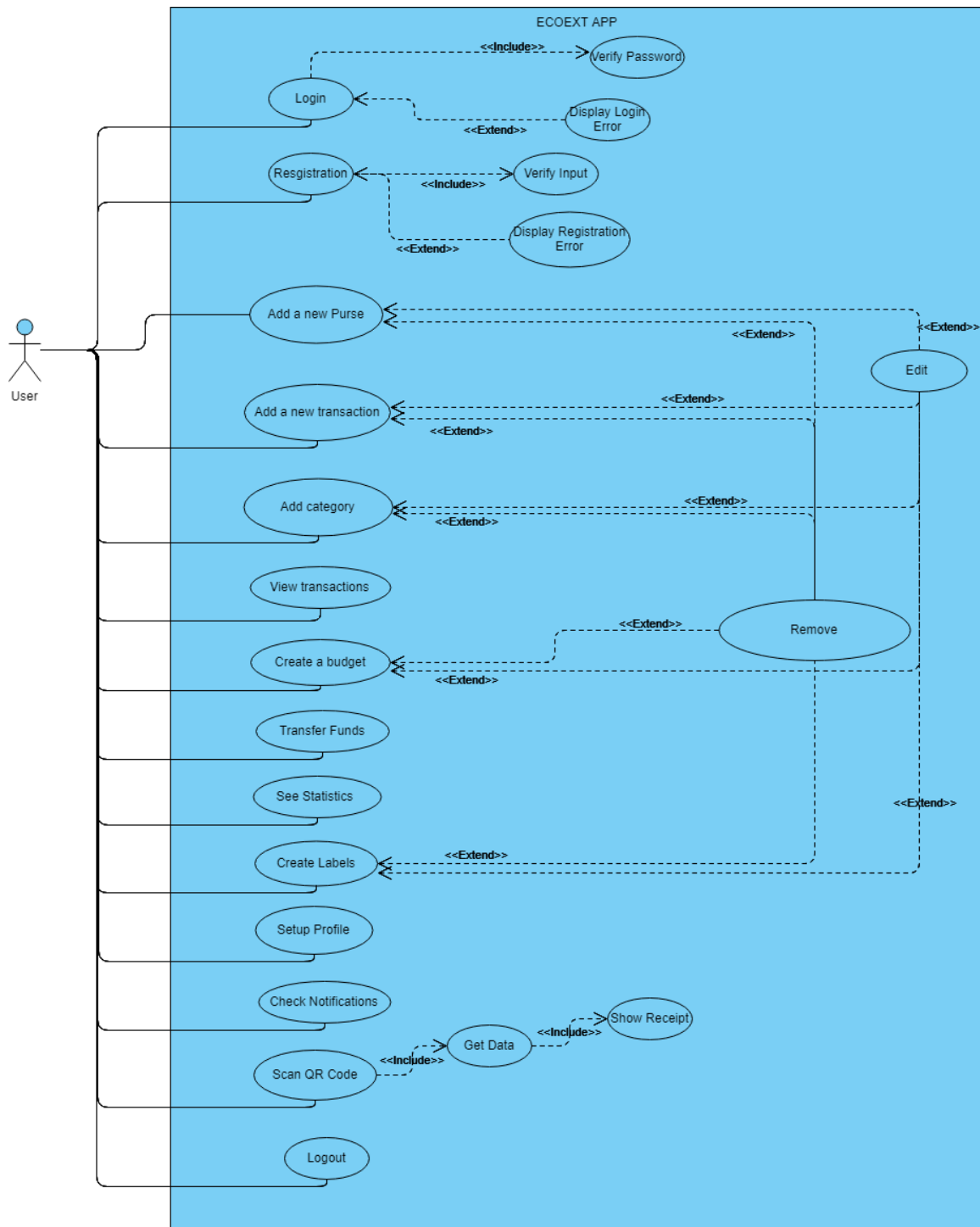
For EcoExT, we have two different Use Cases, the first one is for the EcoExt Application. Each use case includes three main elements:

- **Actors.** These are the users outside the system that interact with the system.
- **System.** The system is described by functional requirements that define an intended behaviour of the product.
- **Goals.** The purposes of the interaction between the users and the system are outlined as goals.

For EcoExT Application, the user interacts with the system by:

1. Login, which the user is able to verify the password and if It does not work, will display a login error.
2. Register, which the user is able to input a registration and if It does not work, will display a registration error.
3. Add a new purse, which the user is able to edit this new purse.
4. Add a new transaction, which the user is able to edit this new transaction and remove the transaction.
5. Create a budget, which the user is able to edit and remove this budget.
6. Transfer funds, which the user is able to transfer funds from one purse to another.
7. See statistics, which the user is able to check the statistics of the money that has been spent.
8. Create labels, which the user is able to create labels, edit and delete the labels.

9. Setup Profile, which the user is able to setup a new profile, edit and delete it.
10. Check Notification, which the user is able to check the notifications of the application.
11. Scan QR Code, which the user is able to connect to the Raspberry Pi by scanning the QR Code generated and also gathering the receipt.
12. Logout, which the user is able to log out of the system.



The second Use Case diagram is for the EcoExT Raspberry Pi client application.

For EcoExT Raspberry Pi client application, the actor interacts with the system by:

1. Authenticate, the Raspberry Pi authenticates.
2. Receive and collect data from Point of Sales system, which the Raspberry Pi is able to communicate with the POS System which in this case is another actor and will send back the information to the initial actor.
3. Create transaction, which the Raspberry Pi is able to create a new transaction.
4. Generate QR Code, which the Raspberry Pi is able to generate a new QR code.
5. Get the user ID, which the Raspberry Pi is able to interact with the Server which is another actor and the Server will send back information to the initial actor which is the Raspberry Pi.
6. Sends Transaction to the server, which the Raspberry Pi is able to interact with the Server which is another actor and the Server will send back information to the initial actor, the Raspberry Pi.

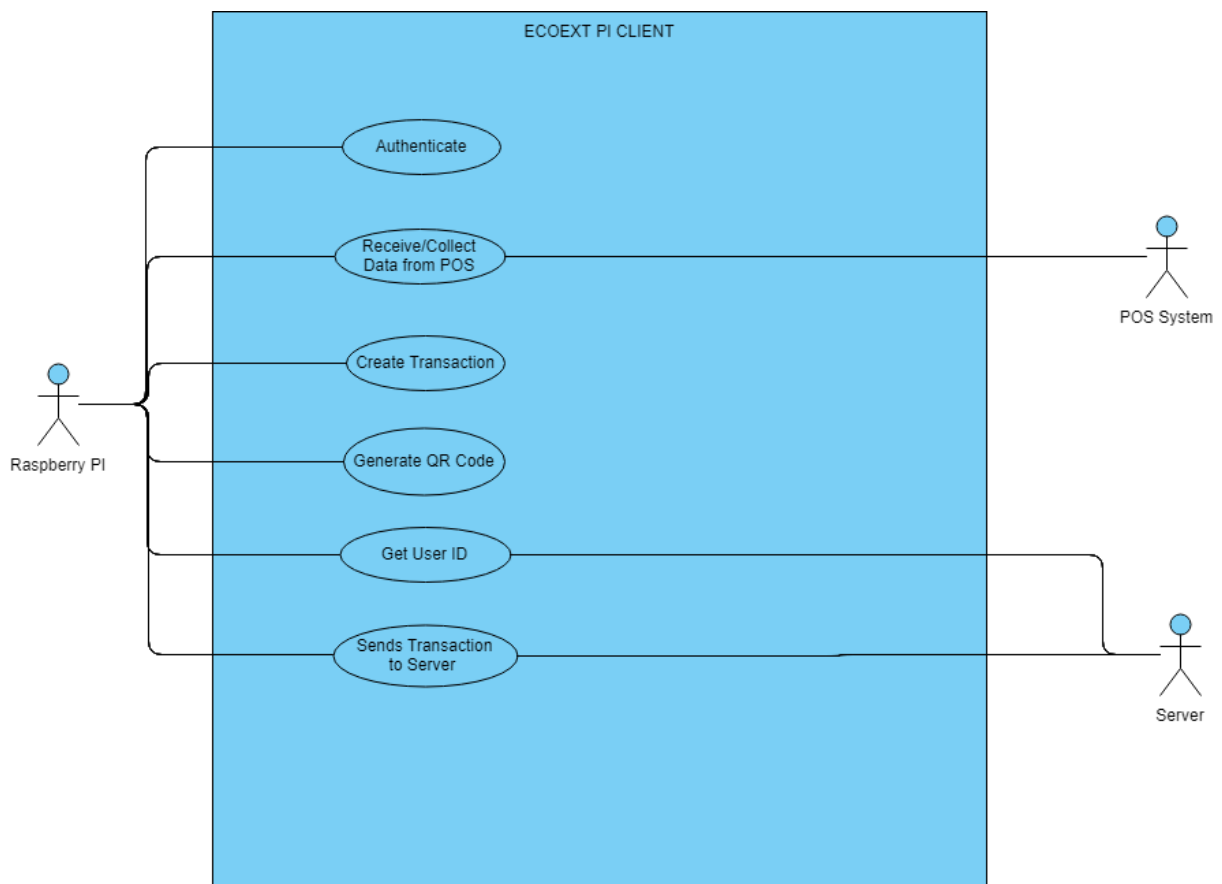


Figure 16: EcoExT Raspberry Pi Use Case Diagram.

### 2.2.2 Use Cases Specifications

For EcoExt, we also designed use case specifications, that represents the sequence of events along with other information that relates to the use cases. The following use case specifications will have information such as descriptions, pre and post interaction condition, basic operation path, alternative path and exception path.

#### 2.2.2.1 USE CASE SPECIFICATION - LOGIN

Use Case Identification and History			
Use Case ID:	ecoext_login		
Use Case Name:	user Login	Version No:	0.0.1
End Objective:	User logs in the system		
Created by:	Gabriel Oliveira	On (date):	19/10/2018
Last Update by:		On (date):	
Approved by:		On (date):	
User/Actor:	User		
Business Owner Name:		Contact Details:	
Trigger:	User		
Frequency of Use:			

Preconditions
ecoext_register must have been true

Basic Flow		
Step	User Actions	System Actions
1	User inputs credentials	

2		System validates credentials
2a		Credentials are invalid <ul style="list-style-type: none"> <li>• informs user</li> </ul>
3	User is sent to main page	

#### 2.2.2.2 USE CASE SPECIFICATION – REGISTER

Use Case Identification and History			
Use Case ID:	ecoext_register		
Use Case Name:	User registration	Version No:	0.0.1
End Objective:	User is registered		
Created by:	Gabriel Oliveira	On (date):	19/10/2018
Last Update by:		On (date):	
Approved by:		On (date):	
User/Actor:	User		
Business Owner Name:		Contact Details:	
Trigger:	User		
Frequency of Use:			

Preconditions
User is not registered

Basic Flow		
Step	User Actions	System Actions
1	User input email and password	
1a		check if email is already registered

1b		Informs user
2		User is registered
2b		User is sent to login page

Post conditions
1. User has been granted with access

Special Requirements
1. user must not be registered

### ***2.2.2.3 USE CASE SPECIFICATION - ADD PURSE***

Use Case Identification and History			
Use Case ID:	ecoext_addPurse		
Use Case Name:	Add purse	Version No:	0.0.1
End Objective:	User creates a new purse		
Created by:	Gabriel Oliveira	On (date):	19/10/2018
Last Update by:		On (date):	
Approved by:		On (date):	
User/Actor:	User		
Business Owner Name:		Contact Details:	
Trigger:	User		
Frequency of Use:			

Preconditions
---------------



ecoext\_register and ecoext\_login must have been true

Basic Flow		
Step	User Actions	System Actions
1	User selects the option to create a new purse	
2		System creates a new purse
2b		User has exceeded purses' limit <ul style="list-style-type: none"> <li>• informs user</li> </ul>
3		System displays the main view with the recently created purse
Post conditions		
1. A New purse has been added to user's account		

#### 2.2.2.4 USE CASE SPECIFICATION - ADD TRANSACTION

Use Case Identification and History			
Use Case ID:	ecoext_user_addTransaction		
Use Case Name:	User adds a transaction	Version No:	0.0.1
End Objective:	A transaction is added to a purse		
Created by:	Gabriel Oliveira	On (date):	19/10/2018
Last Update by:		On (date):	
Approved by:		On (date):	
User/Actor:	User		
Business Owner Name:		Contact Details:	
Trigger:	User		
Frequency of Use:			

Preconditions
ecoext_register; ecoext_login and at least one ecoext_addPurse

Basic Flow		
Step	User Actions	System Actions
1	User hits the button add transaction	
2		Display the view add transaction
3	User select purse(s)	
4	User selects type of transaction (Income / Expense)	
4a	User selects multiple forms of payment	
4b	User informs for each type of payment the amount	
4c	User selects currency for each type of payment	
5	User selects category(ies)	
6	User selects date of transaction	
7	User selects labels	
8	User hist add transaction	
9		Validates transaction input
9a		Input is invalid <ul style="list-style-type: none"> <li>• informs user</li> </ul>
10		Updates row transaction / purse
11		Display new transaction

Post conditions
1. A new transactions is added to user's account

### 2.2.3 User Stories

For EcoExT we have created four (4) different User Stories, which represents a natural description of the system proposed along with the perspective of an end user or the business users of EcoExT systems.

The first user stories below is on the customer perspective side:

*As a Customer, I want to be able to get e-receipts so that I can save paper and help the environment.*

Figure 17: User Stories Number 1.

The second user stories below is on the sales establishments side:

*As a Sales Establishment, I want to issue e-receipts so that I can reduce the costs of printing paper receipts (paper rolls, ink, etc) and help the environment.*

Figure 18: User Stories Number 2.

The third user stories below is on the end user side:

*As a User of EcoExT, I want to be able to get data of my purchases so that I can*

Figure 19: User Stories Number 3.

The fourth user stories below is also on the end user side:

*As a User of EcoExT, I want to set/get alerts so that I can be better guided on how to spend my money.*

Figure 20: User Stories Number 4.

The proposed user stories were designed to be implemented in the software, as part of the Agile methodology in order to capture description of features that can be created in a simplified version of requirements.

#### **2.2.4 Activity Diagrams**

EcoExT has designed activity diagrams which visually represents a flow control of the system and the model process of the activity diagrams below are similar to a flowchart, describing the use case diagrams above, each of the activity diagram is specific to the sequence of the user interaction in the application from the beginning state to the final stage of the application.

**2.2.4.1 The first activity diagram below represents the Login:**

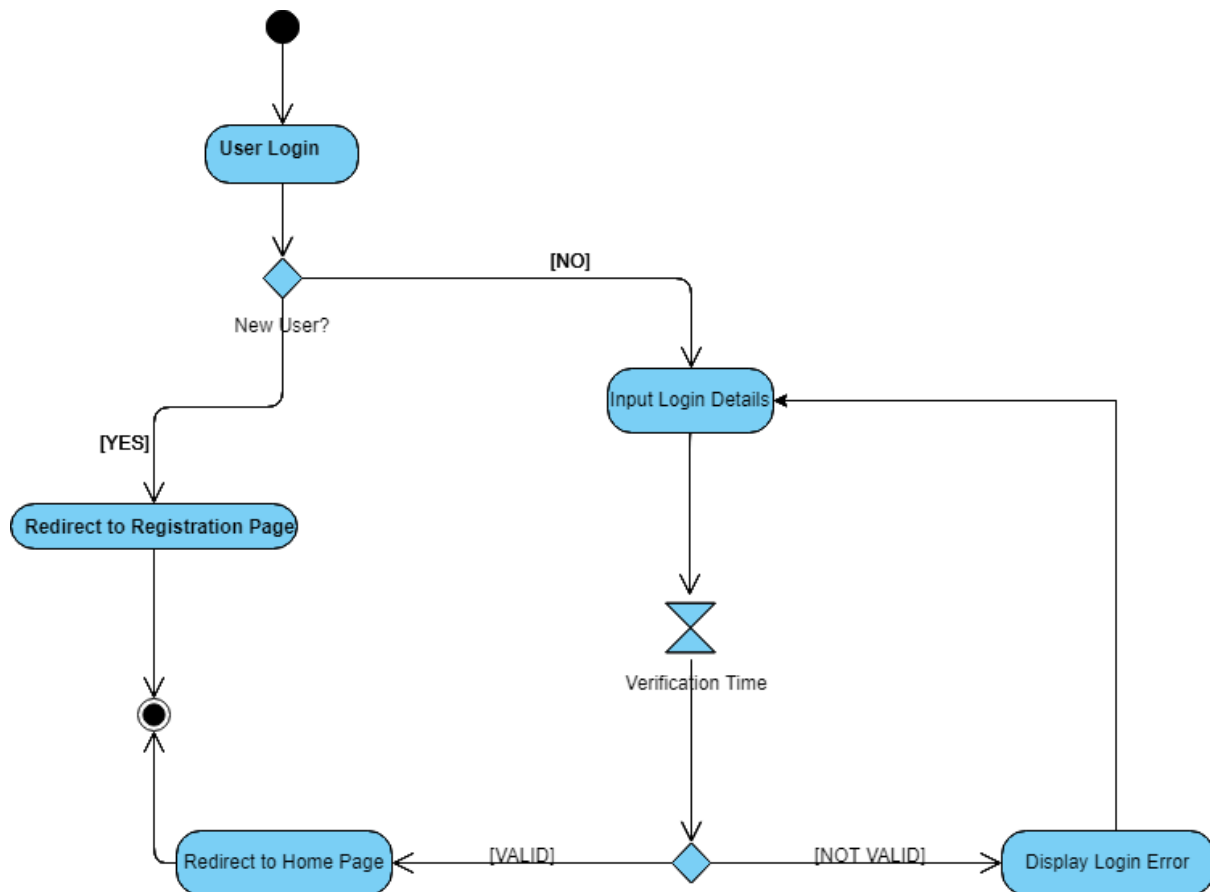


Figure 21: Login Activity Diagram.

**Description:**

- Start point.
- Activity, user logs in. Is it a new user?
- Yes, it is a new user. Redirect to Registration Page.
- No, it is not a new user. Input Login Details.
- Activity, user is redirected to the registration page, end.
- Activity, user inputs login details, time for verification or display login error.
- Activity, the login details are valid, redirect user to home page, end.
- Activity, the login details are not valid, display login error, end.

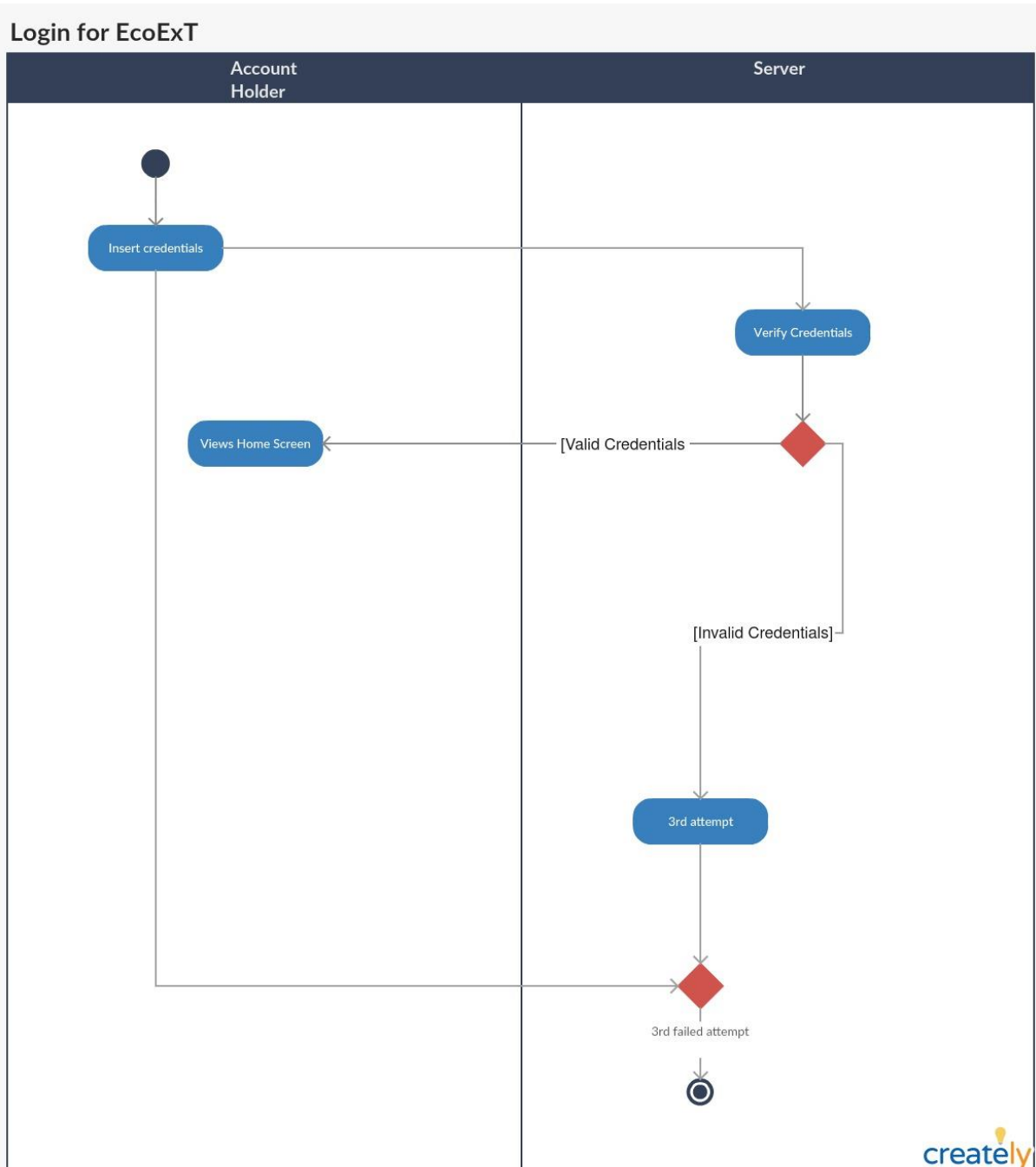


Figure 22: Login Activity Diagram.  
Credits: [www.createely.com](http://www.createely.com)

This is also another Login activity diagram above, where you can see another logic into it.

**Description:**

- start point.
- Account holder activity, insert credentials.

- Server activity, verify credentials.
- Are credentials valid?
- Yes, activity account holder, view home screen.
- No, attempt credentials failed, end.

**2.2.4.2 The second activity diagram below represents the Registration:**

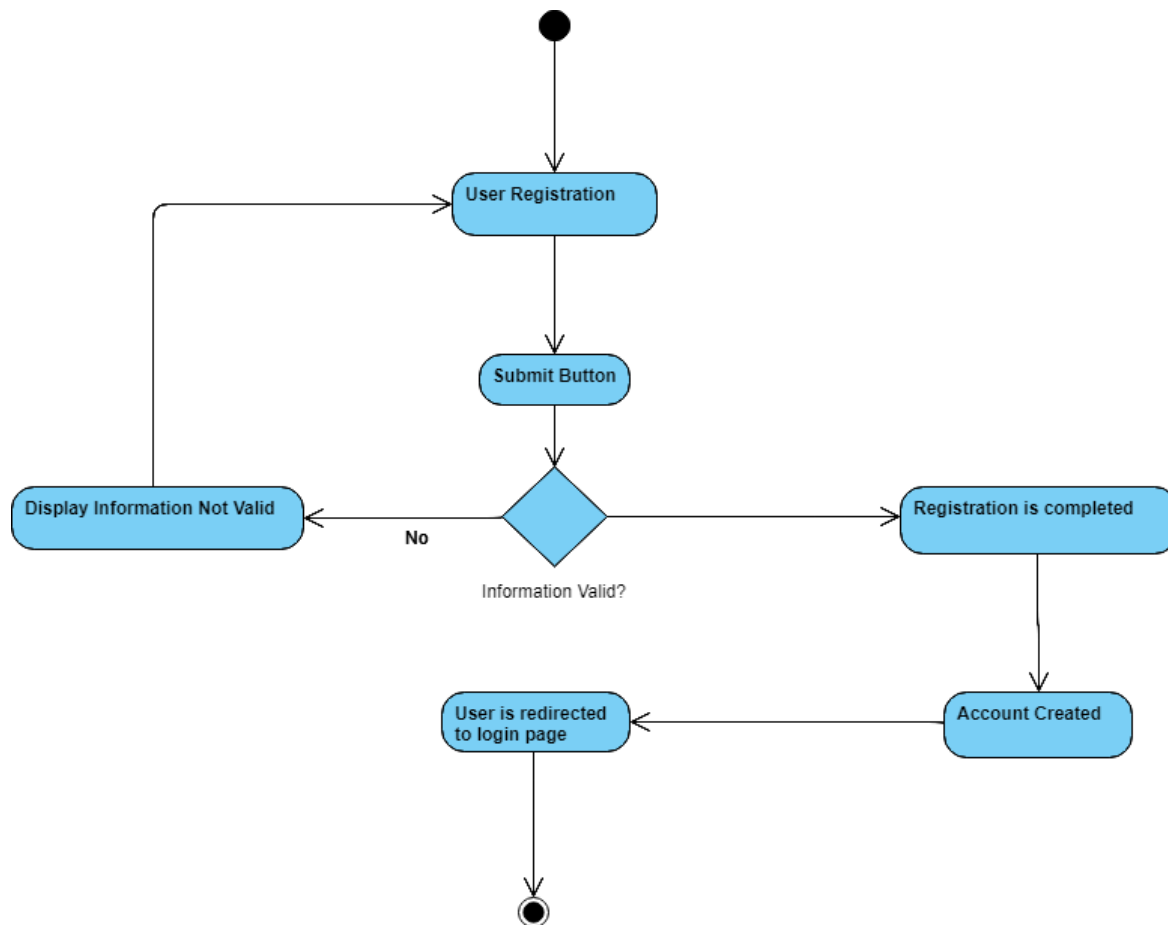


Figure 23: Registration Activity Diagram.

**Description:**

- Start point.
- Activity, user registration.
- Activity, user submits the registration.
- Is the information entered valid? Yes, registration is completed.

- No, displays that the information is not valid and goes back to the user registration.
- Activity, registration is completed, an account is created.
- Activity, account is created, user is redirected to the login page, end.

**2.2.4.3 The third activity diagram below represents the Purse:**

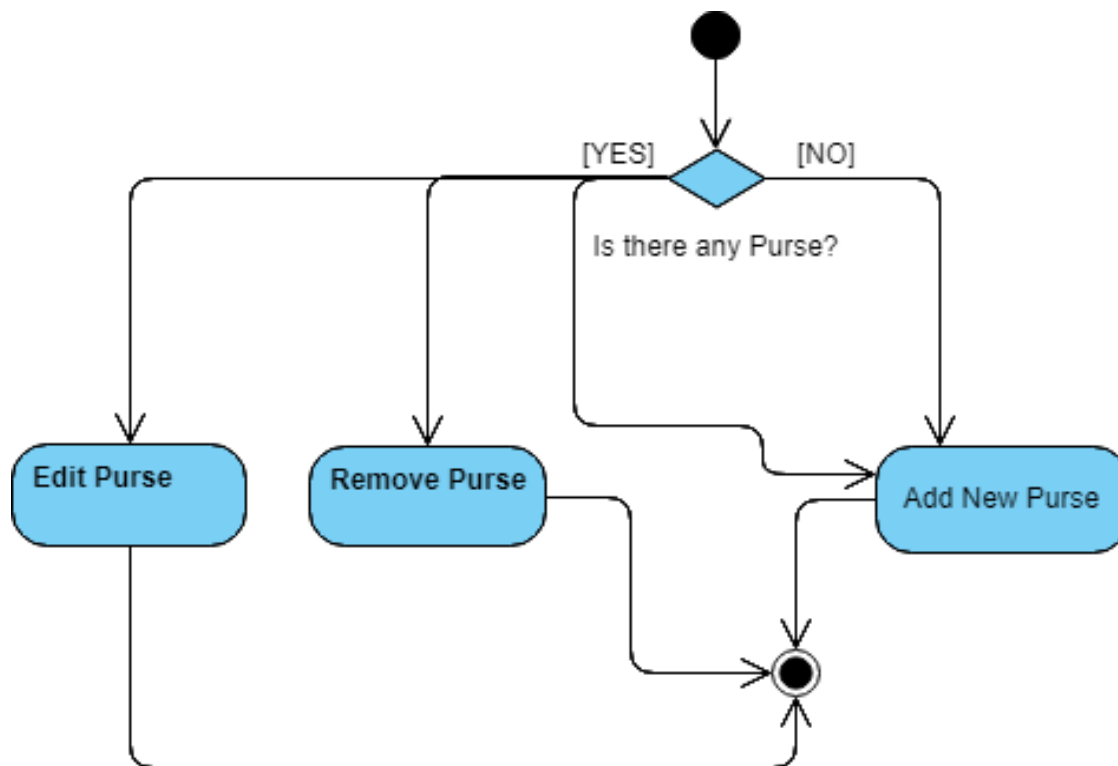


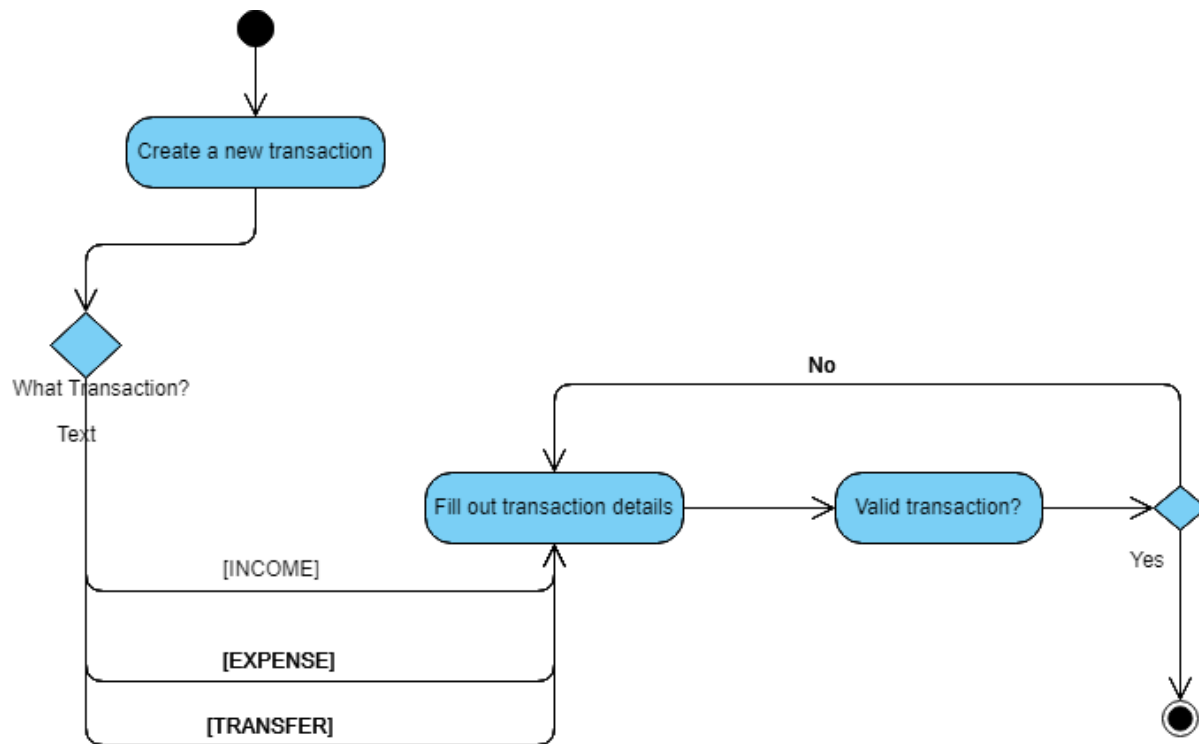
Figure 24: Purse Activity Diagram.

**Description:**

- Start point.
- Is there any purse?
- Yes, edit or remove the existing purse and add a new purse.
- No, add a new purse.
- Activity, edit existing purse, end.
- Activity, removing existing purse, end.
- Activity, adding a new purse, end.



**2.2.4.4 The fourth activity diagram below represents the Transaction:**



*Figure 25: Transaction Activity Diagram.*

**Description:**

- Start point.
- Activity, create a new transaction.
- What transaction? Income, expense or transfer transaction?
- Activity, fill out transaction details.
- Activity, is the transaction valid?
- Yes, the transaction is valid, end.
- No, the transaction is not valid, fill out the transaction details.

2.2.4.5 The fifth activity diagram below represents the Categories:

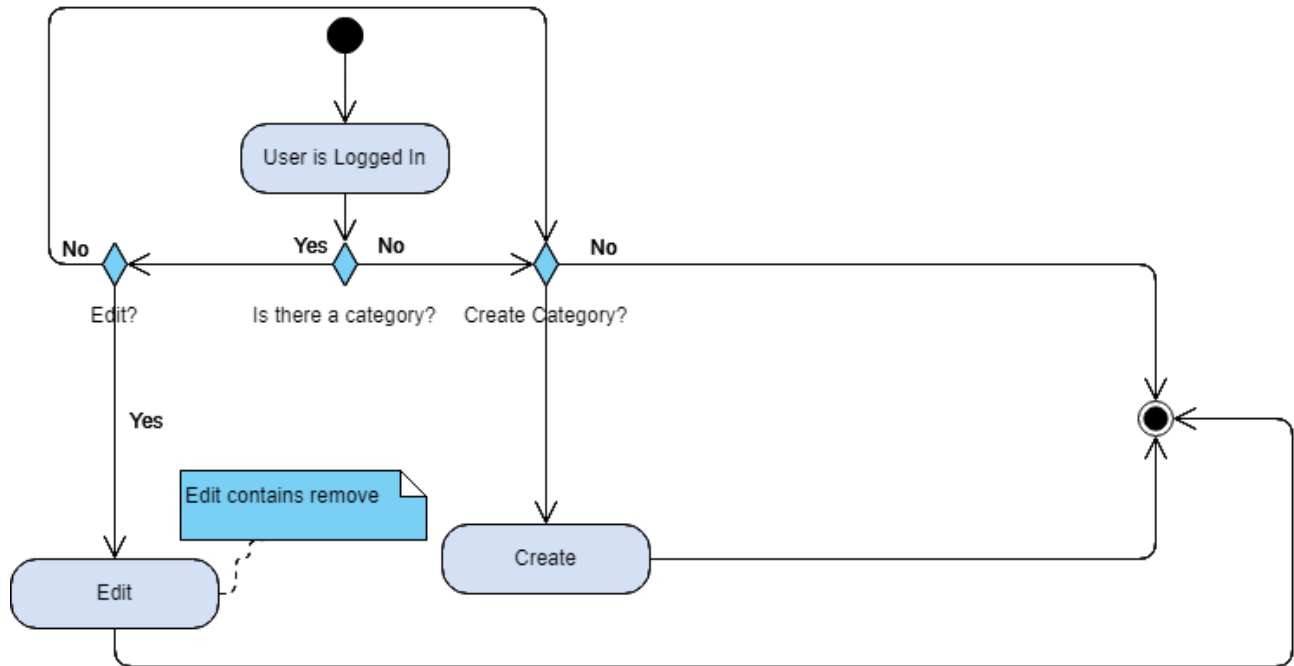


Figure 26: Categories Activity Diagram.

**Description:**

- Start point.
- Activity, the user is logged in.
- Is there a category?
- Yes, the category exists, edit the category.
- No, the category does not exist, creates a new category.
- Edit the category?
- Yes, edit the category, edit contains remove, end.
- No, do not edit the category, create a new category.
- Create a new category?
- Yes, create a new category, end.
- No, do not create a new category, end.

**2.2.4.6 The sixth activity diagram below represents the Budget:**

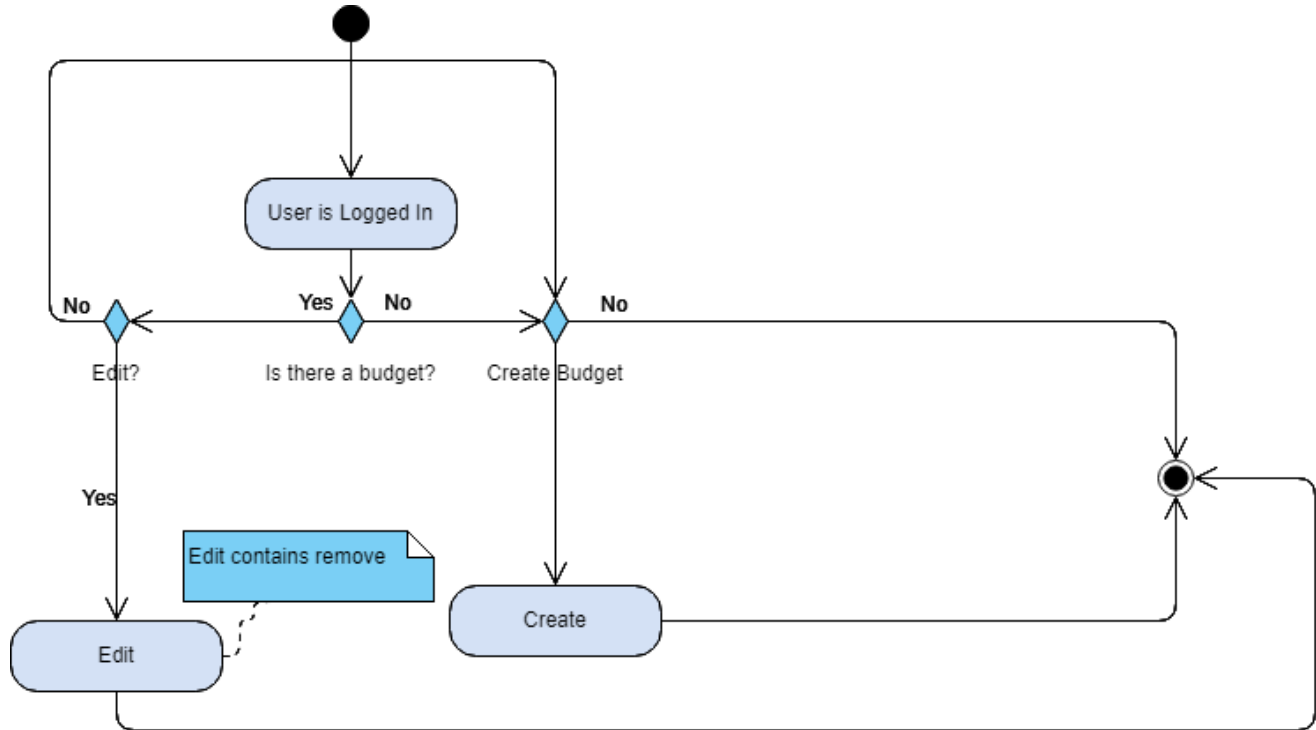


Figure 27: Budget Activity Diagram.

**Description:**

- Start point.
- Activity, the user is logged in.
- Is there a budget?
- Yes, the budget exists, edit the budget.
- No, the budget does not exist, creates a new budget.
- Edit the budget?
- Yes, edit the budget, edit contains remove, end.
- No, do not edit the budget, create a new budget.
- Create a new budget?
- Yes, create a new budget, end.
- No, do not create a new budget, end.

**2.2.4.7 The seventh activity diagram below represents the View Statistics:**

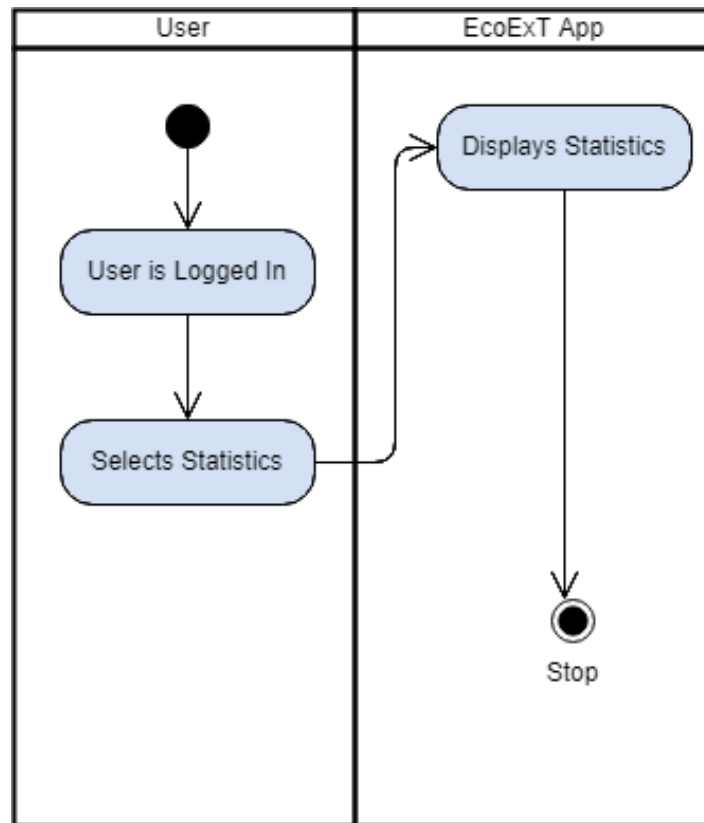


Figure 28: View Statistics Activity Diagram.

**Description:**

- Start point.
- User Activity, the user is logged in.
- User Activity, select view statistics.
- EcoExt App Activity, display statistics, end.

2.2.4.9 The ninth activity diagram below represents the Setup Profile:

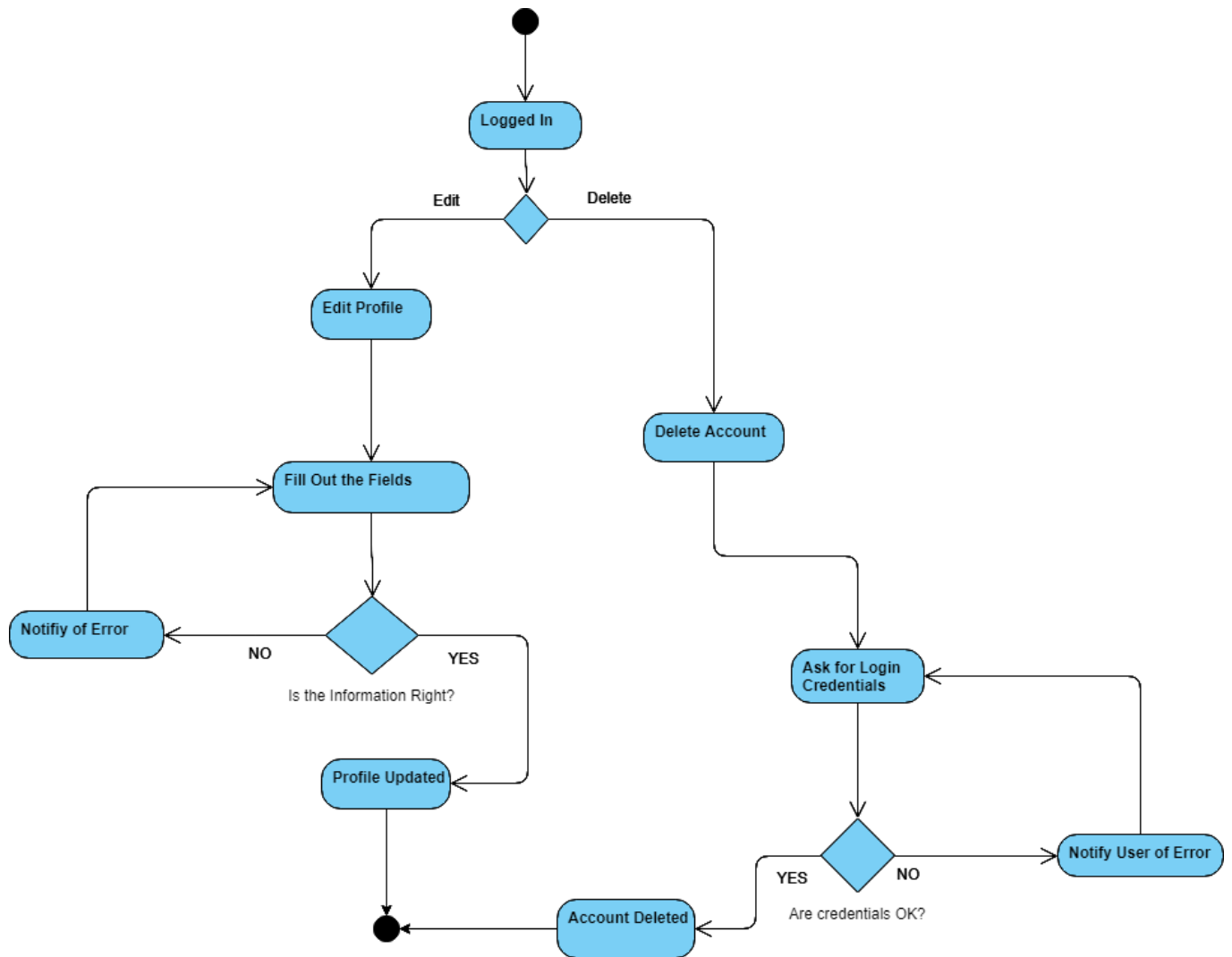


Figure 29: Setup Profile Activity Diagram.

#### Description:

- Start point.
- Activity, the user is logged in.
- Edit Profile?
- Yes, activity, edit profile.
- Delete Profile?

- Yes, activity, delete profile.
- Activity, fill out the fields to edit profile.
- Is the information, right?
- Yes, activity, profile updated, end.
- No, activity, notify of error, fill out the fields to edit profile.
- Activity, delete account.
- Activity, ask for login credentials.
- Are credentials ok?
- Yes, credentials are ok, activity, account deleted, end.
- No, credentials are not ok, activity, notify user of error, ask for login credentials.

**2.2.4.10 The tenth activity diagram below represents the Check Notifications:**

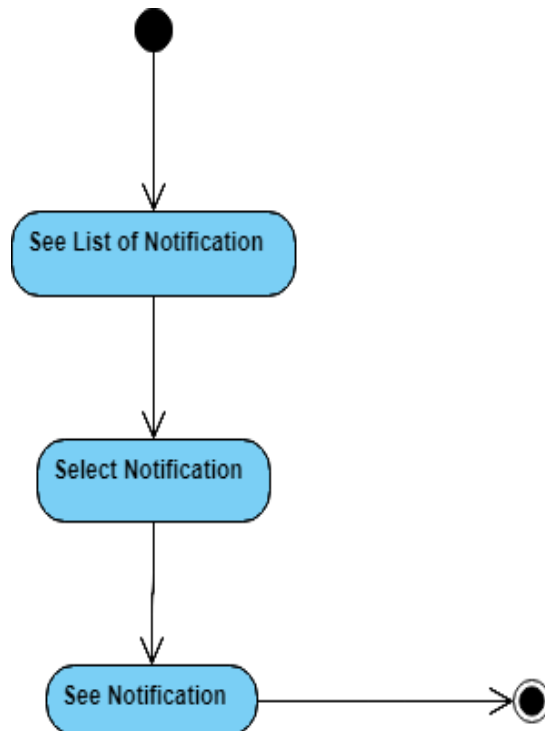


Figure 30: Check Notifications Activity Diagram.

**Description:**

- Start point.
- Activity, see list of notifications.
- Activity, select notification.
- Activity, see notification, end.

2.2.4.11 The eleventh activity diagram below represents the Scan QR Code:

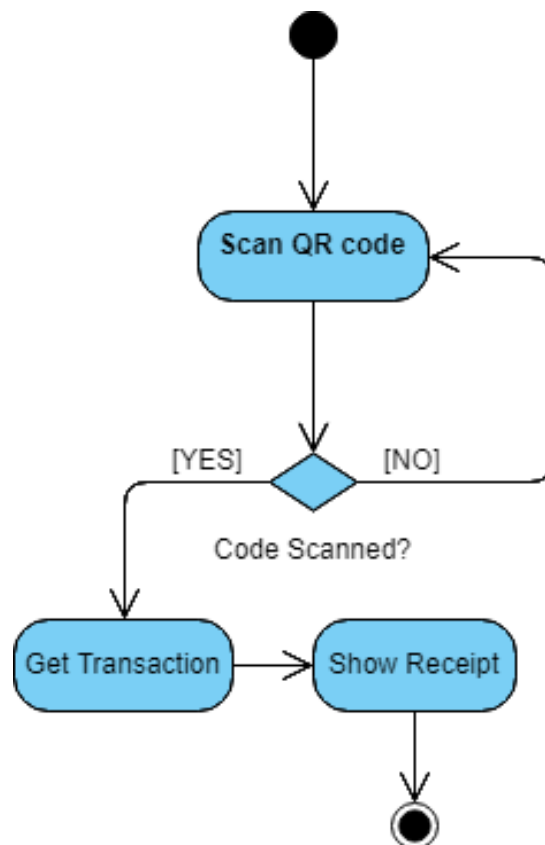


Figure 31: Scan QR Code Activity Diagram.

**Description:**

- Start point.
- Activity, Scan QR Code.
- Is the QR Code scanned?

- Yes, activity, get transaction, show receipt, end.
- No, activity, scan QR Code.

**2.2.4.12** The twelfth activity diagram below represents the Logout:

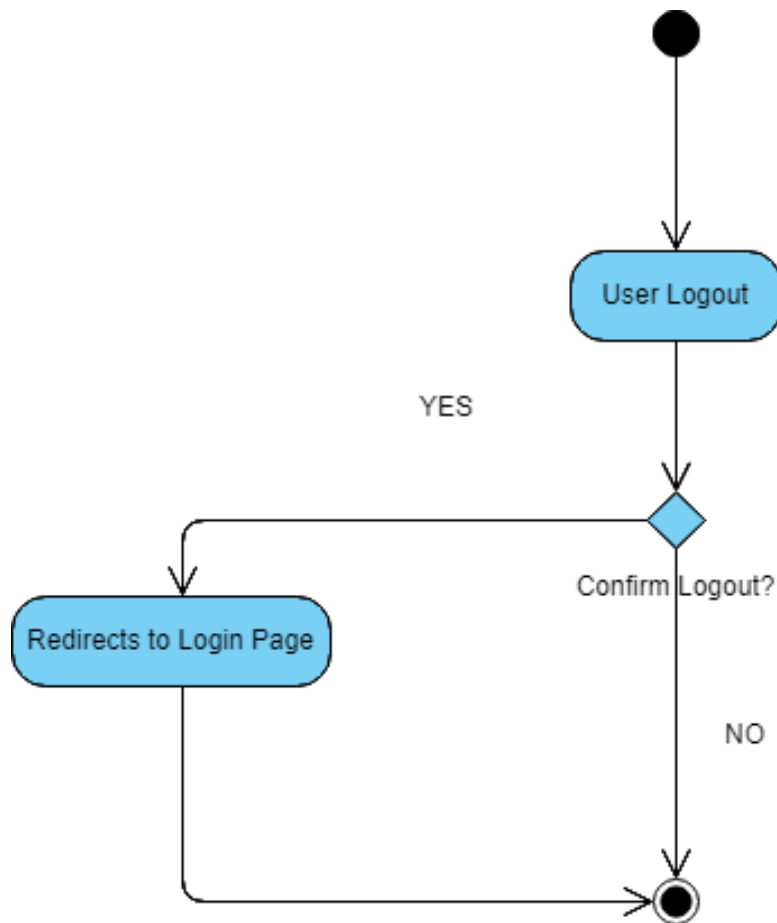


Figure 32: Logout Activity Diagram.

**Description:**

- start point.
- Activity, user logout.
- Confirm logout?
- Yes, activity, redirects to login page, end.



- No, do confirm logout, end.

**2.2.4.13 The thirteenth activity diagram below represents the Raspberry Pi - Create Transaction:**

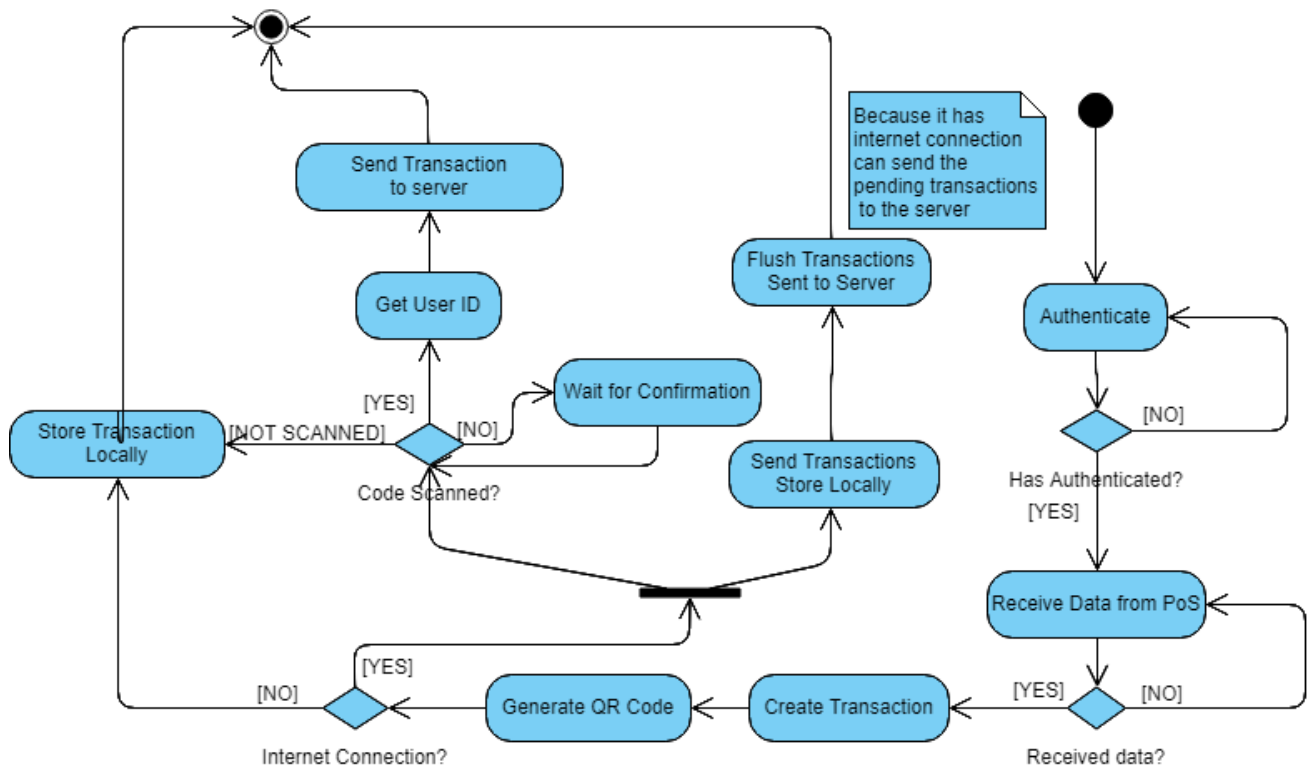


Figure 33: Raspberry Pi - Create Transaction Activity Diagram.

#### Description:

- start point.
- Activity, authentication.
- Has it been authenticated?
- Yes, activity, receive data from PoS (Point of Sale).
- No, it has not authenticated, activity, authenticate.
- Activity, receive data from PoS.
- Received data?
- Yes, data is received, create transaction.

- No, data is not received, activity, receive data from PoS.
- Activity, create transaction, generate QR code.
- Activity, generate QR Code.
- Is there internet connection?
- Yes, there is internet connection, activity, send transactions stored locally, flush transactions sent to server, end.
- No, there is not internet connection, activity, store transaction locally, end.
- Has the QR code being scanned?
- Yes, activity, get user ID, send transaction to server, end.
- No, activity, wait for confirmation, store transaction locally, end.

### 2.2.5 Overview Diagram

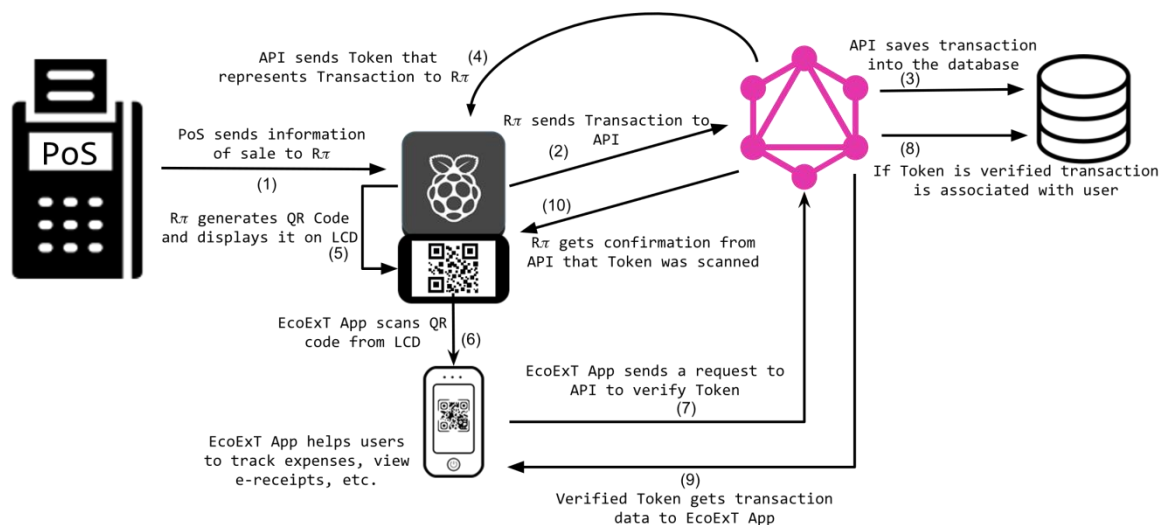


Figure 34: Overview Diagram of EcoExT.

An overview diagram of the whole system above, where it indicates few steps:

1. PoS sends information of the sales to the Raspberry Pi.
2. Raspberry PI sends transaction to the API.

3. API saves transaction into the database.
4. API sends token that represents transaction to the Raspberry Pi.
5. Raspberry Pi generates the QR Code and displays it on LCD.
6. EcoExT App scans QR code from LCD.
7. EcoExT App helps users to track expenses, view e-receipts, etc.

Defining the system or application that is going to be implemented in the raspberry pi. So far, what is known is that the system has to be able to generate e-receipts in a Mobile Application after scanning a QR Code from a Raspberry Pi, or any other device with a screen that is running the Application (it was decided to call it Pi App because it is implemented in a Raspberry Pi but the main idea is not to limit the app to it). The figure above shows the diagram of the flow of information within our system. Here one can see the flow of information from the PoS System, passing through the Raspberry Pi and the API, to the Mobile Application.

### **2.2.6 Sequence Diagram**

A sequence diagram of the information's flow within the EcoExT system when a QR Code is scanned successfully, and in the figure below it shows an activity diagram on how the Raspberry Pi application is intended to behave when functioning in a particular establishment.

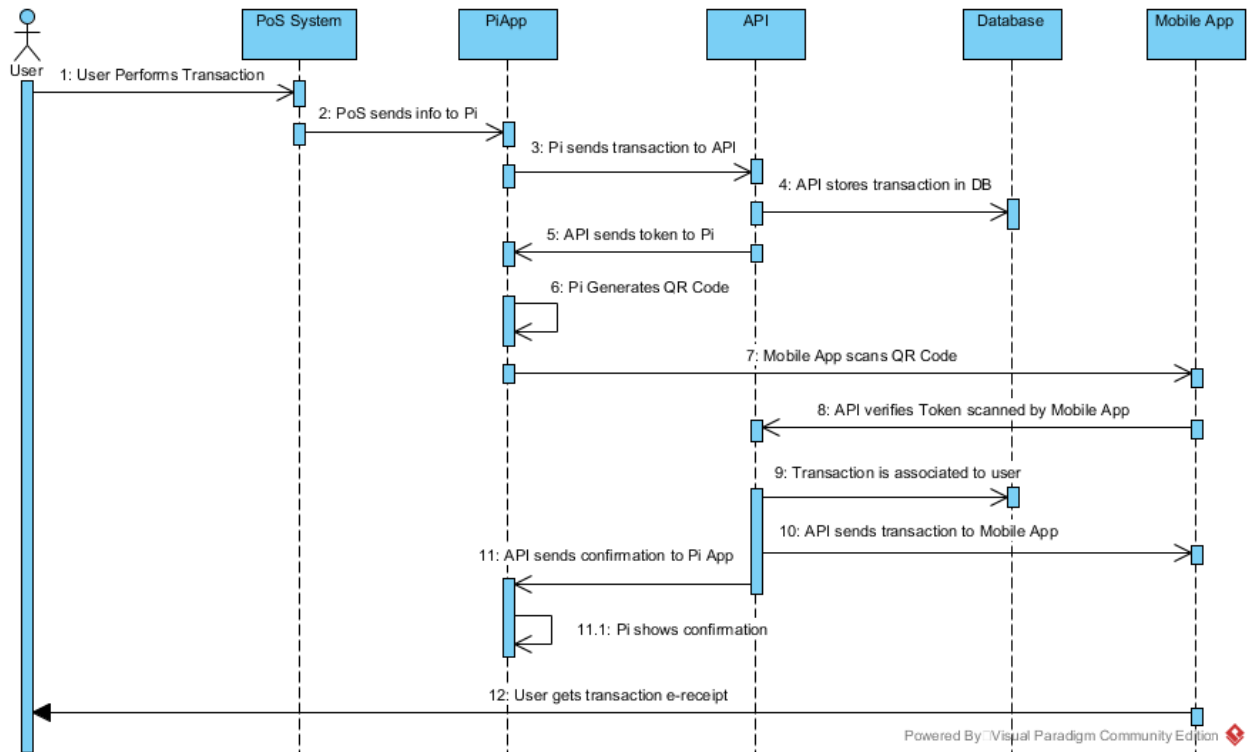


Figure 35: EcoExT System Sequence Diagram.

## 2.3 Logical design of the raspberry pi application

The following shows the basic design of the raspberry pi application for, both, the frontend and the backend sides.

### 2.3.1 Backend Design

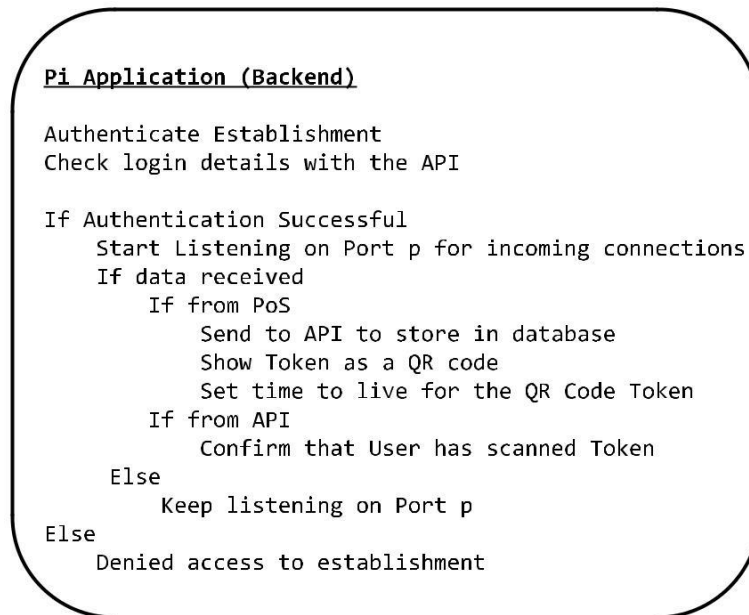
The backend side of the application is going to be in charge of listening for incoming connections from the PoS System. This system is going to send the transaction information in a particular format, so the Raspberry Pi application is able to send this information to the API, that is built using GraphQL, so that the transaction can be stored in the database.

In order to make the Raspberry Pi App listening to the PoS System and the API, the following modules are going to be used:

- **Socket module:** for creating a listening an application that receives incoming connections from the PoS for any transaction, and from the API for confirmation when someone scans a QR code from the Pi LCD.
- **Selectors module:** for creating a queue when multiple connections are received at the same time in the Raspberry Pi. This is to avoid any conflicts when multiple connections are established at the same time.

- **Requests module:** for establishing connections with the API when sending a transaction to it. Using this module allows us to connect to the API build is GraphQL. It not only limited to that, it can be a REST API, but the beauty of this library is that allows us to connect to both kinds, what it matters is the end point (One for GraphQL several for REST) that is defined when creating the connection.
- **Json module:** for manipulating the data sent to the API and received from it.

The figure below shows the pseudo code for the raspberry pi backend application.



**Figure 36: EcoExT System Sequence Diagram.**

All the modules mentioned above will help the Raspberry Pi App to generate connections with all the parties involved. But, how the QR Code is going to be generated? This is going to be achieved using the **module qrcode** together with the **base64** and the **PIL** modules. The base 64 module is going to allow us to create a URL safe token that can be understood by all the devices and the **PIL** is to generate the QR figure. It is important to highlight that the module qrcode is used by the raspberry pi app, but further in development it was decided to delegate the token creation to the API since this one is going to be in charge of authenticating all tokens created for security reasons. And to show the QR Code the same PIL module is used but this is done in the front end of the application.

Now, for authenticating the establishment into the EcoExT System is used the same procedure as when sending the transaction from the Pi App into the API. In the figure above we see a pseudo code of the working of the backend side of the application.

### 2.3.2 Frontend Design

Now, for the frontend design the Tkinter Framework was used. This is a very easy to use modules of UI that can be implemented in a different number of languages, and since the desired language was Python 3, this was the perfect Framework to build the Pi App UI.

The main idea of this is to show a home screen that is going to be there meanwhile there are not incoming connections from the PoS system, see figure 1.8. Once, the PoS system sends the transaction to the API and should transition to the QR code window, this window is set to live for 20 seconds, see figure 1.9, and if the QR code is not scan within the window of time then a window that shows an unsuccessful scanned QR Code, see figure 1.10. Finally, if the QR is scanned, then a successfully scanned window is shown in the Pi LCD, see figure 1.11. The figures below are the wireframes for the frontend side of the application.

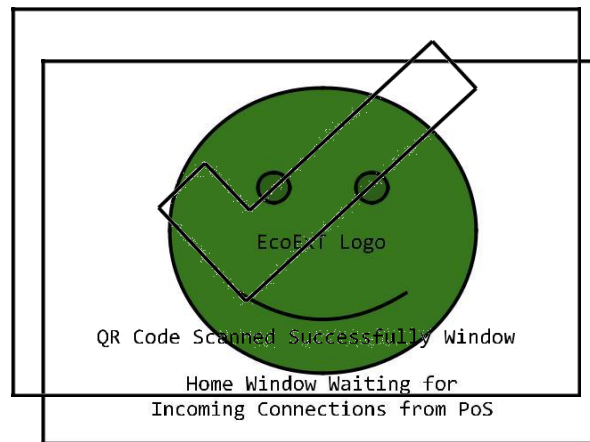


Figure 37: Raspberry Pi Application home window.

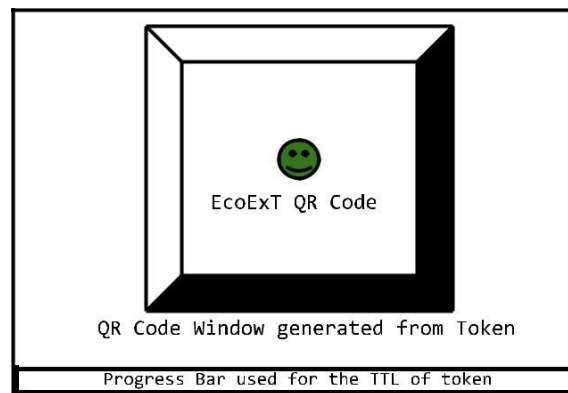


Figure 38: Pi Application QR Code window.

Figure 39: Raspberry Pi Application successfully scanned window.

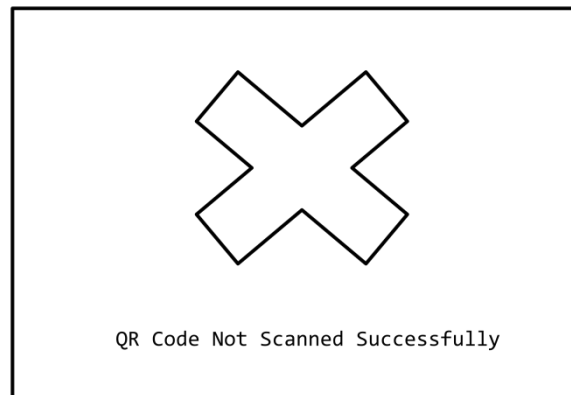


Figure 40: Raspberry Pi Application not scanned window.

Now, in the figure below it is shown the pseudo code of the frontend application functionality. There, it can be seen the logical design of the frontend side of the application.

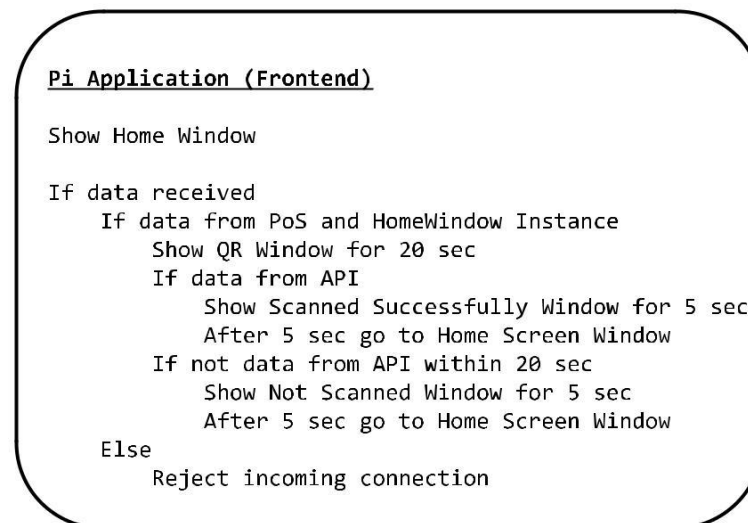


Figure 41: Raspberry Pi Application frontend pseudo code.

## 2.4 User interface design

EcoExT has designed a user interface and user experience design in order to facilitate the tasks of implementation and drawing attention to the usability and graphic design for the user interaction and final user experience, determining then, wireframes of how the product will look

like, which contains buttons, input fields, colours, readable texts and if it is finger accessible, as well as how difficult or easy for the end users is to navigate through the application.

The final state appearance of EcoExT application might not look the same, although, the implementation and development will be based on the following user designs.

### 2.4.1 Wireframes

EcoExt wireframes were part of the design planning for the front-end of the Android application, the wireframes below represent blueprints and a visual guide of the version to be developed with the purpose of arranging elements to best accomplish our purpose according to the diagrams previously designed.

#### 2.3.1.1 Login wireframe below:

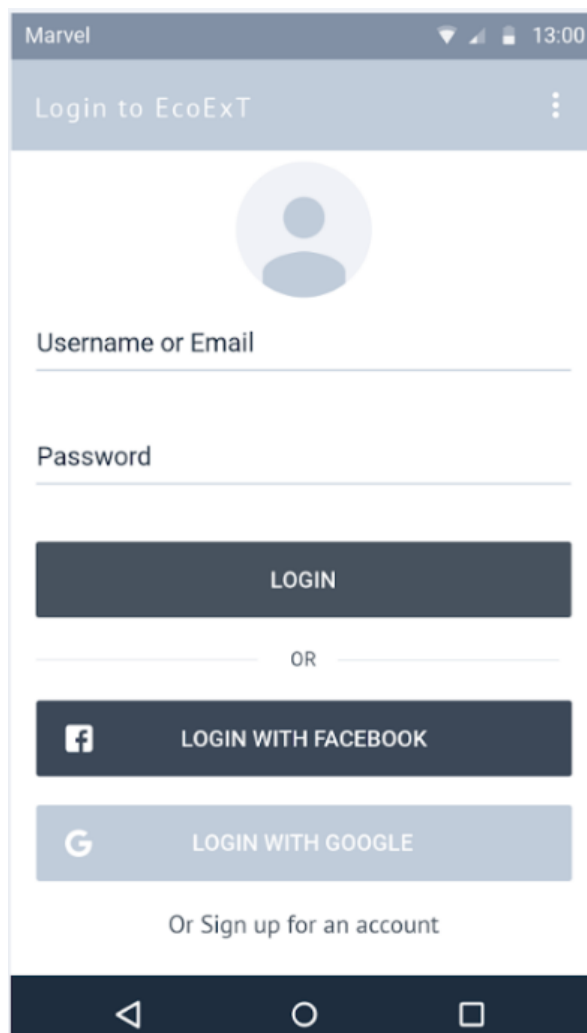


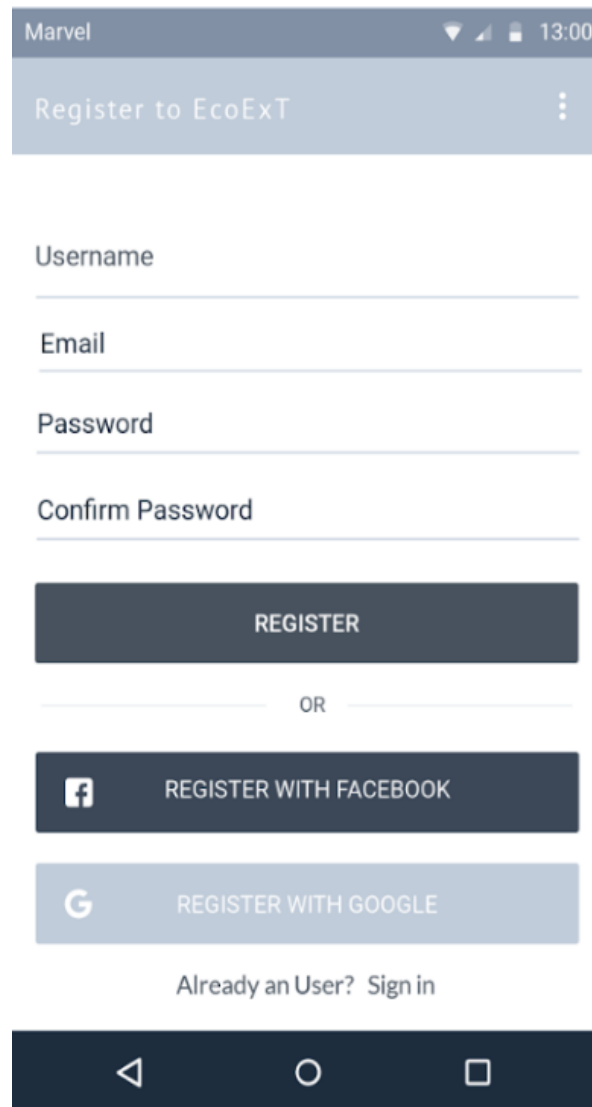
Figure 42: Login wireframe.



Login for EcoExT application.

- Input fields for username or e-mail.
- Login button.
- Login with Facebook.
- Login with Google.
- Sign up for an account.

### ***2.3.1.2 Registration Wireframe:***



The wireframe shows a mobile application interface for registration. At the top, a status bar displays 'Marvel' and the time '13:00'. Below this is a header bar with the text 'Register to EcoExT' and a three-dot menu icon. The main content area contains four input fields labeled 'Username', 'Email', 'Password', and 'Confirm Password'. Below these fields is a dark blue button labeled 'REGISTER'. Underneath the button is a horizontal line with the text 'OR' in the center. Below this line are two more buttons: a dark blue button with a Facebook icon and the text 'REGISTER WITH FACEBOOK', and a light blue button with a Google icon and the text 'REGISTER WITH GOOGLE'. At the bottom of the main content area is the text 'Already an User? Sign in'. The entire interface is framed by a dark blue bar at the bottom containing three navigation icons: a back arrow, a circle, and a square.

Figure 43: Registration wireframe..

Registration for EcoExT application.

- Input fields for username, email, password and confirm password.
- Register button.
- Registration with Facebook.
- Registration with Google.
- Already a user? Sign in.

### 2.3.1.3 Sidebar wireframe:

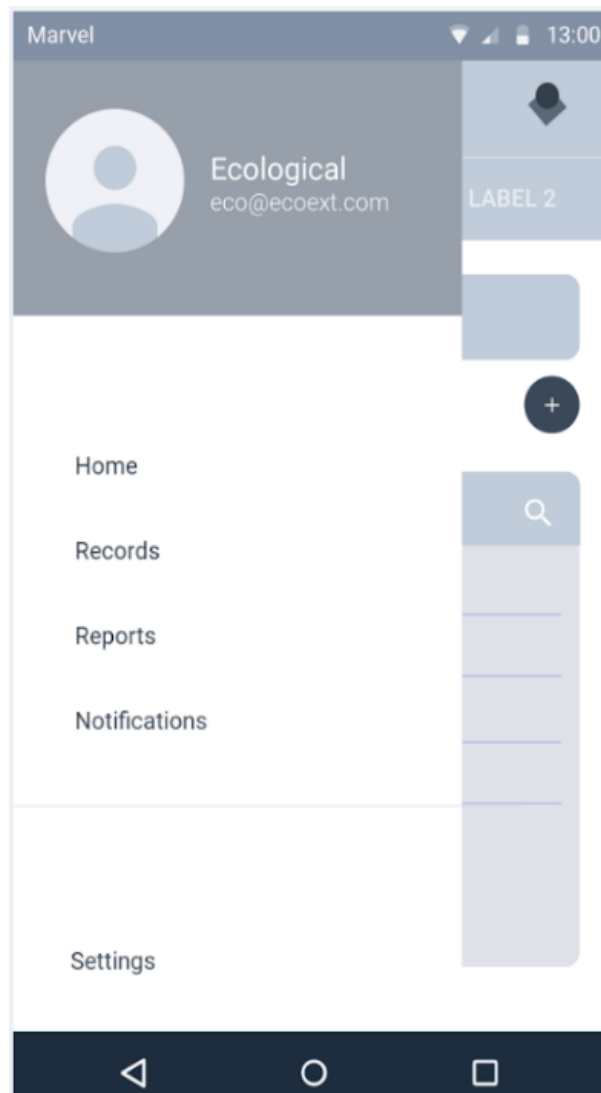


Figure 44: Sidebar wireframe.

Sidebar wireframe for EcoExT application.

Sidebar displays photo, name and email.

Sidebar displays home, records, reports and notification options.

Sidebar displays settings options.

2.3.1.4 Records wireframes:

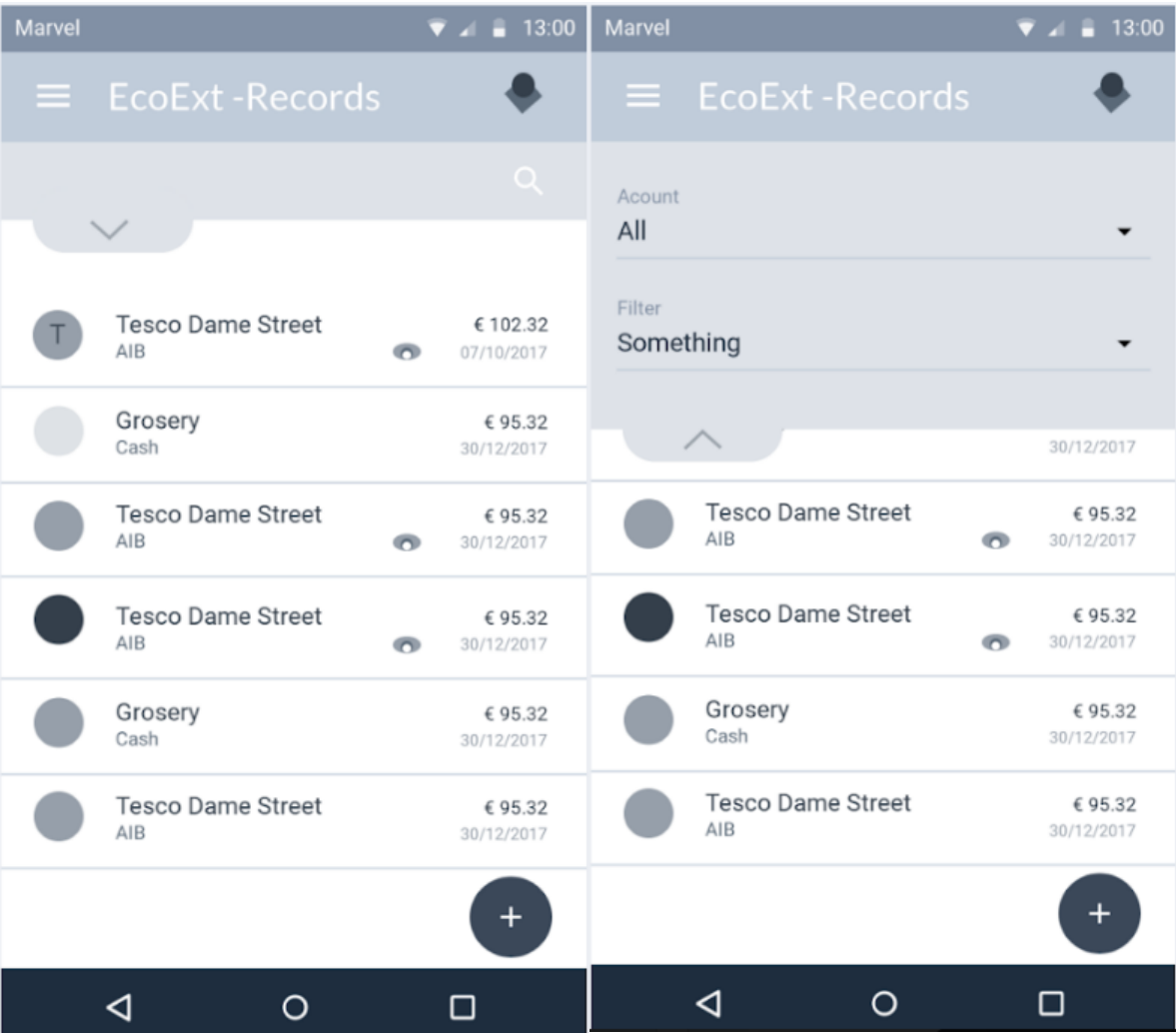


Figure 45: Records wireframes.

Records wireframes for the EcoExT application.

Display records of transactions, name, place of purchase, way of payments, amount spent and date of purchase.

Filters for each account used and something else that can be filtered.

#### ***2.3.1.5 Transfer funds wireframes:***

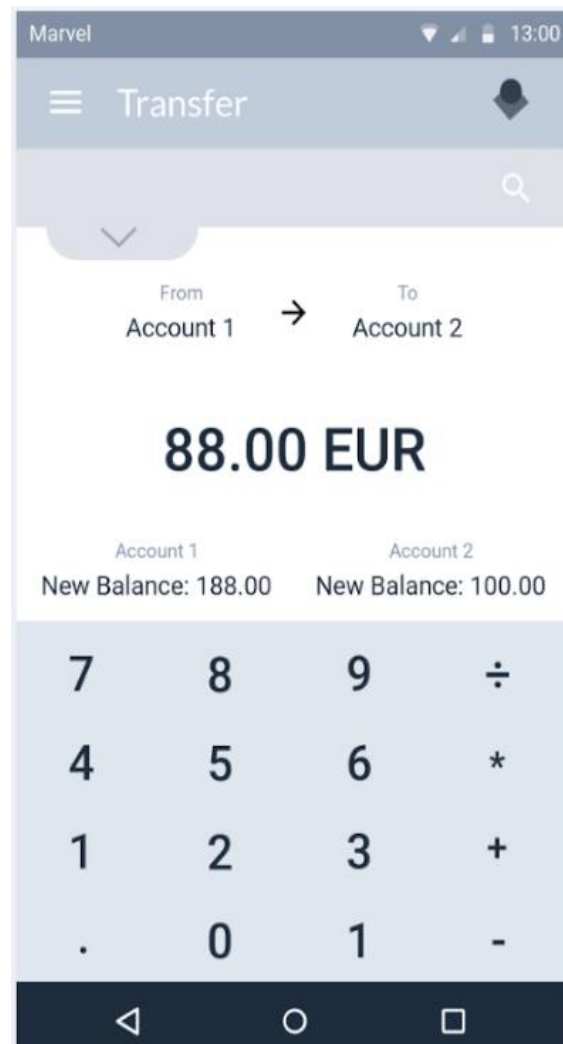


Figure 46: Transfer funds wireframes.

Transfer funds wireframe for EcoExT application.

Accounts are displayed and amount that is being transferred.

The balance of accounts is displayed.

Keyboard to type amount is enabled.

2.3.1.6 Homepage wireframe:

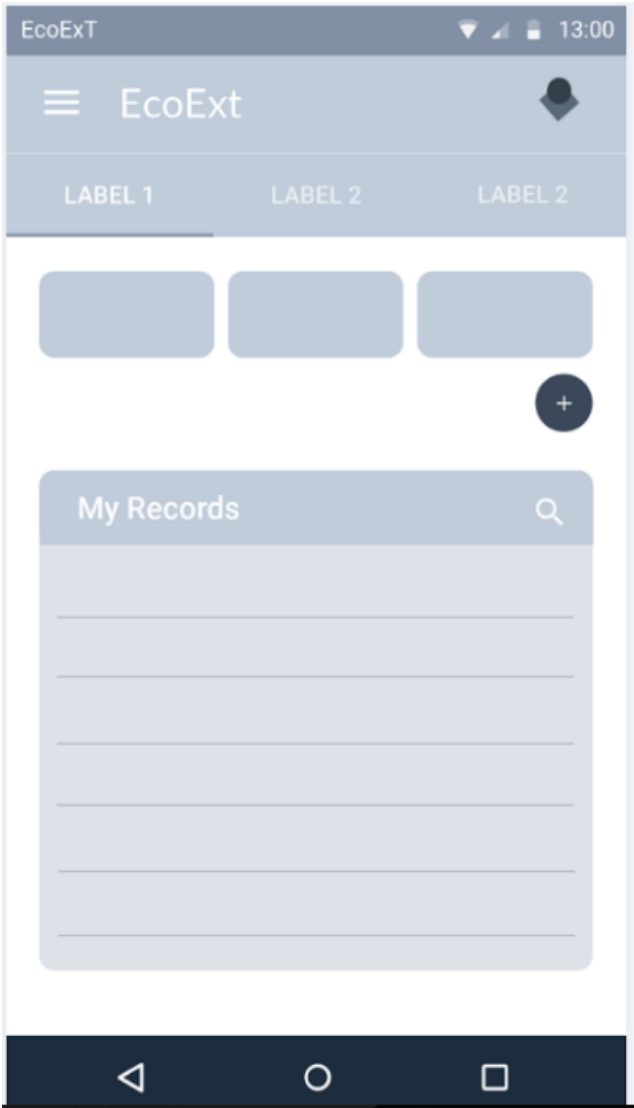


Figure 47: Homepage wireframe.

Homepage wireframe for EcoExt application.

Some visual design displaying existing purses.

A list of recent records.

2.3.1.7 Notifications wireframe:

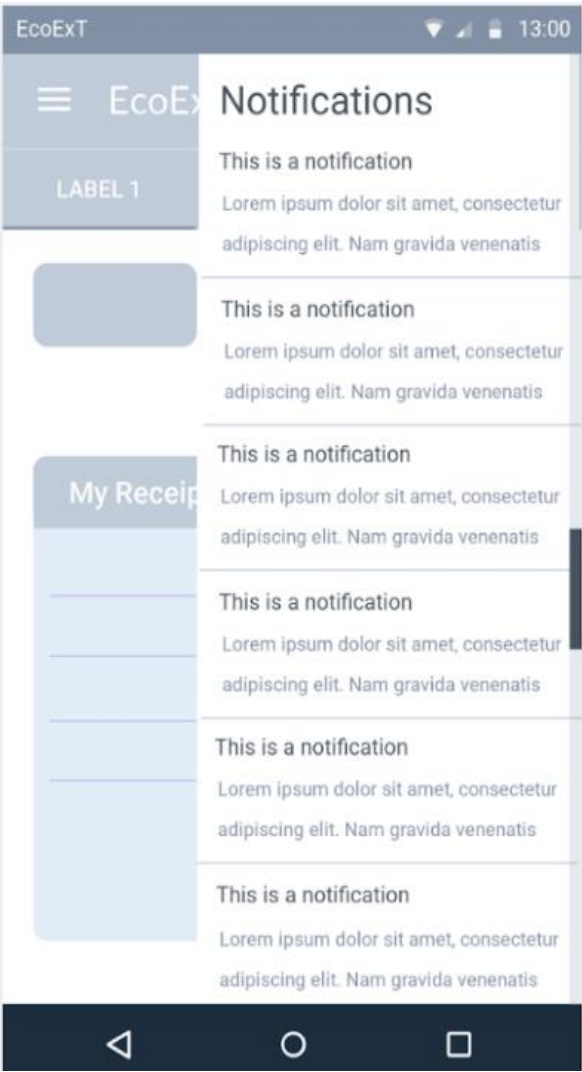
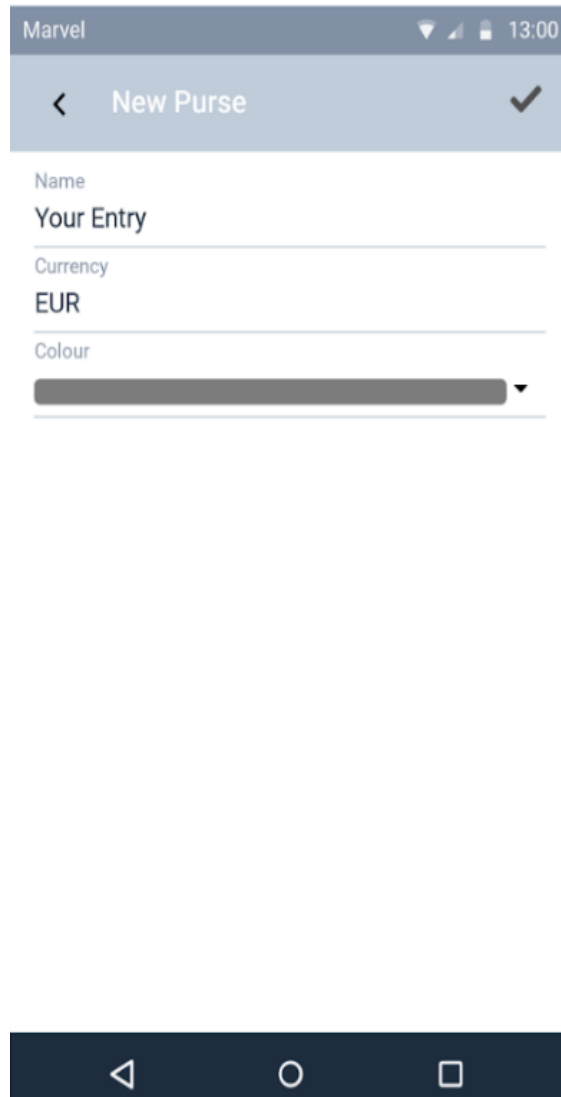


Figure 48: Notifications wireframe.

Notification wireframe for EcoExT application.

Display a list of notifications.

### 2.3.1.8 Creating a new purse wireframe:



The wireframe shows a mobile application interface for creating a new purse. At the top, there is a status bar with the text 'Marvel' and icons for signal, battery, and time (13:00). Below this is a header bar with a back arrow, the title 'New Purse', and a checkmark icon. The main content area contains three input fields: 'Name' with the text 'Your Entry', 'Currency' with the text 'EUR', and 'Colour' with a dark grey dropdown menu. The bottom of the screen features a dark blue navigation bar with three icons: a back arrow, a circle, and a square.

Figure 49: Creating a new purse wireframe.

Creating a new purse wireframe for EcoExT application.

Input fields for name of the purse, the currency and a colour label.

### 2.3.1.9 Labels wireframes:

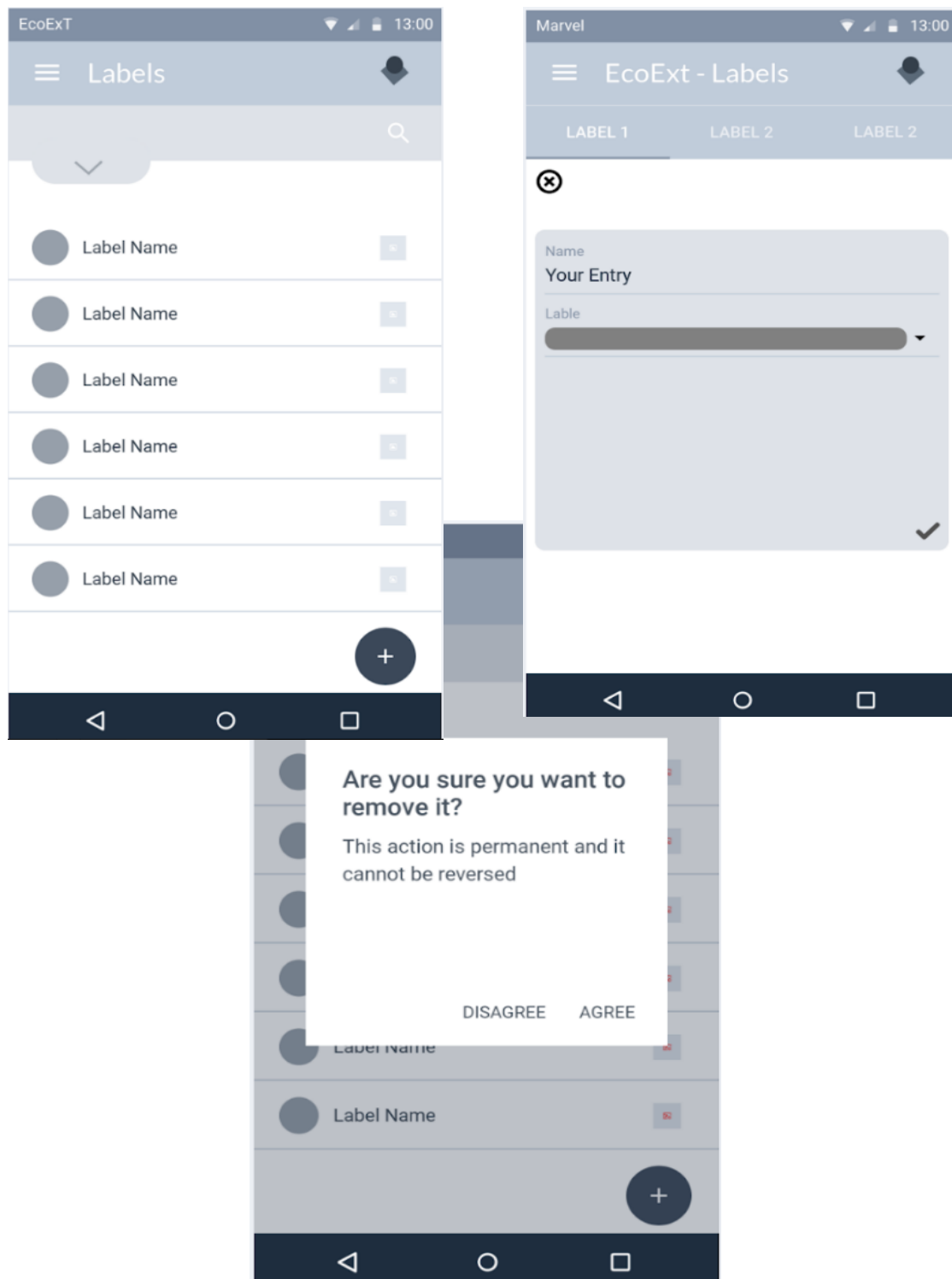


Figure 50: Labels wireframes.



Labels wireframes for EcoExT application.

Creating a new label.

Input field for label’s name and label colour.

List of created label, label’s name and label colour.

Filter search bar for labels search.

Dialog box for confirmation of label’s deletion.

**2.3.1.10 Categories wireframes:**

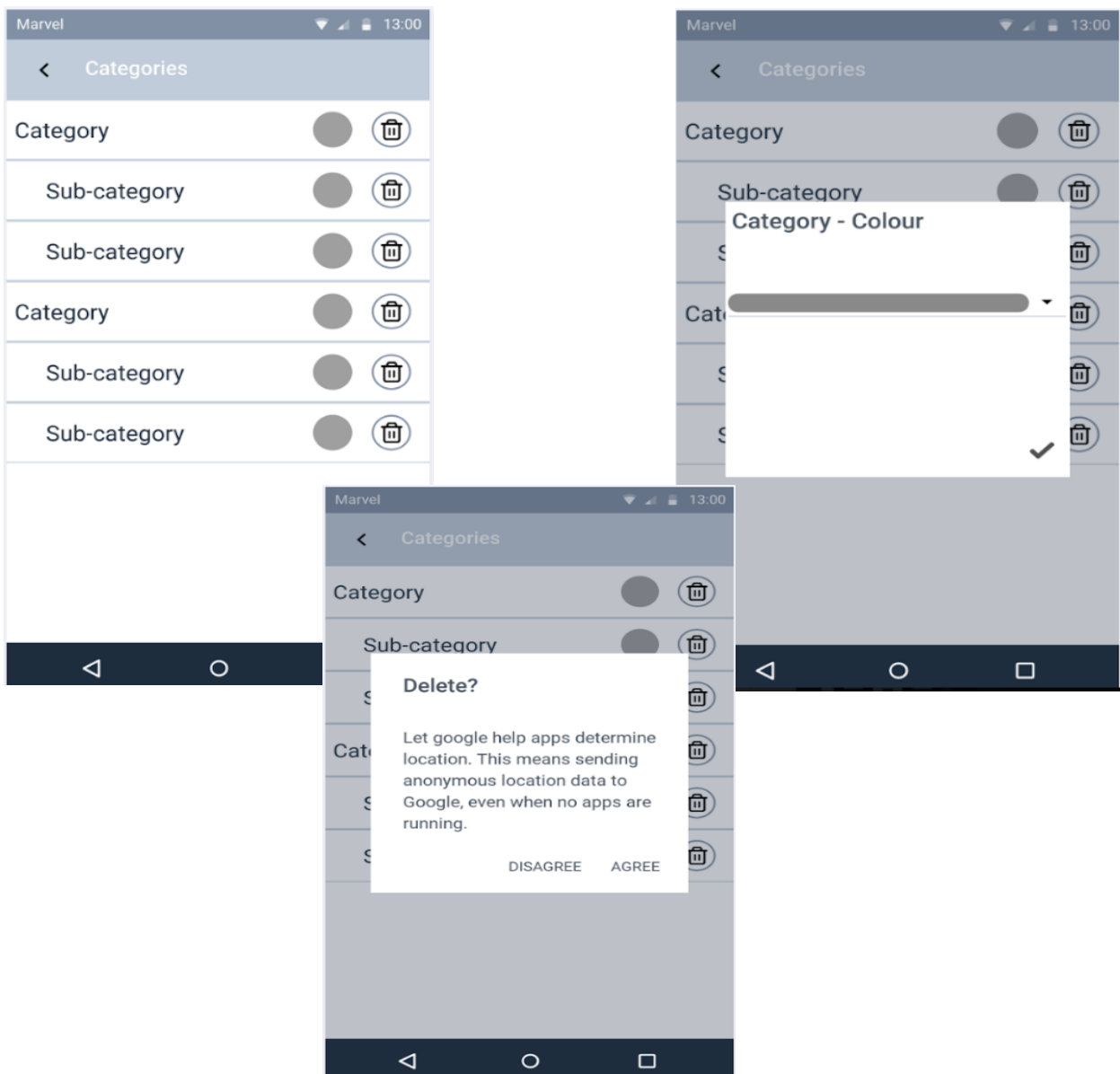


Figure 51: Categories Wireframes.

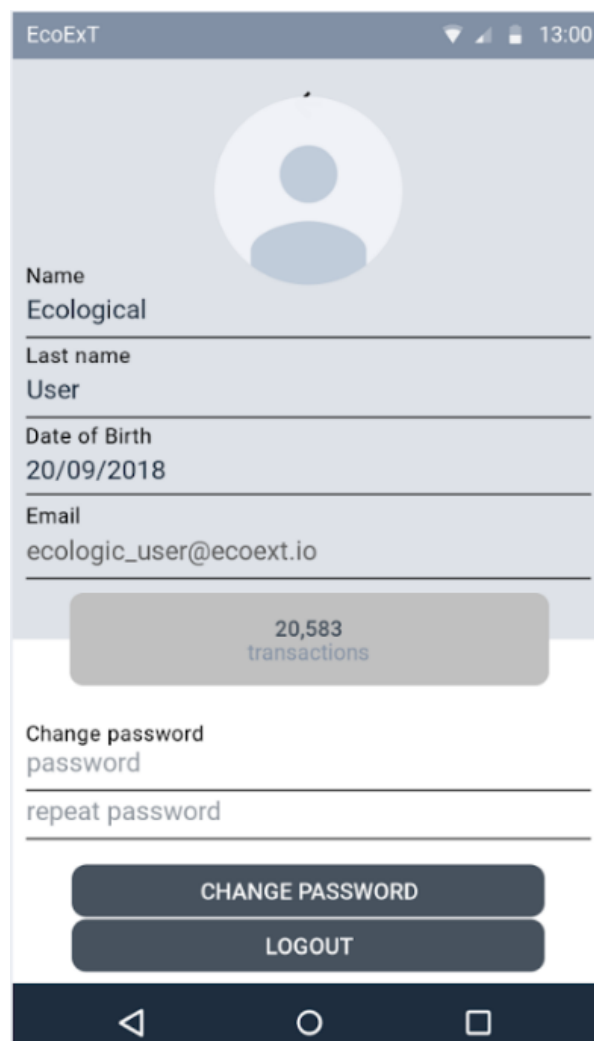
Categories wireframes for EcoExt application.

Choose category colour.

Display list of categories, category's colour and deletion option.

Dialog box for confirmation of deletion of category.

### ***2.3.1.11 Edit Profile wireframe:***



The wireframe shows a mobile application interface for editing a profile. At the top, there's a header bar with the text 'EcoExT' and a status bar showing signal, battery, and time '13:00'. Below the header is a large circular profile picture placeholder. Underneath the picture are several input fields with labels: 'Name' (containing 'Ecological'), 'Last name' (containing 'User'), 'Date of Birth' (containing '20/09/2018'), and 'Email' (containing 'ecologic\_user@ecoext.io'). Below these fields is a grey button labeled '20,583 transactions'. Further down are two more input fields labeled 'Change password' and 'repeat password'. At the bottom are two dark blue buttons labeled 'CHANGE PASSWORD' and 'LOGOUT'. The entire interface is set against a light blue background with a dark blue bottom navigation bar.

Figure 52: Edit Profile wireframe.

Edition of Profile wireframe for EcoExT application.

Input fields for name, last name, date of birth, email and password.

Change password button and repeat change password button.

Logout button.

2.3.1.12 QR code and Receipt wireframes:

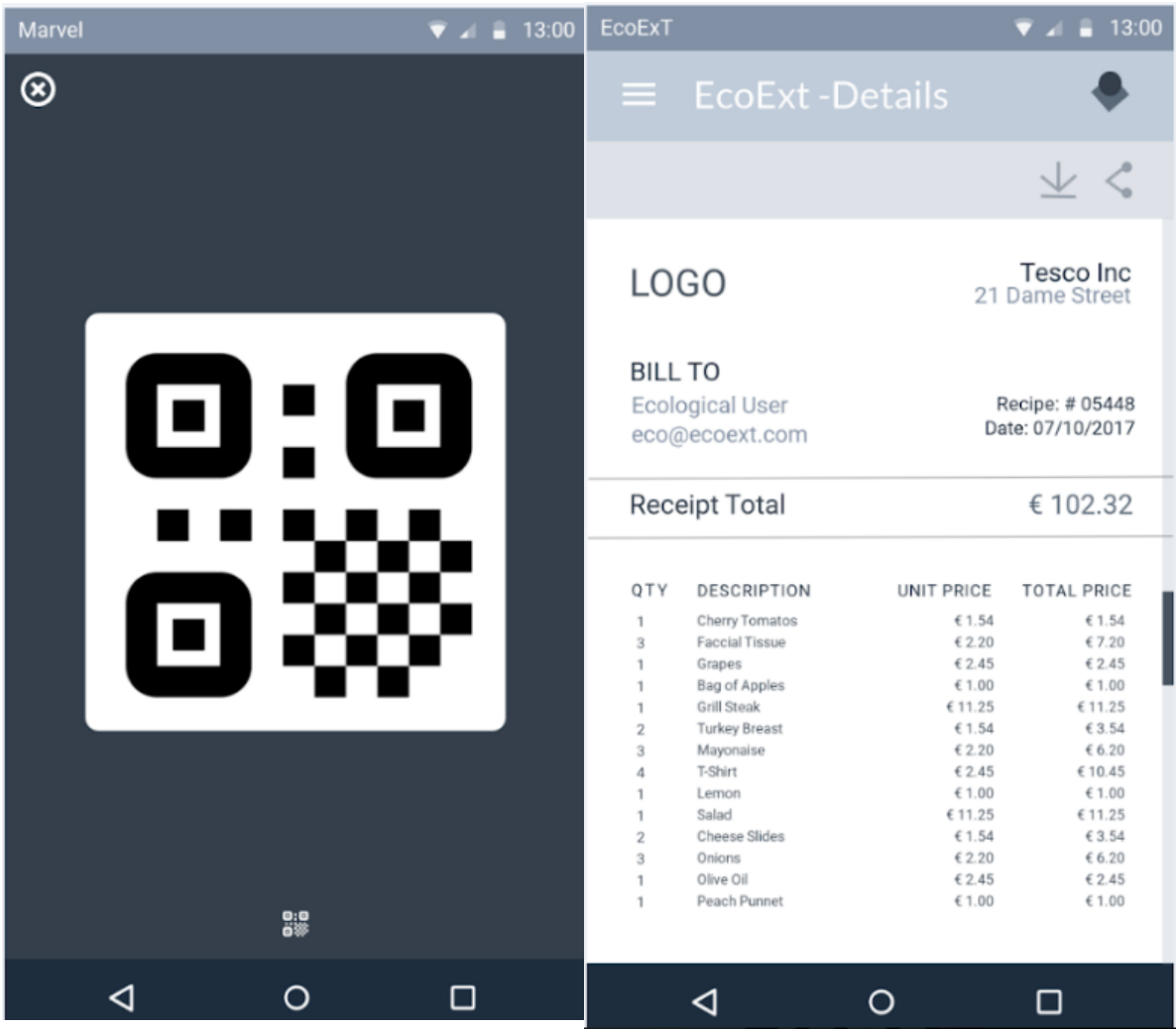


Figure 53: QR code and Receipt Wireframe.

QR Code wireframe for EcoExt application.

Displays QR code when you click Scan QR code button.

Receipt wireframe for EcoExt application.

Display receipt after the QR code has been scanned.

Receipt displays establishment logo and address, number of receipts, the total amount spent, quantity of items, description, unit price and total price of items.

## **2.5 Data design**

The sample data design is a significantly reduced and adapted analysis for EcoExT in order to explain and illustrate the requirements purpose for the first meeting with the team of EcoExT.

As expressed by the term ‘learning by doing’, the sample Data Design and database requirements aims to enhance the discussion about the development of the system as well as in respect of maintaining the necessary standards of software development. The analysis is performed assisted and accompanied, i.e., explanations and instructions are provided for the overall outcome and the better understanding of the idea. A comprehension of what will be implemented is crucially important as prerequisite for a reflection of the data design and the method’s potential when the implementation is being performed.

The database of a system is a structured set of data usually held in a computer and this data is accessible in different ways. The stored data is classified in different approaches and normally data about the system’s users is stored together with the transaction and activities they perform within that system, from a Relational Database point of view, there will be a relation between every user and their data store in the database.

Defining the database of EcoExT, the team has discussed every bit of data that was potentially be taken in every device and transactions in EcoExT System, so that the conceptual database could be defined. From this, several iterations of an Entity Relational Diagram (ER diagram) were created in order to achieve the best readability possible, then using the Seven Steps algorithm the ER diagram was converted into the Relational Model.

### **2.5.1 Database Requirements**

This section contains every database requirement for EcoExT and the following are detailed descriptions and possible situations of a user of the software.

#### ***2.5.1.1 First Scenario***

The customers that uses EcoExT mobile application are registered using an e-mail and a password. These data are going to be stored and will contain the following information:

- First name
- Last name
- Gender

- Date of birth

#### ***2.5.1.2 Second Scenario***

The customers that uses EcoExT mobile application, go into an establishment that is also registered into EcoExT Raspberry Pi system and with the first stored data information, the account related to this establishment has the following information:

- Name
- Address
- Phone Numbers
- Geo-location Coordinates

#### ***2.5.1.3 Third Scenario***

The customer that uses EcoExT mobile application, after purchasing some products in the establishment that also uses EcoExt system, have paid for their products and then will be able to access an option in their mobile application to scan a QR Code and such, will make the users to retrieve the data of their purchase. Every transaction will have the following information:

- A label
- Date
- Items
- Amount
- Description
- Types of Payments

#### ***2.5.1.4 Fourth Scenario***

Every transaction is going to generate a notification. This notification is going to have a name and a type. After generated, this notification is going to be shown into the account. Many accounts can be notified by one transaction and the account can receive many notifications.

All transactions are going to be performed by one or many accounts, and are going to be hosted by only one establishment.

Every account is going to record the data of every transaction in containers that we are calling purses. These containers are going to be described by:

- Name
- Description

Finally, the main objective and purpose of EcoExT is to design a database that follows the requirements set above.

### **2.5.2 Data Repository**

The repositories encountered on EcoExT are ensuring that the requirements are met and the data is accessible, in order to be preserved and stable the data that comes from the user is evaluated to be accepted from reliable sources and the submission process, the data may also be deposited to these repositories temporarily and stored in memory.

### **2.5.3 Storage of Data**

The application in order to access the storage of data needs to be associated with their data and needs to access the data. EcoExT is able to handle some data from flash memory to a network-area storage and are physically allocated on the network, since the data have yet not become a large storage, the application consumes little power in memory.

### **2.5.4 Entity relationship diagram**

An entity relationship diagram (ERD) with CHEN notation was created in order to show the relationships of entities that are stored in the database. Below the diagram shows the attributes that defines EcoExT properties and those entities shows the relationship between them. The illustration is the logical structure of the database.

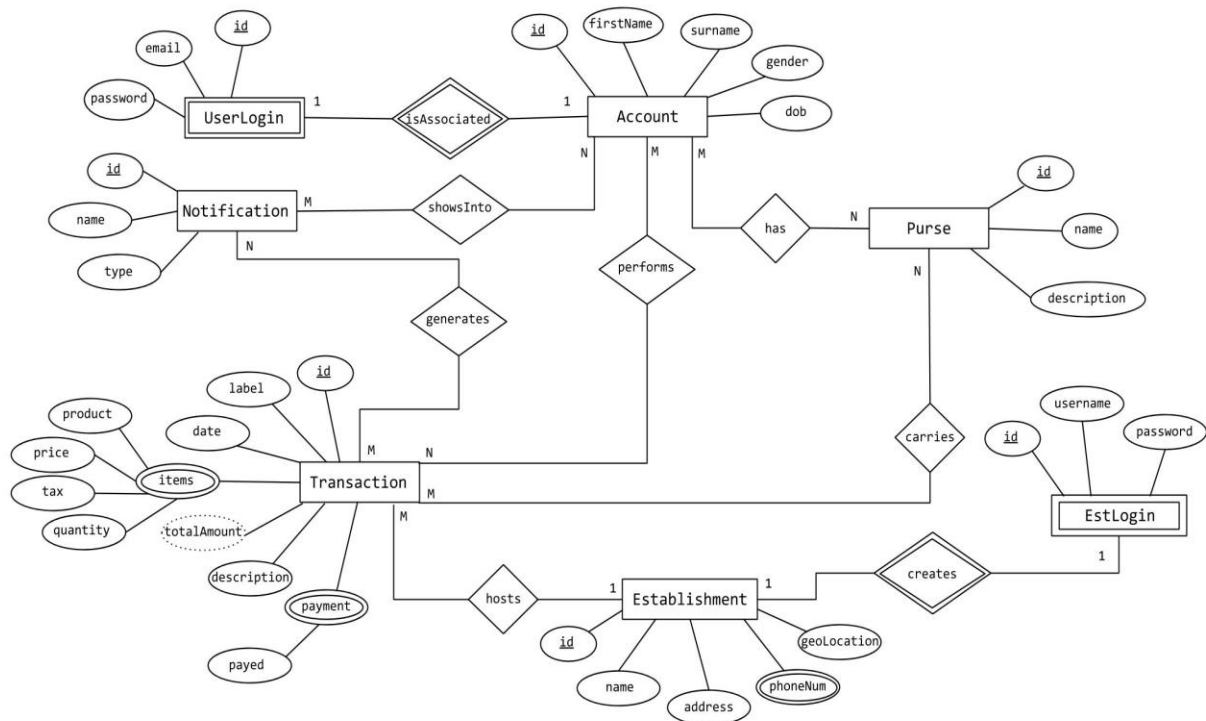


Figure 54: Entity Relation Diagram using CHEN Notation.

The figure 53 shows the ER diagram of EcoExT database requirements using the CHEN notation. Depending on how the project will be developed, the database is due to changes. The above diagram is the final version and it shows the relationships between entities, the final entities are *UserLogin*, *Notification*, *Account*, *Purse*, *Transaction*, *Establishment* and *EstLongin*.

Admitting that the final version of the diagram was created after we created two separate versions and came to a conclusion of the final design of the diagram.

The below figures show the two first versions of the Entity Relationship Diagram (ERD).

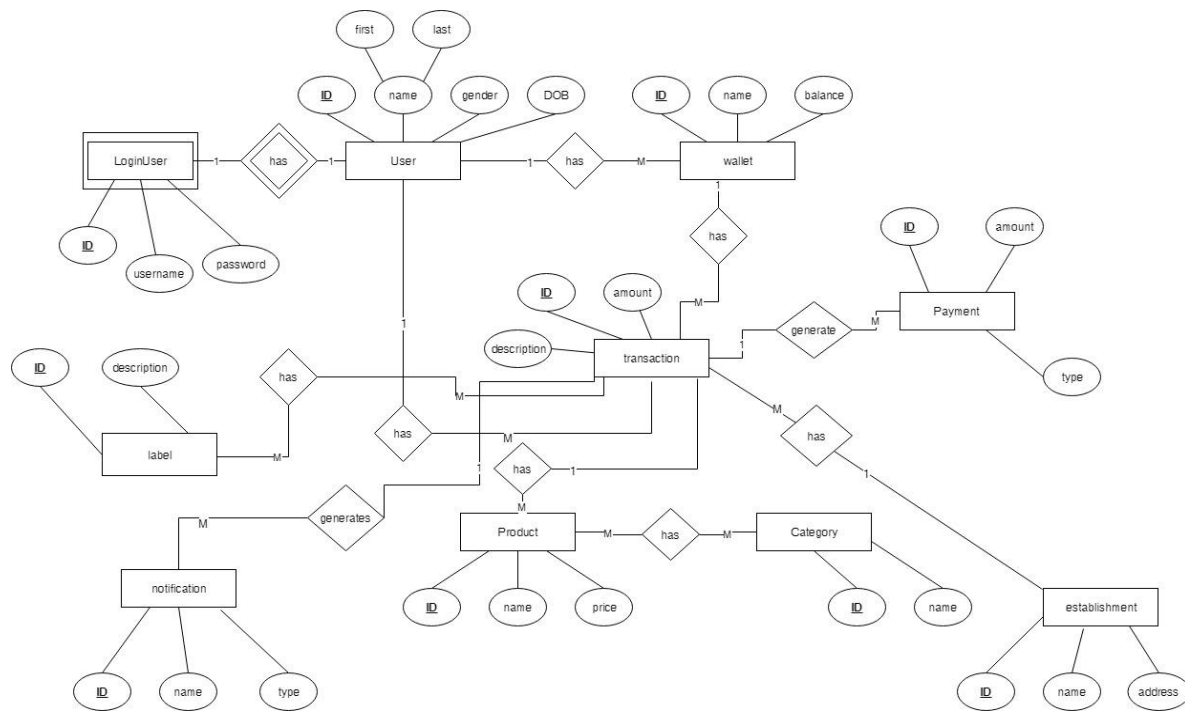


Figure 55: First version of Entity Relation Diagram.

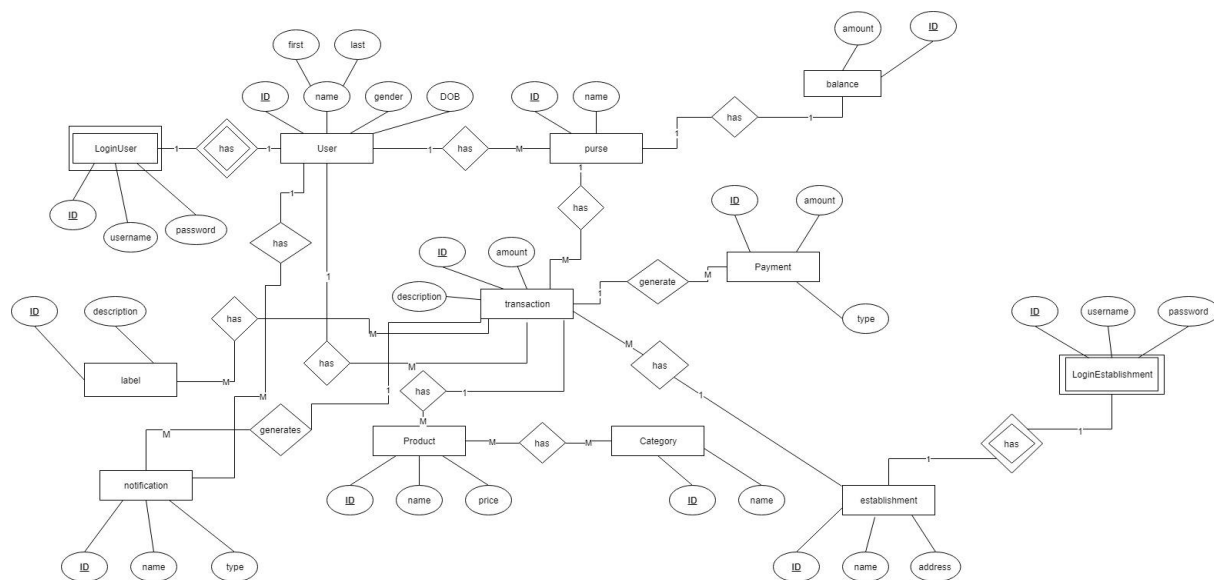


Figure 56: Second version of Entity Relation Diagram.

Pointing that both of the versions, on figures 54 and 55, the entities that were created such as, user, wallet, payment, balance, product, categories and labels, do not appear anymore in the final version since the normalization occurred and we simplified with a better solution to the final version of the diagram.



## 2.5.5 Relational Model

In order to obtain the relational model, the seven steps algorithm was used to allow the transition of the entity relationship diagram into the relational model.

### 2.5.5.1 Mapping of regular entity types

Here, the only regular entities present in the ER diagram are:

- Account
- Purse
- Transaction
- Notification
- Establishment

Now, for each regular entity a relation is created and this relation must contain all its simple attributes. The mapping of the regular entity types is shown in the figure 57.

#### Account

<u>id</u>	firstName	surname	gender	dob
-----------	-----------	---------	--------	-----

#### Purse

<u>id</u>	name	description
-----------	------	-------------

#### Transaction

<u>id</u>	label	date	description
-----------	-------	------	-------------

#### Notification

<u>id</u>	name	type
-----------	------	------

#### Establishment

<u>id</u>	name	address	geoLocation
-----------	------	---------	-------------

Figure 57: Mapped Regular Entity types.

### 2.5.5.2 Mapping of Weak Entity types

Here, the only weak entities present in the ER diagram are:

- UserLogin
- EstLogin

Now, for each weak entity type a relation is created that contains all simple attributes and include as a foreign key the primary key of the owner entity. The primary key of these relations will be the combination of their foreign key and partial key. The mapping of the weak entity types is shown in the figure 58.

**UserLogin**  

email	accountID	password
-------	-----------	----------

 Figure 58: Mapped Weak Entity Types.

<u>username</u>	<u>estID</u>	password
-----------------	--------------	----------

#### 2.5.5.3 Mapping of Binary 1:1 Relationship Types

In our case, the ER diagram does not have Binary 1:1 Relationship Types.

#### 2.5.5.4 Mapping of Binary 1:N Relationship Types

Here, we have one binary 1:N relationship type. This one is the relation that an Establishment hosts many transactions. In the figure 4.4 is shown the mapping of a binary 1:N relationship type.

**Transaction**

<u>id</u>	label	date	description	<u>estID</u>
-----------	-------	------	-------------	--------------

Figure 59: Mapped Binary 1:N Relationship Type.

Now, for each binary 1 to N relationship the foreign key has to be included in the N side of the relation and this is going to be the primary key of the 1 side of the relations. All the simple attributes of this relation are to be included in the N side relation.

#### 2.5.5.5 Mapping of Binary M:N Relationship Types

Here, the only binary M:N relationship types are:

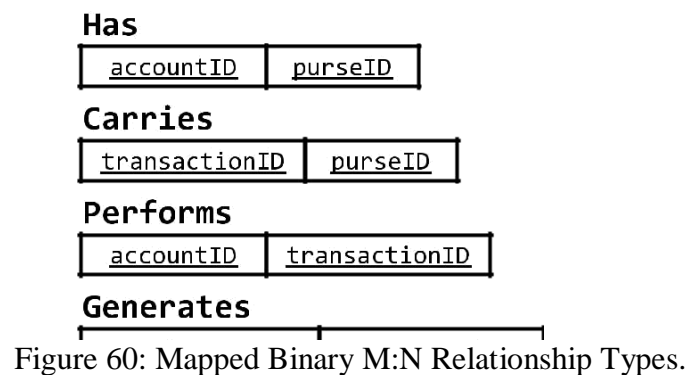
- An account has many purses and a purse is held by many accounts
- A purse carries many transactions and a transaction is carried by many purses
- An account performs many transactions and a transaction is performed by many accounts
- A transaction generates many notifications and a notification is generated by many transactions

- A notification is showed into many accounts and an account displays many notifications.

Now, for each of the above binary M:N relationship types a new creation has to be created that includes:

- The primary keys of both relations as foreign keys
- The primary key of this new relation is a composite one formed by the foreign keys
- Any simple attributes of the M:N relationship

The figure 60 shows the mapping of binary the M:N relationship types.



#### 2.5.5.6 Mapping of Multivalued Attributes

Within our ER diagram are found the following multivalued attributes:

- Establishment phone number
- Transaction Items
- Transaction Payments

Now, for each multivalued attribute a new relation has to be created. This relation will include:

- An attribute corresponding to this multivalued one
- The primary key of the relation this multivalued attribute belongs to as a foreign key
- The primary key will be the the combination of the foreign key and the attribute that represents this multivalued attribute
- If composite, include its simple components

The figure 61 shows the mapping of the multivalued attributes.



### TransactionPayments

<u>transactionid</u>	type	payed
----------------------	------	-------

Figure 61: Mapped Multivalued Attributes.

#### 2.5.5.7 Mapping of N-ary Relationship Types

In our case, the ER diagram does not have N-ary Relationship Types.

#### 2.5.5.8 Extra considerations

In the above-mentioned transactions relations, there is a foreign key that represents the establishment relationship with a transaction. But, in our design, a user can input transactions into the system that are not going to be related with any establishment. If the table in the [figure 4.4](#) stays that way then the value of its foreign key has to be filled up with a nonsensical value that indicates that it was a transaction created by a user.

But, if a table is created that relates establishments with transactions (as shown in the figure 62) and another that relates accounts with transactions (as shown in the figure 60) then there is no need to fill the transactions relation with nonsensical values when a user inputs manually a transaction.

Figure 62: Relationship Between a Transaction and an Establishment

#### 2.5.5.9 Result of the Mapping from ER Diagram to Relational Model

After applying the steps from the subsections above and the extra considerations from the subsection below and the figure 63 and 64, we obtained the relations.

### Hosts

<u>transactionID</u>	<u>estID</u>
----------------------	--------------

### Account

<u>id</u>	firstName	surname	gender	dob
-----------	-----------	---------	--------	-----

Figure 63: Result of the Mapping of Entities, Relationships and Multivalued Attributes.

<u>id</u>	label	date	description	<u>id</u>	name	description
-----------	-------	------	-------------	-----------	------	-------------

**Notification**

<u>id</u>	name	type
-----------	------	------

**Establishment**

<u>id</u>	name	address	geoLocation
-----------	------	---------	-------------

**UserLogin**

<u>email</u>	<u>accountID</u>	password
--------------	------------------	----------

**EstLogin**

<u>username</u>	<u>estID</u>	password
-----------------	--------------	----------

**Has**

<u>accountID</u>	<u>purseID</u>
------------------	----------------

**Hosts**

<u>transactionID</u>	<u>estID</u>
----------------------	--------------

**Carries**

<u>transactionID</u>	<u>purseID</u>
----------------------	----------------

**Performs**

<u>accountID</u>	<u>transactionID</u>
------------------	----------------------

**Generates**

<u>notificationID</u>	<u>transactionID</u>
-----------------------	----------------------

**ShowsInto**

<u>notificationID</u>	<u>accountID</u>
-----------------------	------------------

**EstablishmentPhoneNumber**

<u>estID</u>	<u>estPhoneNum</u>
--------------	--------------------

**TransactionItems**

<u>transactionid</u>	<u>product</u>	price	quantity	tax
----------------------	----------------	-------	----------	-----

**TransactionPayments**

<u>transactionid</u>	<u>type</u>	payed
----------------------	-------------	-------

Figure 64: Result of the Mapping of Entities, Relationships and Multivalued Attributes.

## 2.6 Database Normalization

To minimize the redundancy of the database, the normalization process occurs and in order to eliminate and reduce the redundancy in the database tables, four normal forms were used and the following

subsections will explain why EcoExT database it is in fact, complying with a fourth normal form. The Relational Model is going to be examined to see if all the tables that have multivalued attributes are in the levels of normalization.

### 2.6.1 First Normal Form (1NF)

For a relation to be in 1NF it should comply with the following rules:

1. It should only have single (atomic) valued attributes.

All the attributes of our 16 relations have atomic values, so that means that they follow this rule.

2. Values Stored on the same column are in the same domain.

This means that all the attributes of the same column should store the same type of data. In our tables it is done that for every column all the values are going to be of the same type, for this our design follows this rule.

3. All the columns on a relation have unique names.

If we examine closely the figures 63 and 64, it can be seen that for every relation all of its attributes have unique name so that confusion is avoid. Our relations follow this rule.

4. And, the order in which data is store is independent.

For all of our relations, it does not matter the order employed to store the data in every one of them, the result in going to be the same.

Because of this, our database is in first normal form.

### 2.6.2 Second Normal Form (2NF)

For a relation to be in 2NF, it should comply with the following rules:

1. It should be in the 1NF.

All of our relations are in 1NF.

2. And, it should not have partial dependency.

If the relations are examined it can be determined that all of the attributes in each relation depend on the whole key, so, no partial dependencies exist.

Now, we can say that our relations are in second normal form.

### 2.6.3 Third Normal Form (3NF)

For a relation to be in 3NF, it should comply with the following rules:

1. It is in 2NF.

All the relations are in 2NF.

2. And, it does not have transitive dependency.

This means that non-prime attributes should depend on the key and nothing but the key.

Examining the figures 63 and 64 it can be noticed that all of the non-prime attributes only depend on the key.

All of this means that our relations are in third normal form.

#### 2.6.4 Fourth Normal Form (4NF)

For a table to be in 4NF, given what is discussed in [1], the following conditions must be followed:

1. The table should be in 3NF,
2. And, the table should not have any Multi-valued Dependency.

In the figures 63 and 64 is shown the Relational Model of the database. We can see on the figure that the two entities have multivalued attributes, and which possible could have Multi-valued Dependency, are **Establishment** and **Transaction**.

If the entity **Establishment** is examined first, we can see that every establishment is completely defined by the key and nothing but the key, so, it is in the 3NF. Now, the table called **EstablishmentPhoneNumber** exists to avoid a Multi-valued Dependency between attributes that are independent form each other, such as address (geoLocation is equivalent to address) and phone numbers. Given that there is no a Multi-valued Dependency both, **Establishment** and **EstablishmentPhoneNumber**, are in 4NF.

Now, putting all attention into the **Transaction** entity, it can be seen that this one has two multivalued attributes, items and payments. In the figure 62 can be seen that the **Transaction** entity is defined by the key and nothing but the key, so, it means that it is in 3NF. Now, two extra tables are created to avoid a Multi-valued Dependency between attributes that are independent form each other, these tables are **TransactionItems** and **TransactionPayments**. Because of this, the attributes that are defined by the key are independent form each other, so the three tables, **Transaction**, **TransactionItems**, and **TransactionPayments**, have no Multi- valued Dependency, in conclusion, all the database is in Fourth Normal Form (4NF).

#### 2.7 Data dictionary for the relations

In this section, the data dictionary shows the production of each of the 16 relations. The tables from 2.6.1 to 2.6.16 shown these dictionaries.

In this section, the data dictionary shows the production of each of the 16 relations. The tables from 2.6.1 to 2.6.16 shown these dictionaries.

Attribute	Contents	Type	Format	Range	Required	Key
id	Account id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK
rstName	User First Name	VARCHAR(20)	XXXXXXXXXX...	NA	Y	
surname	User Surname	VARCHAR(20)	XXXXXXXXXX...	NA	Y	
gender	User Gender	VARCHAR(20)	XXXXXXXXXX...	NA		
dob	User Day of birth	VARCHAR(10)	XXXXXXXXXX	NA		

Table 2.6.1: Account Relation.

Attribute	Contents	Type	Format	Range	Required	Key
id	Purse id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK
name	Purse Name	VARCHAR(20)	XXXXXXXXXX...	NA	Y	
description	Purse description	VARCHAR(250)	XXXXXXXXXX...	NA		

Table 2.6.2: Purse Relation.

Attribute	Contents	Type	Format	Range	Required	Key
id	Transaction id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK
label	Transaction label	VARCHAR(20)	XXXXXXXXXX...	NA	Y	
date	Transaction date	VARCHAR(20)	XXXXXXXXXX...	NA	Y	
description	Transaction description	VARCHAR(20)	XXXXXXXXXX...	NA		

Table 2.6.3: Transaction Relation.

Attribute	Contents	Type	Format	Range	Required	Key
id	Notication id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK
name	Notication Name	VARCHAR(50)	XXXXXXXXXX...	NA	Y	
type	Notication type	VARCHAR(50)	XXXXXXXXXX...	NA	Y	

Table 2.6.4: Notication Relation.

Attribute	Contents	Type	Format	Range	Required	Key
id	Store id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK
name	Store Name	VARCHAR(50)	XXXXXXXXXX...	NA	Y	
address	Store Address	VARCHAR(100)	XXXXXXXXXX...	NA	Y	
geoLocation	Store Coordinates	VARCHAR(10)	XXXXXXXXXX...	NA	Y	

Table 2.6.5: Establishment Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
email	User email	VARCHAR(50)	XXXXXXXXXX...	NA	Y	PK	
accountID	Account ID	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	FK	Account
password	User Password	VARCHAR(200)	XXXXXXXXXX...	NA	Y		

Table 2.6.6: UserLogin Relation.



Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
username	Store Username	VARCHAR(50)	Xxxxxxxxxx...	NA	Y	PK	
estID	Store ID	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Establishment
password	Store Password	VARCHAR(200)	Xxxxxxxxxx...	NA	Y		

Table 2.6.7: EstLogin Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
accountID	Account id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Account
purseID	Purse id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Purse

Table 2.6.8: Has Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
transactionID	Transaction id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Transaction
estID	Establishment id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Establishment

Table 2.6.9: Hosts Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
transactionID	Transaction id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Transaction
purseID	Purse id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Purse

Table 2.6.10: Carries Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
accountID	Account id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Account
transactionID	Transaction id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Transaction

Table 2.6.11: Performs Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
notificationID	Notification id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Notification
transactionID	Transaction id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Transaction

Table 2.6.12: Generates Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
notificationID	Notification id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Notification
accountID	Account id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Account

Table 2.6.13: ShowsInto Relation.

Attribute	Contents	Type	Format	Range	Required	Key	Ref Table
estID	Store id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Establishment
estPhoneNum	Store Phone Number	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK	

Table 2.6.14: EstablishmentPhoneNumber Relation.

Attribute	Contents	Type	Format	Range	Req	Key	Ref Table
transactionID	Transaction id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Transaction
product	Product Name	VARCHAR(20)	Xxxxxxxxxxx...	NA	Y	PK	
price quantity	Product Product	DOUBLE INT	NNNNNN.NN BBBB	8R 2 <sup>64</sup> 0 2 <sup>32</sup>	Y		
	Quantity			2	Y		
tax	Product Tax	DOUBLE	NNNN.NN	8R 2 0 2 <sup>64</sup>	Y		

Table 2.6.15: TransactionItems Relation.

Attribute	Contents	Type	Format	Range	Req	Key	Ref Table
transactionID	Transaction id	BIG INT	BBBBBBBB	0 2 <sup>04</sup>	Y	PK, FK	Transaction
type	Type of Payment	VARCHAR(20)	Xxxxxxxxxxx...	NA	Y	PK	
payed	User Surname	DOUBLE	NNNN.NN	8R 2 0 2 <sup>64</sup>	Y		

Table 2.6.16: TransactionPayments Relation.

## 2.7 Database physical design

The sequel statements are physically produced in order to create each relation. Each of the following statements include the constraints for both primary keys and foreign keys.

- Create database statement

```
CREATE DATABASE IF NOT EXISTS ecoextv1;
```

- Use database statement

USE ecoextv1;

- Create the account relation if not exists

```
CREATE TABLE IF NOT EXISTS account (  
  id bigint(20) NOT NULL AUTO_INCREMENT, firstName  
  varchar(20) NOT NULL,  
  surname varchar(20) NOT NULL,  
  gender varchar(20) DEFAULT NULL,  
  dob varchar(10) DEFAULT NULL,  
  CONSTRAINT pk_account PRIMARY KEY (id),  
  CONSTRAINT uc_account UNIQUE (id)  
);
```

- Create the carries relation if not exists

```
CREATE TABLE IF NOT EXISTS carries (  
  transactionID bigint(20) NOT NULL  
  purseID bigint(20) NOT NULL,  
  CONSTRAINT pk_carries PRIMARY KEY (transactionID,purseID),  
  CONSTRAINT uc_carries UNIQUE (transactionID,purseID),  
  KEY FK_carries_purse (purseID),  
  KEY FK_carries_transaction (transactionID),  
  CONSTRAINT FK_carries_purse  
  FOREIGN KEY (purseID)  
  REFERENCES purse (id),  
  CONSTRAINT FK_carries_transaction  
  FOREIGN KEY (transactionID)  
  REFERENCES transactions (id)  
);
```

- Create establishment relation if not exists

```
CREATE TABLE IF NOT EXISTS establishment (  
  id bigint(20) NOT NULL AUTO_INCREMENT,  
  name varchar(50) NOT NULL,  
  address varchar(100) NOT NULL,  
  geoLocation varchar(10) NOT NULL,  
  CONSTRAINT pk_establishment PRIMARY KEY (id),  
  CONSTRAINT uc_establishment UNIQUE (id)  
);
```

- Create establishment phone number relation if not exists

```
CREATE TABLE IF NOT EXISTS establishmentphonenumber (  
  estID bigint(20) NOT NULL, estPhoneNum  
  bigint(20) NOT NULL,  
  CONSTRAINT pk_establishmentphonenumber  
  PRIMARY KEY (estID,estPhoneNum),  
  CONSTRAINT uc_establishmentphonenumber  
  UNIQUE (estID,estPhoneNum),  
  KEY FK_establishmentPhoneNumber_establishment (estID),  
  CONSTRAINT FK_establishmentPhoneNumber_establishment  
  FOREIGN KEY (estID) REFERENCES establishment (id)  
);
```

- Create estlogin relation if not exists

```

CREATE TABLE IF NOT EXISTS estlogin (
username varchar(50) NOT NULL,
estID bigint(20) NOT NULL,
password varchar(200) NOT NULL,
CONSTRAINT pk_estlogin PRIMARY KEY (username,estID),
CONSTRAINT uc_estlogin UNIQUE (username,estID),
CONSTRAINT uc_username UNIQUE (username),
CONSTRAINT uc_estID UNIQUE (estID),
KEY FK_estLogin_establishment (estID),
CONSTRAINT FK_estLogin_establishment FOREIGN KEY (estID)
REFERENCES establishment (id)
);

```

■ Create generates relation if not exists

```

CREATE TABLE IF NOT EXISTS generates (
notificationID bigint(20) NOT NULL, transactionID
bigint(20) NOT NULL,
CONSTRAINT pk_generates
PRIMARY KEY (notificationID,transactionID),
CONSTRAINT uc_generates UNIQUE (notificationID,transactionID), KEY
FK_generates_transaction (transactionID),
KEY FK_generates_notification (notificationID),
CONSTRAINT FK_generates_notification
FOREIGN KEY (notificationID)
REFERENCES notification (id),
CONSTRAINT FK_generates_transaction
FOREIGN KEY (transactionID)
REFERENCES transactions (id)
);

```

■ Create has relation if not exists

```

CREATE TABLE IF NOT EXISTS has (
accountID bigint(20) NOT NULL,
purselD bigint(20) NOT NULL,
CONSTRAINT pk_has PRIMARY KEY (accountID,purselD),
CONSTRAINT uc_has UNIQUE (accountID,purselD),
KEY FK_has_purse (purselD), KEY FK_has_account (accountID),
CONSTRAINT FK_has_account FOREIGN KEY (accountID)
REFERENCES account (id),
CONSTRAINT FK_has_purse FOREIGN KEY (purselD)
REFERENCES purse (id)
);

```

■ Create notification relation if not exists

```

CREATE TABLE IF NOT EXISTS notification (
id bigint(20) NOT NULL AUTO_INCREMENT, name
varchar(50) NOT NULL,
type varchar(50) NOT NULL,
CONSTRAINT pk_nofication PRIMARY KEY (id),
CONSTRAINT uc_nofication UNIQUE (id)
);

```

■ Create performs relation if not exists

```
CREATE TABLE IF NOT EXISTS performs (  
  accountID bigint(20) NOT NULL,  
  transactionID bigint(20) NOT NULL,  
  CONSTRAINT pk_performs  
  PRIMARY KEY (accountID,transactionID),  
  CONSTRAINT uc_performs UNIQUE (accountID,transactionID), KEY  
  FK_performs_transaction (transactionID), KEY FK_performs_account  
  (accountID),  
  CONSTRAINT FK_performs_account FOREIGN KEY (accountID)  
  REFERENCES account (id),  
  CONSTRAINT FK_performs_transaction FOREIGN KEY (transactionID)  
  REFERENCES transactions (id)  
);
```

■ Create purse relation if not exists

```
CREATE TABLE IF NOT EXISTS purse (  
  id bigint(20) NOT NULL AUTO_INCREMENT, name  
  varchar(20) NOT NULL,  
  description varchar(250) DEFAULT 'fabulous purse', CONSTRAINT  
  pk_purse PRIMARY KEY (id), CONSTRAINT uc_purse UNIQUE  
  (id)  
);
```

■ Create showsinto relation if not exists

```
CREATE TABLE IF NOT EXISTS showsinto (  
  notificationID bigint(20) NOT NULL,  
  accountID bigint(20) NOT NULL,  
  CONSTRAINT pk_showsinto PRIMARY KEY (notificationID,accountID),  
  CONSTRAINT uc_showsinto UNIQUE (notificationID,accountID),  
  KEY FK_showsInto_account (accountID),  
  KEY FK_showsInto_notification (notificationID),  
  CONSTRAINT FK_showsInto_account  
  FOREIGN KEY (accountID)  
  REFERENCES account (id),  
  CONSTRAINT FK_showsInto_notification  
  FOREIGN KEY (notificationID)  
  REFERENCES notification (id)  
);
```

■ Create transactionitems relation if not exists

```
CREATE TABLE IF NOT EXISTS transactionitems (  
  transactionID bigint(20) NOT NULL, product  
  bigint(20) NOT NULL,  
  price double NOT NULL,  
  quantity int(11) NOT NULL,  
  tax double NOT NULL,  
  CONSTRAINT pk_transactionitems  
  PRIMARY KEY (transactionID,product),  
  CONSTRAINT uc_transactionitems UNIQUE (transactionID,product), KEY  
  FK_transactionItems_transaction (transactionID),  
  CONSTRAINT FK_transactionItems_transaction  
  FOREIGN KEY (transactionID)
```

```
REFERENCES transactions (id)
);
```

■ Create transactionpayment relation if not exists

```
CREATE TABLE IF NOT EXISTS transactionpayment ( transactionID
bigint(20) NOT NULL,
type varchar(20) NOT NULL,
payed double NOT NULL,
CONSTRAINT pk_transactionpayment PRIMARY KEY (transactionID,type),
CONSTRAINT uc_transactionpayment UNIQUE (transactionID,type),
KEY FK_transactionPayment_transaction (transactionID),
CONSTRAINT FK_transactionPayment_transaction
FOREIGN KEY (transactionID)
REFERENCES transactions (id)
);
```

■ Create transactions relation if not exists

```
CREATE TABLE IF NOT EXISTS transactions (
id bigint(20) NOT NULL AUTO_INCREMENT, label
varchar(20) NOT NULL,
date bigint(20) NOT NULL,
description varchar(250) DEFAULT NULL, estID
bigint(20) NOT NULL,
CONSTRAINT pk_transactions PRIMARY KEY (id),
CONSTRAINT uc_transactions UNIQUE (id),
KEY FK_transaction_establishment (estID),
CONSTRAINT FK_transaction_establishment FOREIGN KEY (estID)
REFERENCES establishment (id)
);
```

■ Create userlogin relation if not exists

```
CREATE TABLE IF NOT EXISTS userlogin (
email varchar(50) NOT NULL, accountID
bigint(20) NOT NULL, password
varchar(200) NOT NULL,
CONSTRAINT pk_userlogin PRIMARY KEY (email,accountID),
CONSTRAINT uc_userlogin UNIQUE (email,accountID),
CONSTRAINT uc_accountID UNIQUE (accountID),
CONSTRAINT uc_email UNIQUE (email),
KEY FK_userLogin_account (accountID),
CONSTRAINT FK_userLogin_account
FOREIGN KEY (accountID)
REFERENCES account (id)
);
```

## 2.8 Database final conclusions and Database schema

As it was shown in the previous sections, designing the database and gathering requirements about how the actors in the system are going to be and what are all the bits of data that have to be store within

the database. After that a conceptual design of a database can be done by the use of an ER diagram and from this using the seven steps algorithm where we are able to obtain the logical design, also known as the relational model.

EcoExt has implemented and normalised the relational model so that we do not have redundancy, we have all the fields in its simplest form and that every attribute depends on the whole key and nothing but the key. A data dictionary is a fantastic idea in other to know what every attribute in the relations is, what they describe, the type of data and the format of it and if they are or not the key so that it is easy to go back and check vital information about the attributes within every relation.

Finally, to obtain the physical design of the Version 1 of our database, the statements created have to be written so that all the relations are created in our database. It is good to point out that depending on the needs that can be presented to us in the future this database might change, but it is always a good practice to do prototypes and versioning so if something fails, we can always go back to a stable version.

Below you can find the database schema created in order to follow all the requirements needed to the final prototype:

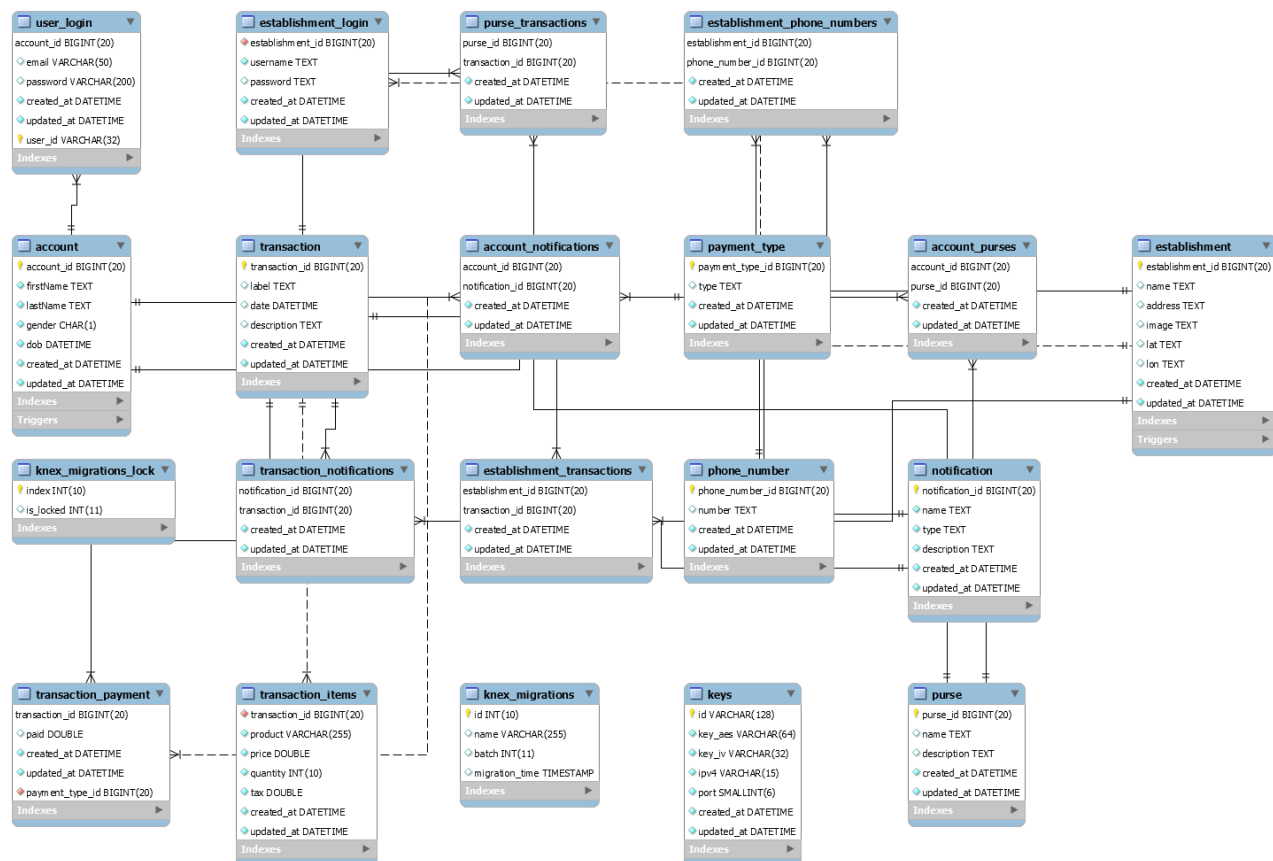


Figure 65: Database Final Schema Diagram.

## Chapter 3 - Implementation of the System

EcoExT implementation uses the structure that was created and designed during the design and analysis and the results of it in order to construct the best prototype according to the requirements developed in the system life cycle phases, which can be considered that are integrated and also aggregated to complete the final complete system.

With the aim of providing the best version of the product, the system was developed and the production involved the hardware process of forming, removing, joining and finishing, the software process of coding and testing and the implementation process required to finalise the design properties and requirements.

During the implementation process, EcoExT engineers applied their knowledge in order to meet the requirements process and produce the code, using technologies, arrangements, standards and information flows to outline the physical and logical part of the program.

### 3.1 Technologies used

Several technologies were necessary to build a reliable prototype, some of them were of extreme necessity for the development of such. They could be distributed into technologies to manage, to develop and hardware. EcoExt adapts the most modern technologies.

#### 3.1.1 Code Hosting/Versioning

A source code repository was needed for web hosting facilities where EcoExt uses a large amount of source code to develop the application software and the database. The open-source project is being hosted in [GitHub](#). This tool is used to handle the project, the different versions will be conducted using Git and this tool was especially improved in order to multi-developer the whole project.

Below you can find a screenshot of the GitHub repositories:



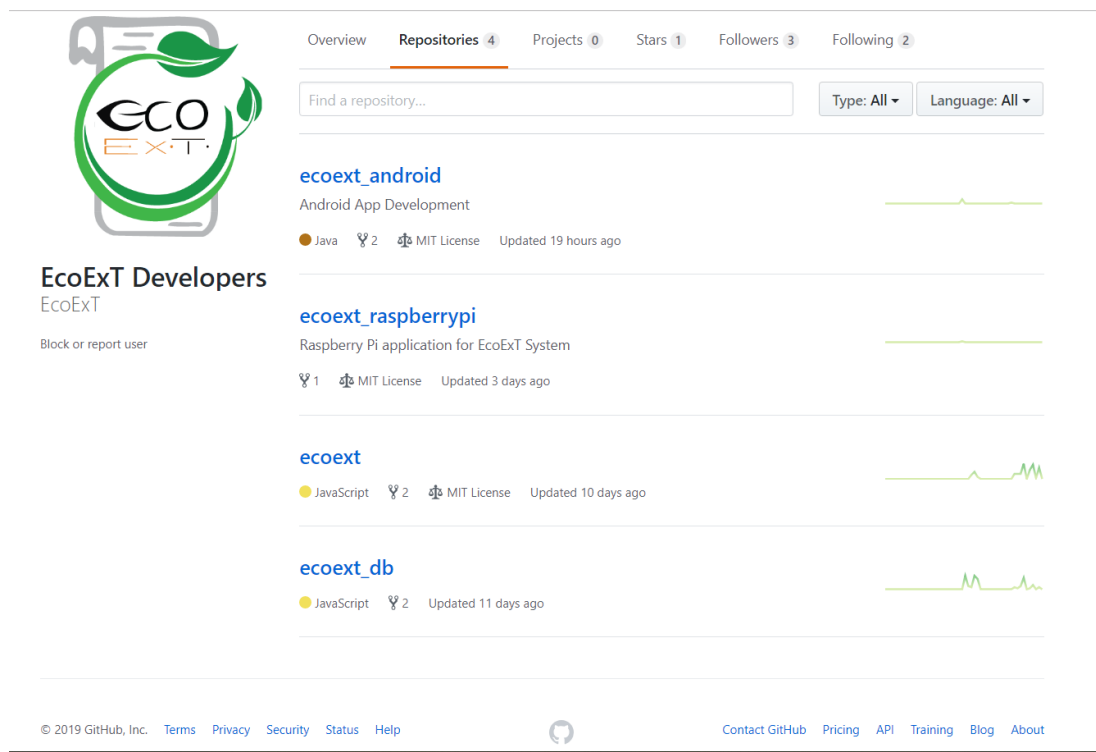


Figure 66: Screenshot of the GitHub online repositories.

Source: <https://github.com/EcoExT?tab=repositories>

### 3.1.2 Database Implementation

Database management system was very relevant to EcoExT as it manages the data in an efficiently way and allow users to easily perform multiple tasks. It also contributes preserving the data in many forms and which can be accessible anywhere, keeping records and providing ways to accomplish the tasks. As EcoExT project uses user data and company data, we decided to use GraphQL as the database tool and language.

Initially we gathered the requirements accordingly to the idea, we designed the relational diagrams and based on that, we started with the normalization of the database. With the normalized database, we started implementing by following the design of it.

We have decided to use Docker to avoid incompatibility when working on different end points and because of that, we also decided to version the database as we knew that unforeseen problem could arise at the development phase.

GraphQL was a long shot as none of us had ever worked with it, some researches and studies were conducted in order to better understand how GraphQL properly worked, as we had to join two new libraries together - Sequelize and GraphQL.

When we finally understood the process on how relations worked, we could move along the implementation phase. We also reviewed studies on how to properly implement the relational database.

We used Knex.js which is a versioning tool for databases and with that implemented, we could track backwards the changes in the design of the system and when they took place.

After having the database set up, the API was implemented over GraphQL, the process took a few weeks to finish as documentation on GraphQL is not very clear as in on how to implement it using Sequelize and MySQL.

Eventually we managed to put everything working together and started testing the API in order to test the API we found Postman and Insomnia to be very helpful at this stage as the feature that consumes the web service was not ready then, we were not able to wait to test it.

Few missing parts have been fixed at this stage, things such as missing triggers, procedures and relations that had been forgotten when implementing it.

When tests were finished, we connected the Android App to the API successfully.

### **3.1.3 Development Environment**

As technology to be used for development environment, EcoExt team members used Docker which was extremely important to the project development phase.

In a study conducted by Anand Akela about The Performance Perspective of Docker, he says: *“In simple terms, the Docker platform is all about making it easier to create, deploy and run applications by using containers. Containers let developers package up an application with all of the necessary parts, such as libraries and other elements it is dependent upon, and then ship it all out as one package. By keeping an app and associated elements within the container, developers can be sure that the apps will run on any Linux machine no matter what kind of customized settings that machine might have, or how it might differ from the machine that was used for writing and testing the code. This is helpful for developers because it makes it easier to work on the app throughout its life cycle.”*

Docker brings the perspective of adding different versions of a program and facilitating the developers so that the application is maintained with the same writing and testing code formatting.

### 3.1.4 File Storage

Every document conducted by EcoExt needed to be archived, as well as any project, there were several documents to be made. Nowadays everything is done online and we chose Google Drive as the main tool for file storage. See figure below on how we decide to manage the storage:

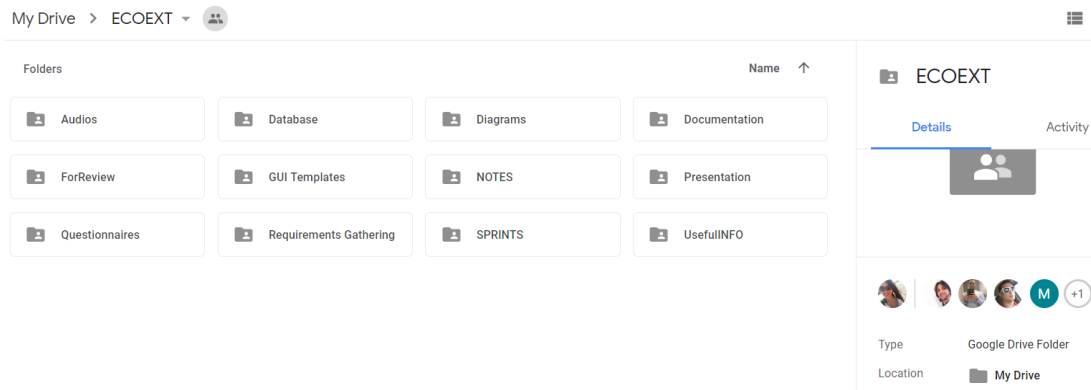


Figure 67: Google Drive in use as a tool to store files.

Source: <https://drive.google.com>

### 3.1.5 Hardware

Some technologies were required to proceed with the project progress and we have decided to buy a Raspberry PI beforehand in order to start understanding how it works as a hardware to be used. Also, smartphones with android operating system are required to start developing the application in Android.

Below is an figure of the Raspberry Pi being used as a Hardware in order to display the QR Code.

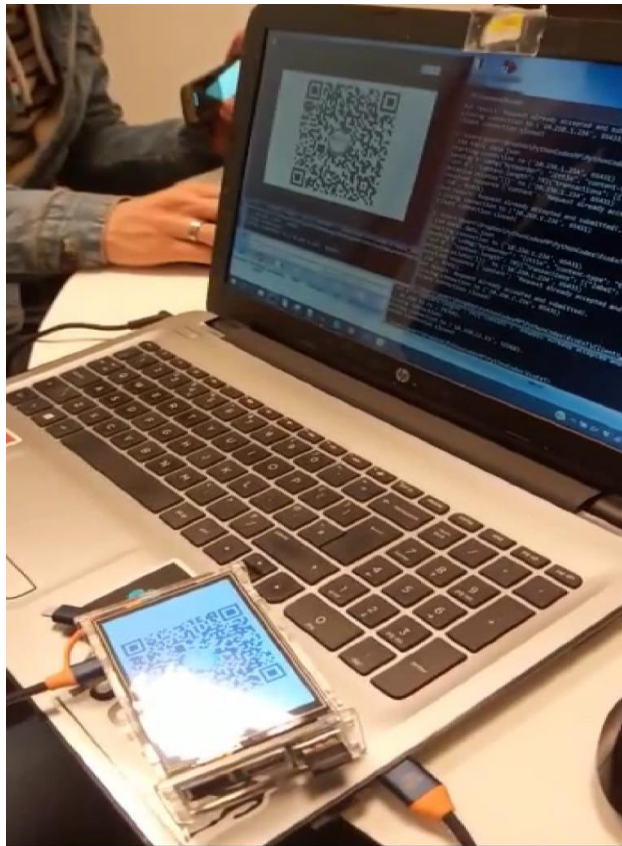


Figure 68: Hardware – Raspberry Pi

### 3.1.6 Programming

The table below represents the programming languages, frameworks and technologies that EcoExt used along the implementation, in order to develop the best approach to the project.

#### 3.1.6.1 Front-End

- Languages:

- Java
- Extensible Markup Language
- GraphQL
- JavaScript
- Python 3



---

- Integrated Development Environment (IDE):

- Android Studio
- Visual Studio Code
- Tkinter



---

- Compliant Client:

- Apollo



---

- Software as a Service (SaaS):

- Firebase
- Google
- Facebook for developers



#### 3.1.6.2 Back-End

---

- Languages:

- JavaScript
- Python
- Knex.js
- SQL
- Jason



- MySQL

- 
- Frameworks:
    - Express.js
    - Socket.io
    - Python sockets
    - Python selectors
    - Python qrcode
    - GraphQL framework
    - Sequelize ORM framework
    - Node.js
- 



---

### ***3.1.6.3 Operating systems used***

---

- Alpine Linux
  - Ubuntu Linux 18.04 LTS
  - Windows 10
- 

### **3.1.7 Task Manager**

EcoExt team proposed to manage their tasks through the hybrid methodology, in order to do that a tool was needed and as previously mentioned, Trello is being used as a task management that involves planning, tracking and reporting as well as helping individuals achieve goals and the group collaborate to accomplish collective goals.

Basecamp was also an alternative for managing tasks and it was recommended by college, although the group along with the supervisor have decided to use Trello mainly.

### **3.1.8 Visual Design**

The use of visual design was necessary in order to create the logo. The software used for it was Photoshop.

### 3.1.9 Quick Response Code (QR code)

In order to proceed with the proposed solution for EcoExt being fully completed. The way of scanning the receipts when doing a purchase and getting it on your smartphone, was needed a QR code (quick response code) which is a type of two-dimensional bar code that is used to provide easy access to information through a smartphone.

The users of EcoExt along with their smartphones will be able to point the camera, where it will work along with a barcode reader, to a QR code and generate a mobile tagging process. The QR codes will be generated by the Raspberry Pi and will look like the figure below:



Figure 69: Prototype of EcoExt QR Code

EcoExt QR Codes are going to be essentially a token that represents the transaction identifications. This is done in order to centralize any request to the server.

## 3.2 Developing Raspberry Pi Application

The actual coding of the Raspberry Pi application is going to take place. In order to show what was done class diagrams are going to be implemented instead of showing code here. For full code check the link [https://github.com/EcoExt/eoext\\_raspberrypi.git](https://github.com/EcoExt/eoext_raspberrypi.git), where all the code that was written when developing the application what versioned in the **develop branch**.

### 3.2.1 Listener side of the Raspberry Pi application

The main functionality of the application is being able to listen from incoming connection from a PoS System for transactions, connect to the API, to respond the PoS System regarding the status of the

transaction sent, and to receive a request from the API asking to tell the user that the QR Code was scanned successfully.

First, a class that creates accepts any incoming connection coming into a particular port, when listening in a particular ipv4 address, has to be created. In the figure 1.13 can be seen said class. Here can be seen that the class has multiple elements to receive any incoming connections, such as:

- Selector to handle multiple connections
- A socket to listen to incoming connections
- References to view and controller to handle UI events
- Methods that allows the application to:
  - Get Selector and events, and set listener socket.
  - To accept incoming connections and wrap them up
  - To start monitoring the socket for incoming connections

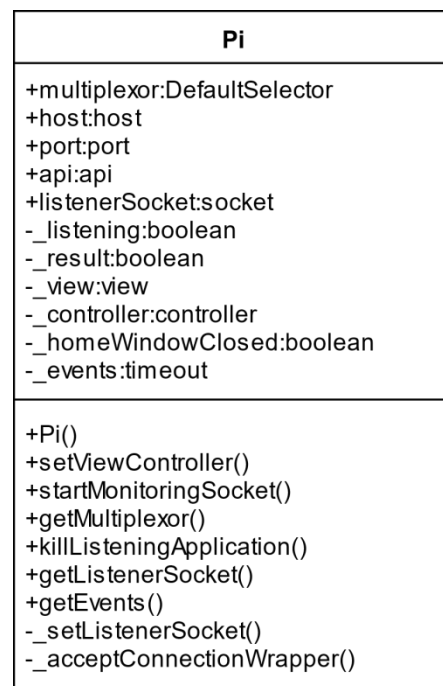


Figure 70: Pi Class UML Representation.

But now, it is needed to process the data that is coming through the socket. For this, it is known that the Pi Class has a PiMessage, so this means that an instance of this class is created within the Pi



Class. The figure 1.14 show the relation between these two classes. This figures also show that this class is related as well to the APIConnection Class, because there is where a connection with the GraphQL API is establish to store the transactions. The response of the API to the request of storing a transaction should be a token, when is done successfully, that represents the transaction, and this token is used by the EcoExT QR Code Generator Class to create the QR Code to be scanned. All this can be seen in the figure below:

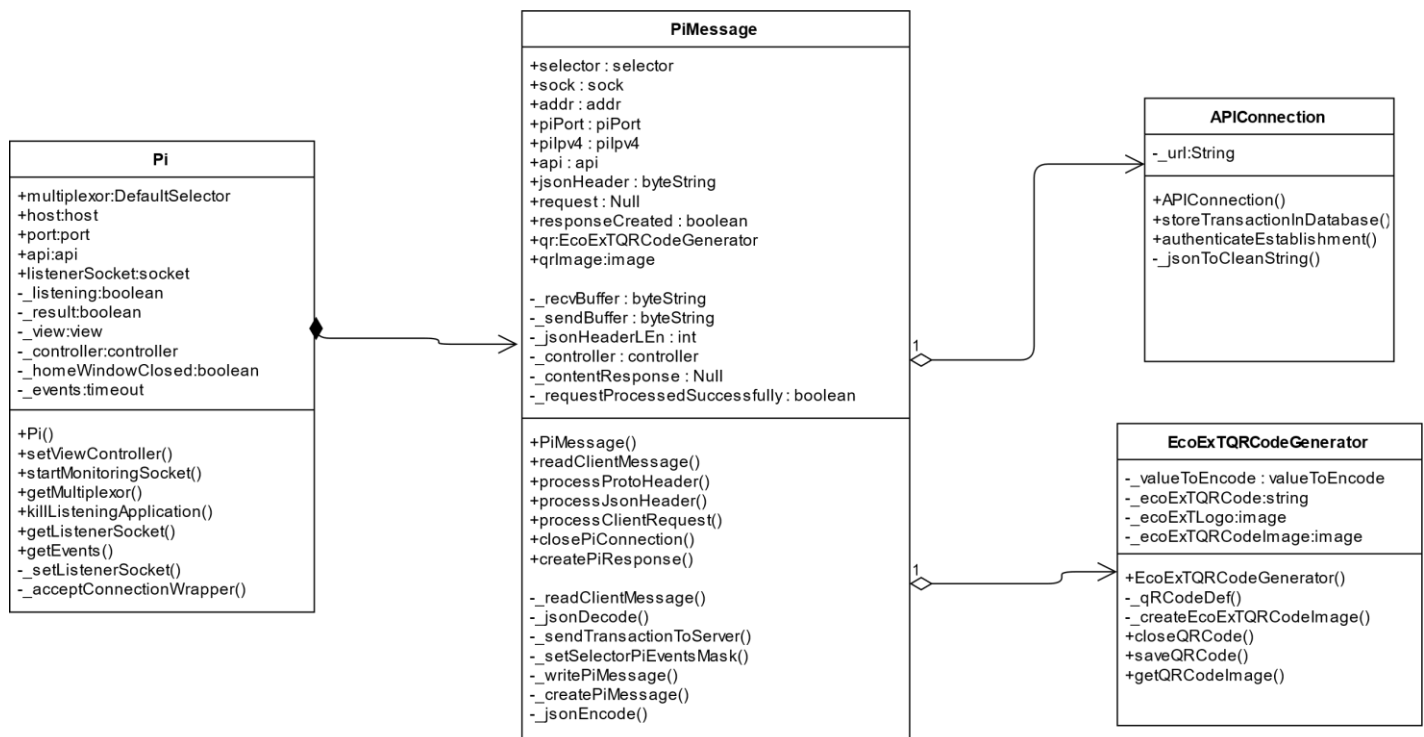


Figure 71: Class Diagram of the Listening Side of the Application

In the PiMessage Class can be seen that the application is handling the requests and generating responses, in the APIConnection Class that connections to the unique endpoint of the API is being done, and from the EcoExTQRCodeGenerator Class that from the token received a QR Code is generated. Now, in the following section let us see how GUI is implemented.

### 3.2.2 GUI side of the Raspberry Pi application

What has been developed and implemented in the above section will achieve a listening application for connections without showing anything to the user, the application encounters with the need of a user interface that can show on the following:

- An idle window waiting for connections,
- The Token that represents the transaction as a QR Code showing a progress bar underneath that tells the user the amount of time they have to scan the code,
- A window that tells the user that a QR Code was scanned successfully by the user within the window of time defined,
- A window that tells the user that they did not scan the code in the window of time determined.

In the class diagram of the figure 1.15 can be seen how the different windows of the UI side of the application are interlaced between each other. The WindowsCreator Class is the one in charge of creating the windows are going to be shown to the users. The idle window will be HomeWindow that is a RootWindow and a Tk Object as well. The QRCodeWindow is the one in charge of showing the token as a QR Code and the progress bar to tell the users the time they have left to scan said code. Objects instantiated with this class is a TopLevelWindow and a TopLevel Object. The ScannedDoneWindow and NotScannedDoneWindow both are a TopLevelWindow and a TopLevel Object, and this classes are show to the users when a QR Code was scanned and when it was not within the window of time selected.

All these classes take general classes that were created to be used in any window just by changing a few parameters. These utility classes are FrameFigure, CanvasFigure and FigureLoader.

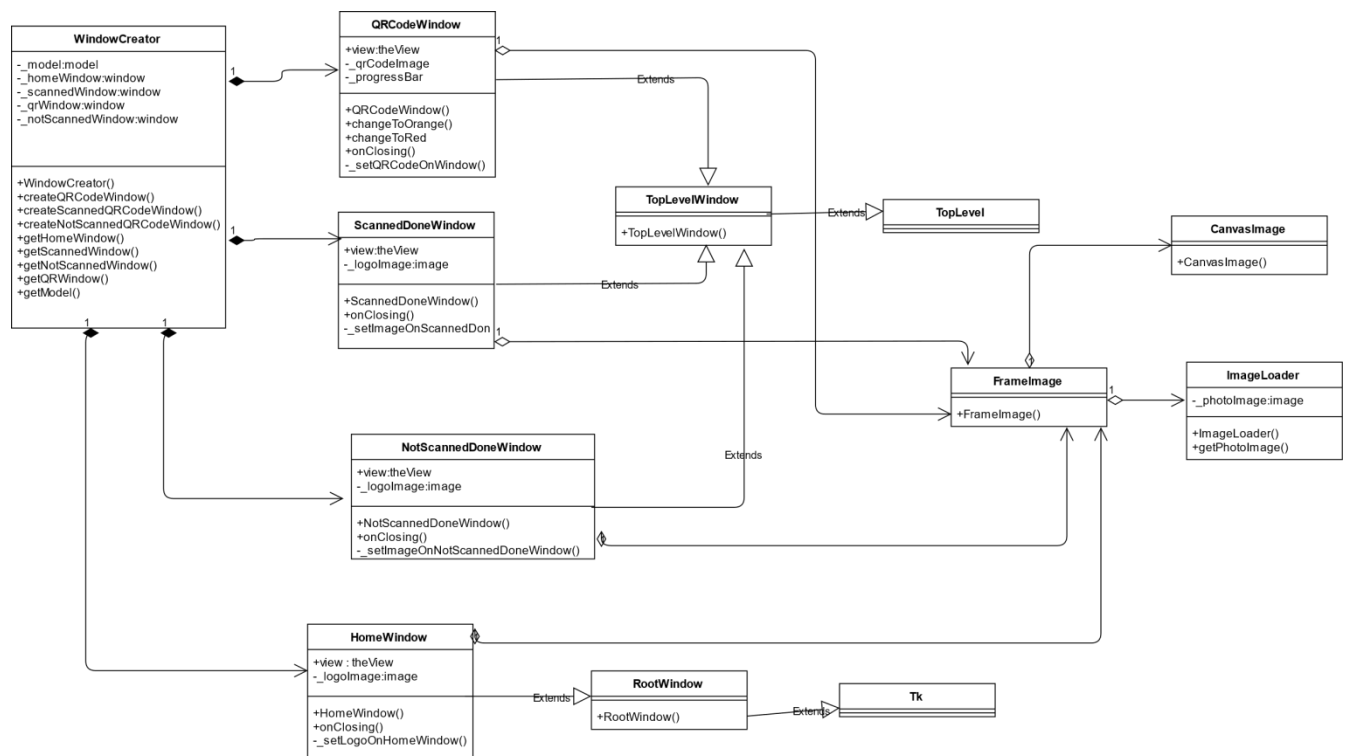


Figure 72: Class Diagram of the GUI Side of the Application.

### 3.2.3 Controlling both sides of the Raspberry Pi application

With the model and the view developed, it is time to create the controller of the Raspberry Pi Application. This controller is going to make the GUI side of the application move between windows depending on the data received in the listening side. The figure below it shows the class diagram of the controller of the Raspberry Pi application.

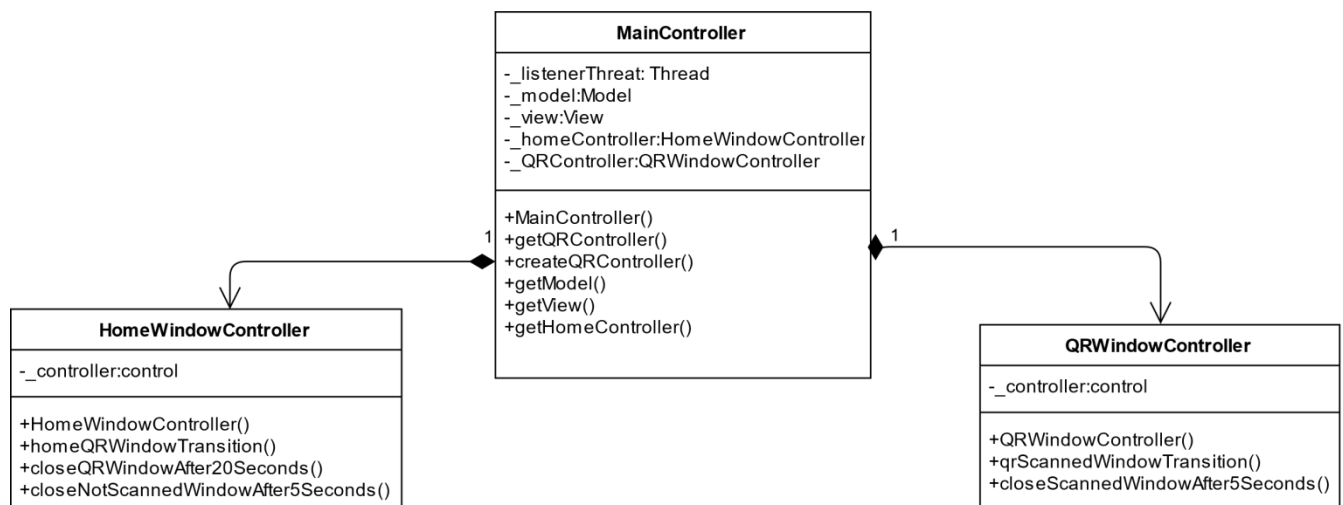


Figure 73: Class Diagram of the GUI Side of the Application



In the MainController Class is where the two threads, the listening and the GUI, are run for the application to start working. In the other two classes present there is where the transitions between windows are going to be handled.

The HomeWindowController is where the following transitions are going to be handled:

- When receiving a token move from the Home to the QR Code Window.
- After 20 seconds of the QR Code being shown, if it was not scanned show the Not Scanned window.
- If the Not Scanned window is being shown, after 5 seconds go back to the Home window.

And, the QRWindowController is where the following transitions are going to be handled:

- If QR code is scanned within the 20 seconds window then the API is going to send a confirmation message to the Pi App, this will allow the transition from the QR Code to the Scanned window.
- After 5 seconds the Scanned window will transition to the Home window.

And every time the application is in the Home window is going to wait for incoming connections from the PoS System, if a connection comes up from another parties the application will not do anything.

### **3.2.4 Running the Raspberry Pi application**

In the figure 72 above, we can see the class diagram of the Raspberry Pi application. In there, we can see that the classes that represent the MVC model of the application are MainPi, MainController and WindowCreator, where all of the objects that are interacting in the application are instantiated. The class MainApp is the one that create an instance of every party in the MVC model implemented in the application and the Main Class is the one with the main method.

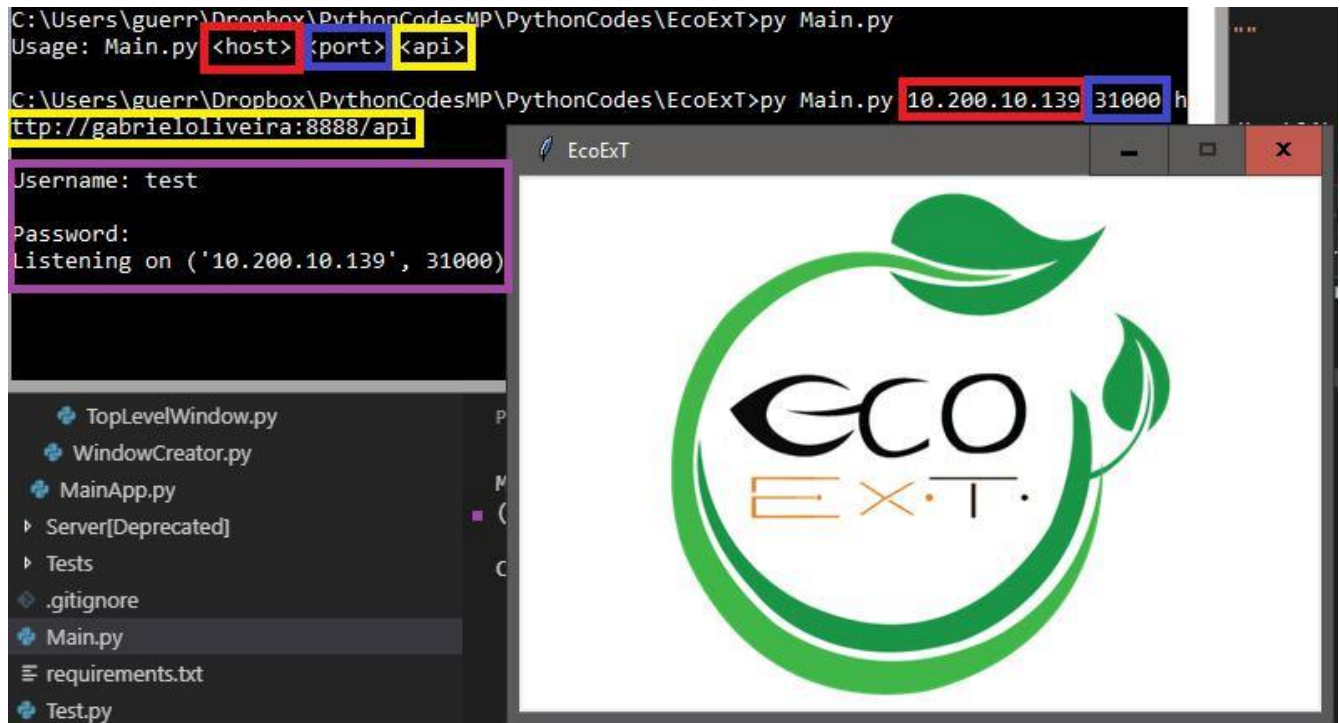


Figure 75: Running the Pi Application.

It is important to point out that the way the application runs is asking for command line parameters that are going to be accessed using the **sys module** built in Python 3. Figure 73 above shows how the application is run when it does get the parameters and when they are given by the person that is starting the application. There can be seen that host ip4 address, host port, and API ip4 address has to be given to run the application.

Then, an establishment login was implemented, so if login details are successfully the application will start running. All these is shown in the figure 73 above. After all this procedure the application is going to be up and running for the tests and for future improvements.

### 3.3 Problems encountered

As any system, the development process can face some problems and this was not different with EcoExT, below we explain the problems we faced and how we overcame them.

### **3.3.1 Impeditive factor of database model**

When developing the DB Diagram and discussing with the group we realized that few tables needed to be added in order to resolve problems with the redundancy and efficiency as well as resource to get to the user transactions in a better way. This table was the userAccount and transactionAccount which connected together with every transaction and allows segregating the data and categorizing in a more comprehensible way.

When generating the receipts in the EcoExT app the user should also be able to see the details about the establishment, because of that, we decided that the best approach for this situation was to add an establishment table to the system, as well as a LoginEstablishment which will contain the login details. The possibility of putting this information with the LoginUser table was also discussed and argued between the developers, however that would cause problems in terms of primary and foreign keys when relating the tables together. So then after analysing and testing possible errors, the final proposal is as show in the diagram on figure 53.

### **3.3.2 Impeditive factors of android application development**

During the development process of the Records layout, pulling information from the database was an issue since there were initially decisions made to be taken in terms of performance against best implementation of the code.

An initial way of approaching the issue was querying the database every time the records page (Layout) was requested, this in terms of performance and resources consuming was not a good option. Instead we decided make a query to the database as soon as the application loads the home page, store the data in local memory in the phone and from there, passing reference of the data into the different pages and activities in the application.

As java developer, the first intuition to do so is by sending lists of objects as reference in the constructor of the java class. However, android studio recommends not to do so since this could cause problems with the application, resources could be cleared when this is sent to the background and another application request more and more memory (Gunhan Sancar/Mobile Application Developer, 2015). As a solution Android implementation suggests the use of bundles in order to pass the object between activities and fragments.

As mentioned above, this would be the best implementation recommended by android guidelines, however, in order to pass custom objects in a bundle, this object needed to implement the Serializable interface and implement its methods, thus the object class required to be modified. Since we are using the Apollo Library for interaction with GraphQL API, our objects in the application are being automatically generated by Apollo and a single object contain a multiple

nested list of objects inside, then in order to achieve that, we needed to either modify the object that have being manipulated by Apollo or create some type of redundancy by creating new custom classes (objects) that could be manipulated by us to implement the necessary interfaces.

The mentioned implementation could result on a time issue and since the time was a problem concern for the development process, we have decided to come with a solution that would give a solution to the issue without compromising the app and resources.

The first version of the application does not need to consume many resources and could run perfectly without being affected when sending to the background to execute tasks, the reason being is that there are no tasks to be executed in the background so as a solution we have decided to keep passing the information through instantiation of constructor, using an empty constructor as second call that will allow the process to send info back without the whole resource itself, and the implementation of the **@SuppressWarnings** (“Valid Fragment”) as shown in the figure 68 below, the annotation to take out any error given, allowing the application to run perfectly.

```
@SuppressWarnings("ValidFragment")
public RecordsFragment(ArrayList<GetAllUserTransactionsOrderByDateQuery.UserTransaction> userTransactions) {
    this.userTransactions = userTransactions;
}
```

Figure 76: @SuppressWarnings (“Valid Fragment”) annotation

It is important to notice that the second version of the application will implement the recommended way through the use of bundle so the performance can be fully working in case the application requires it. Machine learning is intended to be as one of the features of this second version of the application which will require lot of background activity from the application itself.

### 3.3.3 Problems encountered on database implementation

#### 3.3.3.1 Docker

An environment had to be chosen, although the people involved in the project have different computer configurations therefore having a stable development environment was a challenge. By searching online on ways to achieve this, Docker was mentioned on almost every single post found online and Docker can provide containers, which will run based on an image, therefore the same



figure could be run in any machine having always the very same internal configuration allowing all members to have no concerns about incompatibilities.

Understanding Docker generally does not happen overnight as its internals are still unknowns when you first run it, configuring Docker volumes, figures, por mapping and having proper containers running our applications had shown to be a challenge.

Some training with the group member was made necessary and a couple of hours have been invested into it, however it has shown us that we made a good decision.

#### ***3.3.3.2 API – GraphQL***

For the API we decided to go with GraphQL as it is something new, developed by a giant company which improves performance for APIs. It sorts the problem of making too many requests to a remote server and having to connect everything on the client side (Android App).

GraphQL is relatively new, there is proper documentation on it, but additional libraries had to be used in order to put everything to work together, about three (3) weeks were invested on learning how to use it.

It was hard to make relations to work properly as poor documentation was found on how to implement it using MySQL and the ORM library of choice - Sequelize.js

#### ***3.3.3.3 Database - Design***

MySQL is the RDBMS for our project as it is a well stablished database system, we have had no problems at all with MySQL, although we did have problems designing the relations. Even though we normalized the database, the gathering was not as good as it could have been, therefore few adaptations had to be done in order to achieve the ideal design.

When developing the API some relations were made not necessary and by having database versioning it was not complicated to get it properly implemented as all it was needed is a new version of the relations.

#### ***3.3.3.4 Security***

Security is a major problem when it comes to data.

We have no SSL server to host our API, therefore our connections are completely vulnerable. We have done some encryption on server side to avoid parts of our data from being seen by possible attacker when transmitting data in between the application and the server. Considering although that there is a lot more to be done on that matter but we decided to leave it out as time was critical.

#### ***3.3.3.4 Architecture considerations***

- Using Docker should avoid conflicts with platforms and versions.
- Need for SSL on our API

### **3.3.4 Problems encountered when developing the Raspberry Pi application**

#### ***3.3.4.1 Running the GUI thread alongside the listening thread.***

Here, given the way the framework Tkinter is built, is very hard for a thread running in parallel to communicate to it. To overcome this problem, it was used an indirect form of the Observer Design pattern, given that, by the time this was implemented our knowledge regarding that design pattern was almost null.

#### ***3.3.4.2 Sending a confirmation message from the API when a QR Code is scanned by a user using the Mobile App.***

A Python code that sends this message was built, but, running sub-processes in a different language that of the API was not going to be very efficient, performance wise. The solution was to translate this code using basic knowledge in JavaScript and studying deeply the Socket.io library from Node.js.

#### ***3.3.4.3 The format of the data send from the API was not compatible with Python byte string.***

This problem was overcome by using the Node.js library called Python-struct. This allows us to create the message in a format that python could interpret so that the API could tell the Pi App that the QR was created successfully.

#### ***3.3.4.4 Creating the Token that represents the transaction in the API.***

This was built in Python 3, so given the performance of running a Sub-process in a language that was not the API was built on was going to give performance problems. The solution was to translate the code from Python to JavaScript, using the equivalent libraries for hashing, encryption, decryption, encoding, and decoding, as well as method created by the team so that the result was one that resembled the one obtained with Python 3.

## Chapter 4 - Testing and Evaluation

### 4.1 Test plans and objective

EcoExT application version 1 is ready to be tested and the requirements functions such as, registration, login, adding a new purse, adding a transaction, scanning QR code and so on, are going to be tested and evaluated.

The objective of the testing is to access the whole system in order to find if the requirements criteria has been done, to find any errors that the system might have, to assure quality of the product that has been developed.

The functionality and the performance of the application is also going to be tested and if there are any bugs, the developers will fix the issues or set it for a future release.

The user interface and user experience are going to be tested as well as the background performance and logic of the system.

### 4.2 Unit and exploratory testing

The purpose of this testing is to validate that each unit of the software performs as designed. An exploratory testing aims to validate different scenarios in order to check if the requirements designed are meeting the criteria.

### 4.3 Scenarios to be tested

#### 4.3.1 Scenario 1:

As a user, I want to register on EcoExT so that I am able to use the application.

#### Result of testing:

- ✓ Me as a user, I am able to create an account on EcoExT if I agree with the Terms and Conditions and the Privacy Policy regarding to the consent for usage of my personal data. Quality assurance is passed.
- ✗ I am only able to register and login with my Google account. Users should be able to register with another e-mail.

- ✗ The terms and conditions have not been applied.

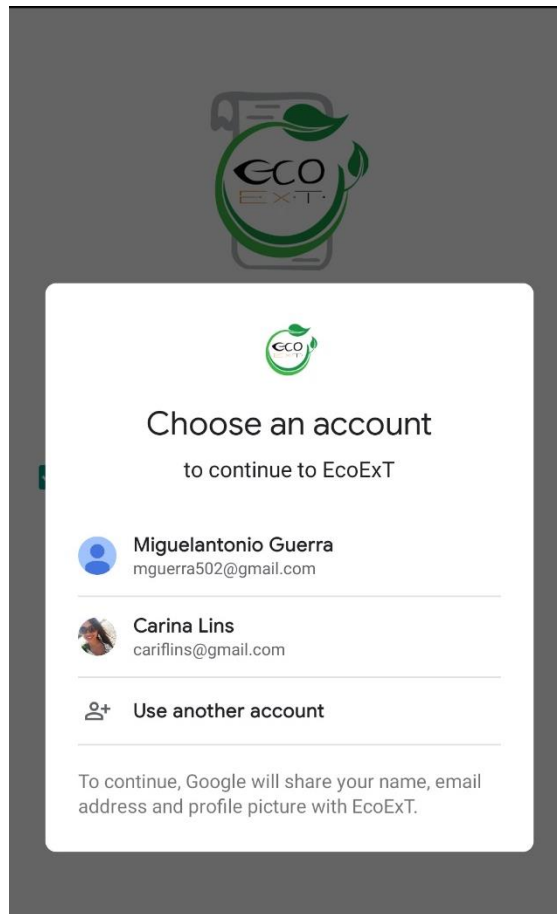


Figure 77: Screenshot of registering with Google account.

#### 4.3.2 Scenario 2:

As a user, I want to login on EcoExT so that I am able to manage my account.

##### Result of testing:

- ✓ Me as a user, I can input my Google login details such as e-mail and password and login into the application.
- ✗ Users should be able to login with another e-mail.

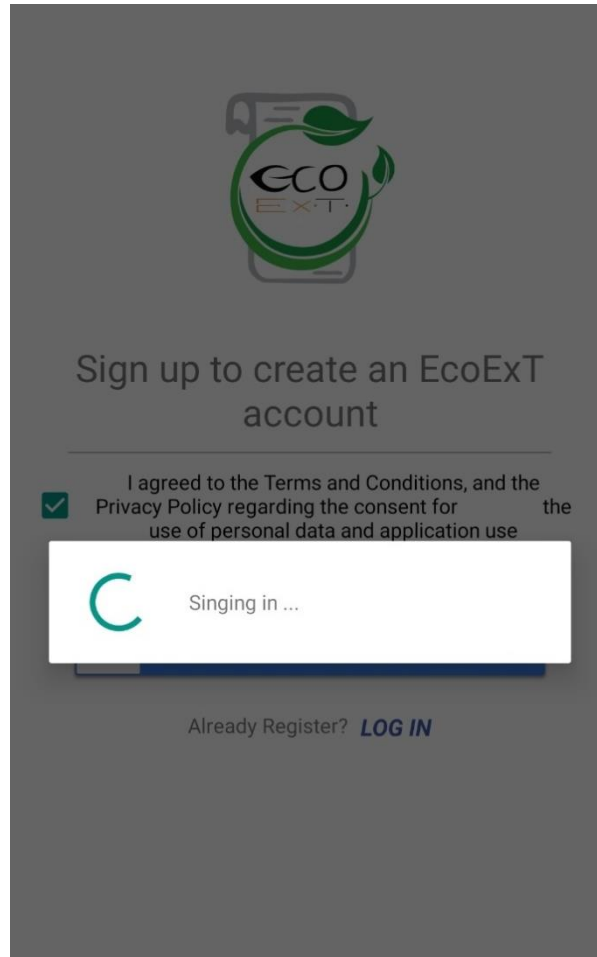


Figure 78: Screenshot of logging in to EcoExT application.

#### 4.3.3 Scenario 3:

As a user, I want to add a new purse, so that I am able to manage the amount of money that I have.

##### Result of testing:

- ✓ Me as a user, I can set up a new purse by adding a purse name, a description and an initial amount.
- ✓ If I don't add a description or an initial amount, I can still create the purse.
- ✓ I can't create the purse without a purse name.
- ✓ There are no restrictions for name and descriptions.

- ✓ I can add as many purses as I want.
  - ✗ Name of creating purse should be longer than 15 characters.
  - ✗ Decimal value when adding an amount should have a limit of two decimal values after the dot, inputting the amount and also showing the amount.
  - ✗ When hitting the button of creating a purse more than once, we are able to create many purses at the same time.
  - ✗ If the initial amount is not set, the application crashes.
  - ✗ Not able to edit the value of a purse.

#### Fixed issues:

- ✓ The name of purse is longer than 15 characters now.
- ✓ Creating purse button only allows one purse to be created.
- ✓ Application does not crash if the initial amount is not set.

#### Future releases:

- Decimal value to be set for two decimals after the dot on the amount value.
- Edit purse and value amount.

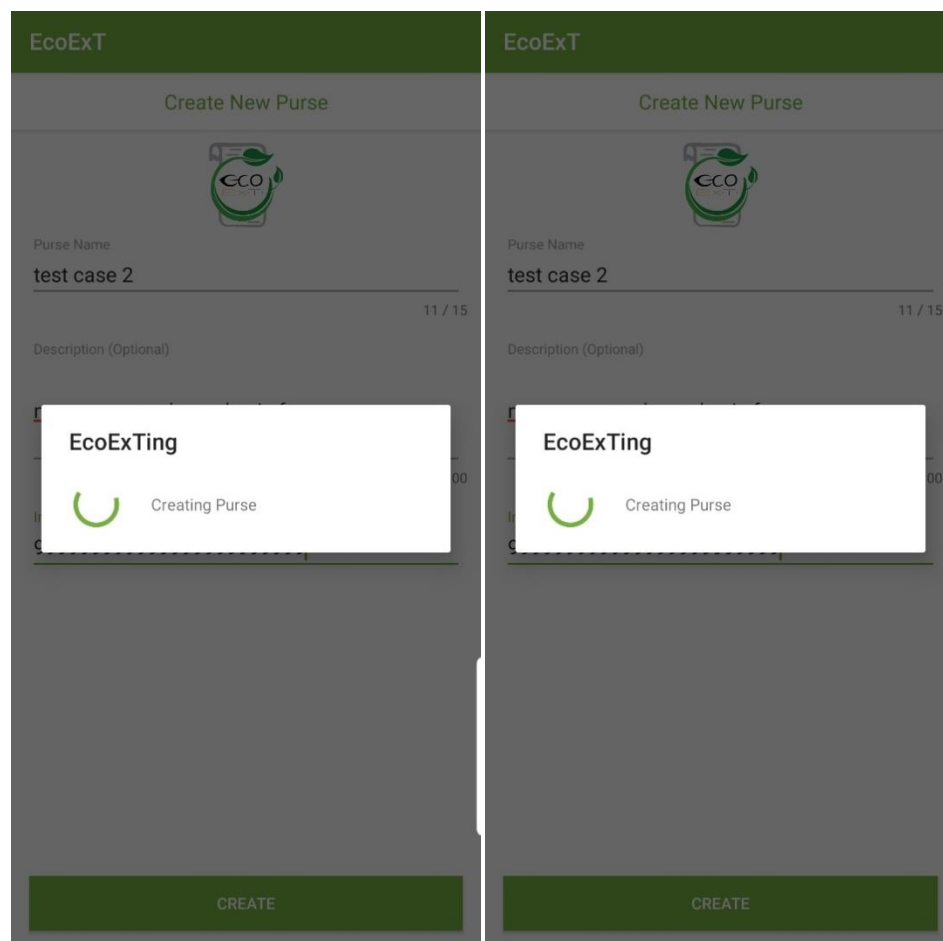


Figure 79: Screenshot of creating and adding purse into EcoExT application.

#### 4.3.4 Scenario 4:

As a user, I want to delete my purse, so that I am able to manage different purses and the values previously added within the purse are also deleted.

##### Result of testing:

- ✓ Me as a user, I can choose to delete the purse and also there is a validation box that pops out to make sure I want to delete the purse.
- ✓ I can delete purses and I am also able to delete all purses.

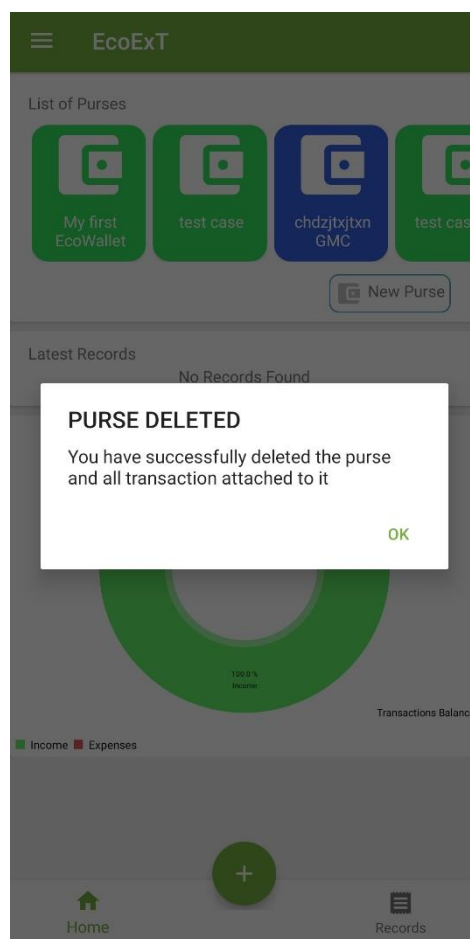


Figure 80: Screenshot of deleting purse on EcoExT application.



#### 4.3.5 Scenario 4:

As a user, I am able to view the information contained within the purse, so that I am able to categorize my expenses and have a better control of it.

##### Result of testing:

- ✓ Me as a user, I am able to access different purses and see the name, description and value of them.
- ✓ I am also able to have a list of records with the amount of records and the date I have created the transaction.
- ✓ If the transaction is an income, the value of the purse adds up.
- ✓ If the transaction is an expense, the value of the purse is deducted.

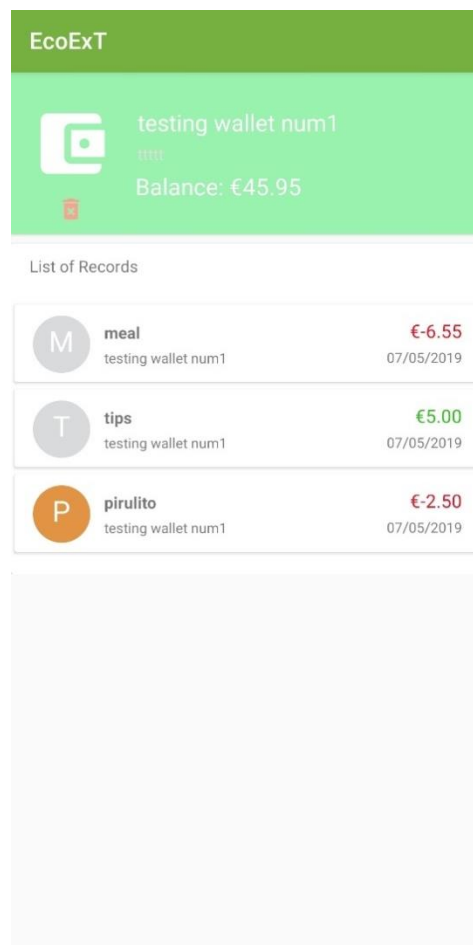


Figure 81: Screenshot of purse details on EcoExT application

#### 4.3.6 Scenario 6:

As a user, I want to manually add a new transaction/create a new record and categorize it in the purse, so that I am able manually manage my records.

##### Result of testing:

- ✓ Me as a user, I am not able to add a new transaction/record without selecting a purse.
- ✓ I am able to add as many transactions as I can. There are no limits for it.
- ✓ I am able to see the date of the transaction when it has been added, although I am not able to change the date for another day. (not a blocker, future implementation).
  - ✗ The transaction name should be more than 15 characters.

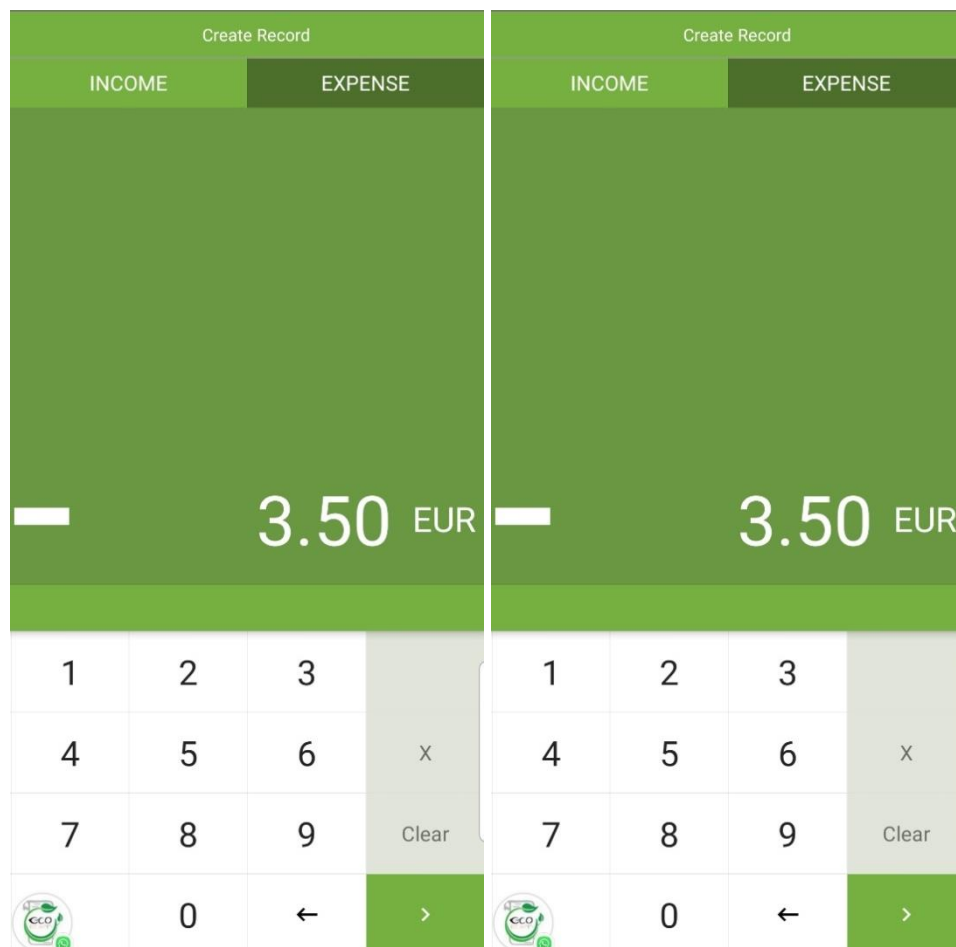


Figure 82: Screenshots of adding a transaction, such as income or expense on EcoExT application.

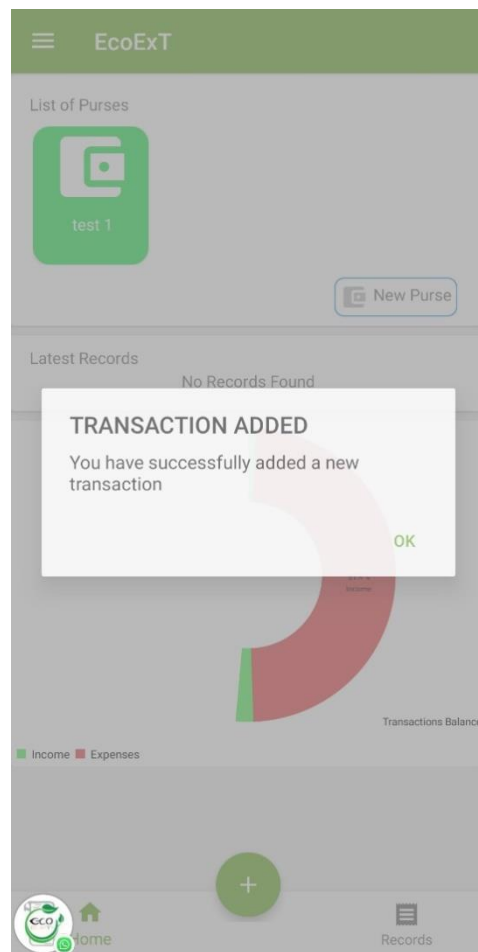


Figure 83: Screenshot of transaction added on EcoExT application.

#### 4.3.7 Scenario 7:

As a user, I want to be able to view and filter my transactions, so that I can have a better control of my expenses.

##### Result of testing:

- ✓ I can access my records tab, filter by the date of the transaction.
- ✓ I am able to see a list of the transactions when It is filtered by date.
- ✓ I can filter by the purse, although from a product point of view, it is redundant as I am able to access my purses and view the transactions on it already.
- ✗ No transactions found missing.
- ✗ Latest records are disappearing when I delete one record, I have to change pages and go back to home page again to show the latest records.

##### Future releases:

- Removing the filter of purse filtering for the first version and applying it on the next version with another UI/UX design.

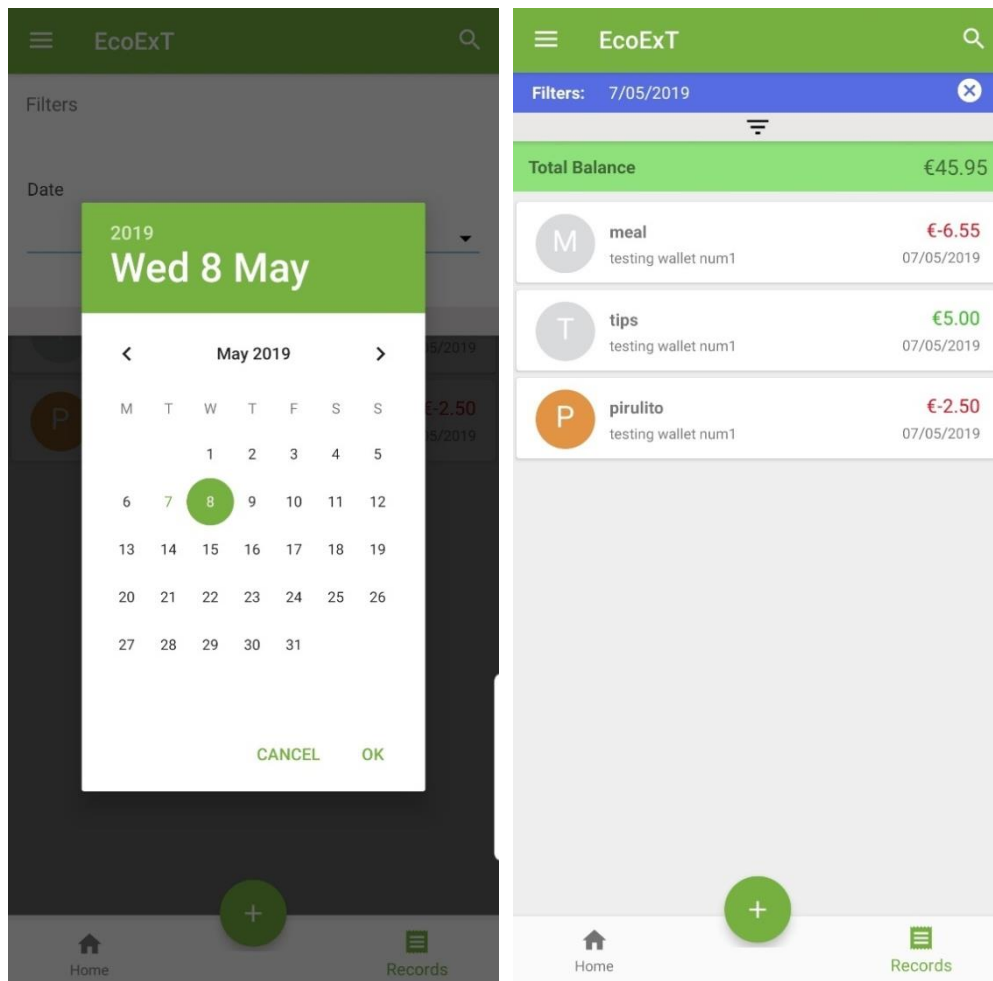


Figure 84: Screenshots of filtering transactions by the date on EcoExT application.

#### 4.3.8 Scenario 8:

As a user, I want to delete the transaction that have been manually created, so that I can control the transactions.

##### Result of testing:

- ✓ Me as a user I am able to delete transactions and also to confirm on the confirmation box that I want to delete the transaction.

- ✓ The amount of money in the purse is updated to the previous without that transaction.

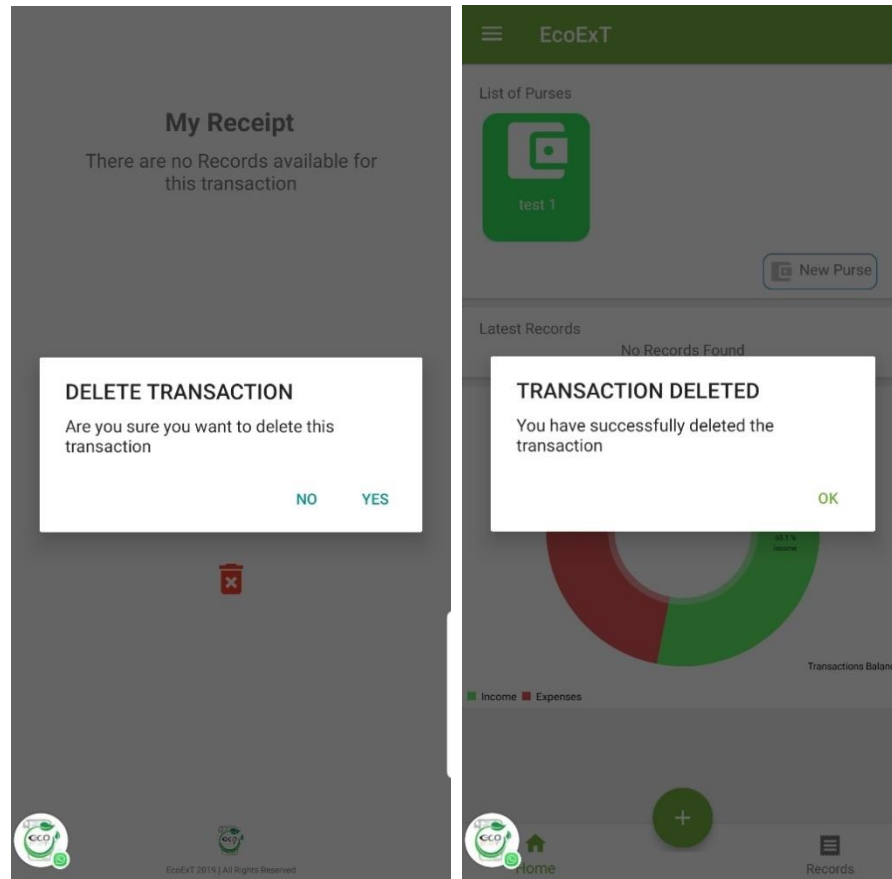


Figure 85: Screenshots of deleting the transaction on EcoExT application

#### 4.3.9 Scenario 9:

As a user, I want to be able to see my expenses' statistics, so that I can get a different perspective on how my money is being spent.

##### Result of testing:

- ✓ Me as a user, I can see the graph statistics with the incomes and expenses.
- ✓ If I delete or add a new transaction, the amount of the money on the graph is changed.
- ✓ I am able to see the percentage of incomes and expenses.



Figure 86: Screenshots of expenses' statistics graph on EcoExT application

#### 4.3.10 Scenario 10:

As a user, I want to be able to access my camera and scan a QR Code so that I am able to retrieve a digital receipt of my transaction.

##### Result of testing:

- ✓ Me as a user I am able to give access to the camera with the application installed at the first time.
- ✓ I am able to scan the QR code generated by EcoExT Raspberry Pi.

- ✓ When I scan the QR Code I retrieve the digital receipt from the establishment, and I am able to see the quantity, items, units and prices of the transactions.
- ✓ I am able to see the total amount of the receipt.
- ✗ When I scan the receipt, it adds automatically to the purse and I am not able to choose which purse it goes to.

Fixed issues:

- ✓ I am able to choose which purse the transactions are going to be added to.

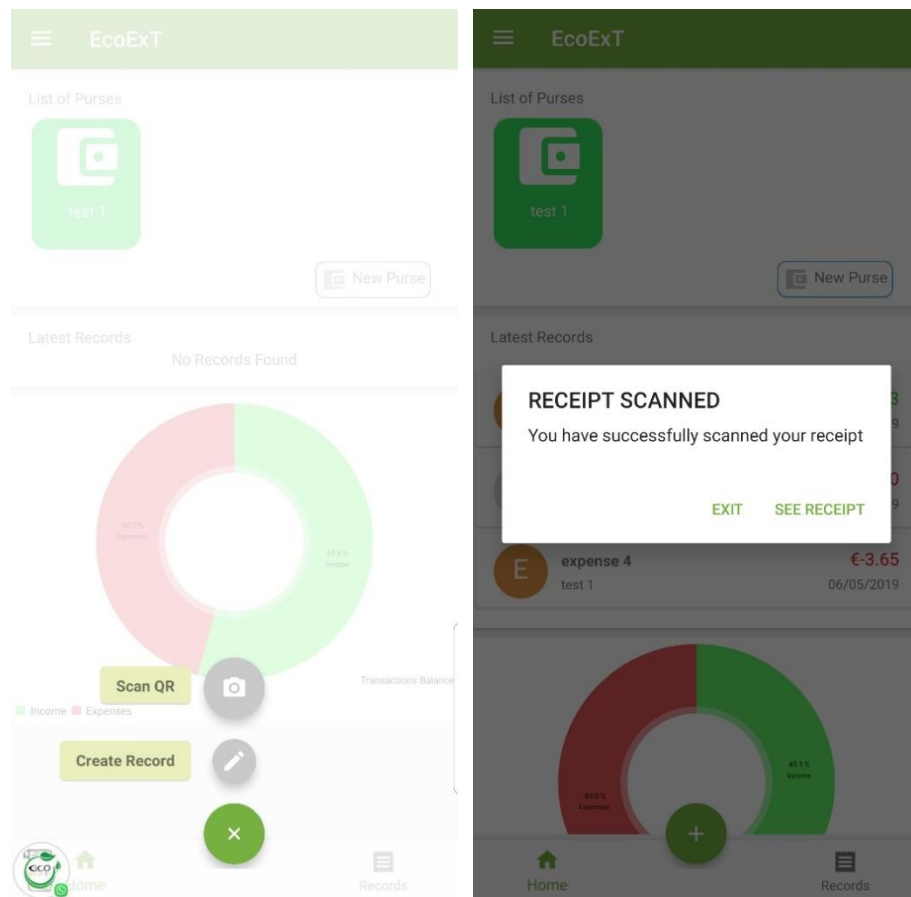


Figure 87: Screenshots of Scan QR Code and receipt scanned on EcoExT application.

EcoExT			
<div> <div>S</div> <div> <b>Establishment Name</b>  Establishment Address  Establishment Phone </div> </div>			
€13367.07			
Carina Lins cariflins@gmail.com		Receipt #00125 06/05/2019	
Qty.	Item	Unit	Total
1	stir-fry		€325.93
7	seeds	€611.69	€4281.83
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15
2	fork	€198.46	€396.92
5	oil	€110.83	€554.15

Figure 88: Screenshot of the Receipt after the QR Code is scanned

#### 4.3.11 Scenario 11:

As a user, I want to logout of the application, so that I am not logged in.

##### Result of testing:

- ✓ I am able to logout of the application.
- ✗ I don't have a dialog box to confirm the logging out of the application.

##### Fixed issue:

- ✓ The dialog box is set in order to confirm the logout of the application.



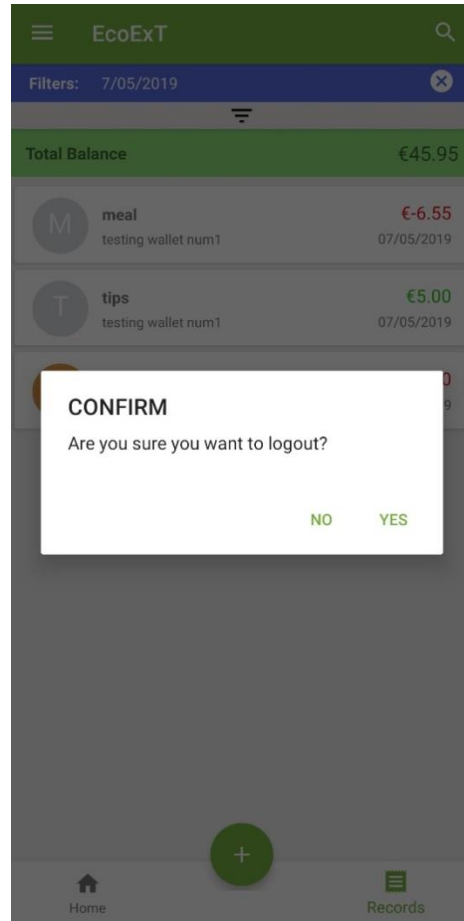


Figure 89: Screenshot of the application logout

## 4.4 Scenarios for Raspberry Pi testing

### 4.4.1 Scenario 1:

As an Establishment, I want to be able to display the QR Code from a transaction, so that the consumer can have access to a digital receipt.

#### Result of testing:

- ✓ The establishment is able to display the QR Code on the Raspberry Pi.

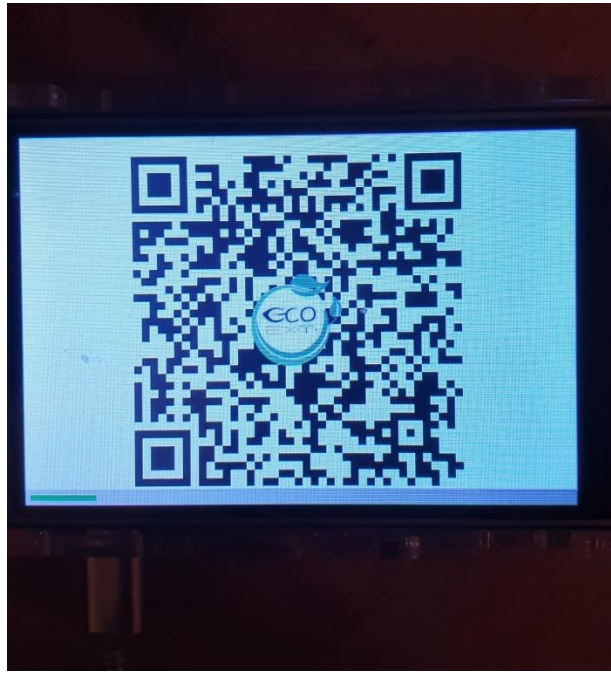


Figure 90: EcoExT QR Code being displayed on the Raspberry Pi.

#### 4.4.2 Scenario 2:

As an Establishment, I want to be able to display a screen, so that it shows the customer that the QR code was not scanned.

Result of testing:

- ✓ The establishment is able to display an error on the Raspberry Pi.

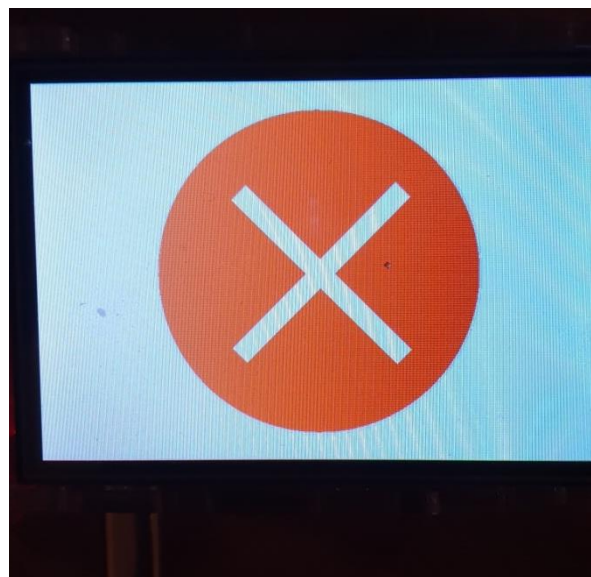


Figure 91: EcoExT error screen being displayed on the Raspberry Pi.

As an Establishment, I want to be able to display a screen, so that it shows the customer that the QR code was successfully scanned.

Result of testing:

- ✓ The establishment is able to display a success screen on the Raspberry Pi.

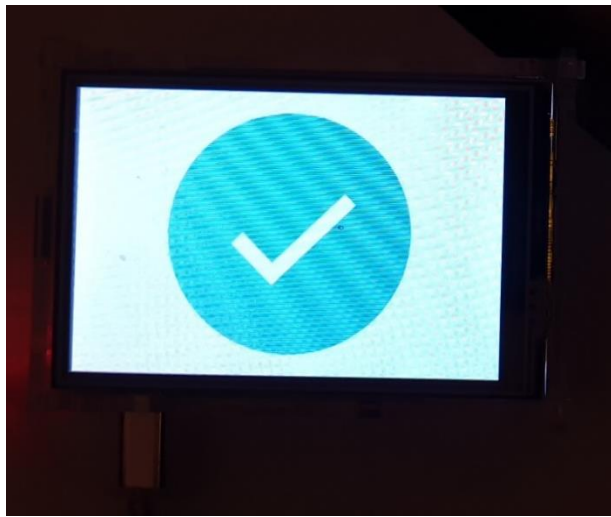


Figure 92: EcoExT success screen being displayed on the Raspberry Pi

#### **4.4.4 Scenario 4:**

As an Establishment, I want that the Raspberry Pi application not to display any other transaction to the user that is not their transaction, so that slow connections do not interrupt the current flow of the particular transaction.

Result of testing:

- ✓ The establishment does not display any other QR codes and the screen is locked on the Raspberry Pi.

#### **4.4.5 Scenario 5:**

As an Establishment, I want to be able to go back to the listening screen (Home Screen) after a QR Code has either been scanned or not, so that the application can listen to new transactions.

Result of testing:

- ✓ The establishment displays a home screen on the Raspberry Pi.



Figure 93: EcoExT home screen being displayed on the Raspberry Pi.

## 4.5 Automated testing

Python offers built in functionality to perform automated tests making it one of the easiest programming languages to perform automated tests. Even though functions can be written and called to perform the required tests, Python 3 comes with a module built in called unit test. This module allows developers to execute tests by creating class that only inherits the TestCase Class inside the module.

Given a quirk in how the GUI Framework tkinter works some testing that required multi-threading for testing could not be applied, but testing the main functionality of the application was achieved. All of the other tests performed were exploratory tests, because the Raspberry Pi Application was tested within the functionality of the whole system built, this includes the API, Database, the Mobile Application, and the Raspberry Pi App.

In the following sections these automated tests are going to be explored and it will be discussed why these were thought to be the most important bits that needed to be tested.

### 4.5.1 Writing the testing class

In Python 3, there are a few ways to write a testing class that will be run when all our automated tests are to be run. In the case of this application, the **unittest** class is going to be used. The first thing is to import this module into our file, as shown in the figure below:

```
1  """Imports of the dependencies of this class."""
2  import unittest
```

Figure 94: Importing the unittest Module into the testing file.

After having imported the module, the testing class has to be created. This, can be defined as any other Python 3 class with the only difference that this class is going to inherit the TestCase Class so when running the main test program this will know which classes to run (this is going to be explained in more detail in the section [II](#)). In the figure below, we can see the definition of this class, and how this inherits the TestCase Class.

```
11  """Class definition for the Application Tests"""
12  class ApplicationTests(unittest.TestCase):
13      """Class for the App Tests"""
```

Figure 95: Class Heading of the Testing Class

Having importing the unittest module and making a class that is a unittest.TestCase is not enough to test the application, now what is left to do is to write the methods that are going to perform the assertions, tests, to the different parts of our software but before getting to that part, let us see how these tests are run.

```
82  if __name__ == '__main__':
83      unittest.main()
84
```

Figure 96: Main Program of the Testing file

### 4.5.2 Running Our Tests

The first thing to do to run the tests is to create a main program that is going to call all the classes and their methods that assert the functionality of our software. The figure below shows how this main program is written.

In there, one can see that the class `ApplicationTests`, is never instantiated, this is due to the way the module unit test works. When running the line `unittest.main()`, see figure 94, the program looks for all those classes that are a `unittest.TestCase` within the file of that command and all the methods that are named `test*` are invoked and whatever assertion they have to do is executed.

Finally, to execute the file, e.g. `Test.py`, the command `python3 -m unittest -v Test.py` (In my system, Windows 10, the way of running a Python 3 script is to use the key word `py` instead of `python3`) has to be issued in the command prompt, in the folder where the file is stored. Then the `unittest.main()` is going to do what was mentioned in the previous paragraph and when looking for the methods which names start with the word `test`, is going to run these in alphabetical order.

After knowing how to write our testing class and how to execute our assertions or tests, the need of setting up the testing environment comes as a priority, after that the methods that are going to test the application can be written.

### 4.5.3 Setting up the testing environment

In order to perform the tests, one needs to set up the files and modules that are going to allow the `Test` file to assert the different scenarios that are going to be tested. In the figure number 95, we can see the `Test.py` file and a folder called `Tests` that contains all the mocking data and the Python modules that are going to be used in these tests.

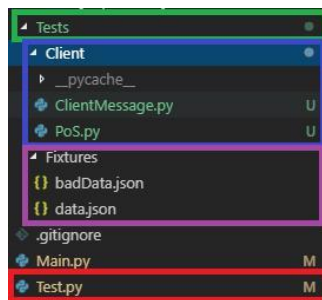


Figure 97: File Structure of the parts that are used to perform the tests

```

1  mutation addTransaction{
2    addTransaction(
3      label: "A Label"
4      description: "A Description"
5      socketInfo: {ipv4: "192.168.0.41", port: 31000}
6      items:[
7        {product: "A Product", quantity: 9, tax: 12.5 price: 1000.00},
8        {product: "Another Product", quantity: 3, tax: 10, price: 500.00}
9      ]
10     pageInfo: {
11       ammount: 10500.00
12       payment_type: "Debit Card"
13     }

```

Figure 98: GraphQL Mutation showing the format of the transaction data to be able to store it in the database successfully.

On figure 95 shows in red colour the test file, which contains the Main Testing Program, the Testing Class and the Assertion Methods. In green colour shows the folder that contains all the utilities to perform the tests. In purple colour can be seen the fixtures that are used to mock the data that is going to be transmitted by the Application, data. Json is a well formatted data of an EcoExT Transaction, that should be sent to the API without any inconvenient. The structure of the data follows the mutation showed in the figure 96. Json does not follow the format in figure 3.5, so it should give a TypeError Exception when that file is sent to the API. Finally, in blue colour, we can see the module that is going to send information to the listener side of the Pi Application. This module simulates a PoS System.

Now that all that is needed for the tests is set up, it is time to write our assertion method

#### 4.5.4 Testing Methods

Now that the test module is selected, our class header or definition is done, and that the testing environment was set up, the test method can be written. Three tests are going to be performed, and they are listed below:

1. Send a well formatted EcoExT Transaction to the GraphQL API and assert that the response is a Token of 128 characters.
2. Send a badly formatted Transaction to the GraphQL API and assert that a TypeError Exception is raised.
3. Send an EcoExT Transaction from the Client Program that simulates a PoS System to the Listening Thread of the Pi App and assert that the connection was established successfully.



All these tests are going to test that the application is always getting a particular response, the EcoExT Transaction has to have the shape and that a Client Program can connect and send a EcoExT Transaction successfully to the Pi App when the Pi App is up and running and waiting for any incoming traffic to its ipv4 address and to an specific port.

#### 4.5.4.1 Test 1 - Sending Transaction to the GraphQL API and Getting a 128 Characters Token

```

15 | def test1_SendTransactionToAPITokenLength(self):
16 |     """Tets that a Transaction can be stored in database and Retrives a 128 Characters Token."""
17 |     # First, we load the json file that represents the transaction from the folder fixtures
18 |     with open(r"Tests\Fixtures\data.json") as f:
19 |         message = json.load(f)
20 |
21 |     # Instance of the API connection
22 |     apiAddress = "127.0.0.1"
23 |     apiConnector = APIConnection(apiAddress)
24 |
25 |     # Sending transaction to the API
26 |     hostAddress = "127.0.0.1" # Host ipv4 address
27 |     hostListenerPort = 31000 # Host listener port
28 |     apiResponse = apiConnector.storeTransactionInDatabase(message["transaction"], hostAddress,
29 |                                                            hostListenerPort)
30 |
31 |     # Here, we assert if the token received has a length of 128 characteres
32 |     # It is difficult to test the partterns in the token given that is pseudo randomly
33 |     # created in the API. The length is the only thing that we want to ensure is 128.
    self.assertEqual(len(apiResponse["data"]["addTransaction"]["token_id"]), 128)

```

Figure 99: Test 1 - Assert Token Length

As discussed is the above section, in order for the `unittest.main()` to execute the assertion methods of our test class, these have to be name like `test*`, for this reason, the method for this first test is going to be called `test1_SendTransactionToAPITokenLength`. This test is first loading the data to be sent to the API, then, an instance of the `APIConnection` class is created, passing the ipv4 address of the API. Then the method that sends the transaction to the API is invoked and if everything is alright a token that represents said transaction such be returned. As establish from the beginning, the data sent will produce a successful response from the API with a Token, this token has to have 128 characters in order to pass the test.

After this, by executing the command `py -m unittest -v Test.py` the result of the test a positive, the test has been passed. What this means is that a 128 characters token was retrieved after an EcoExT Transaction of the form seen in the figure [3.5](#) was sent to the API to be store in the database.

```

C:\Users\guerr\Dropbox\PythonCodes\MP\PythonCodes\EcoExT>py -m unittest -v Test.py
test1_SendTransactionToAPITokenLength (Test.ApplicationTests)
Tets that a Transaction can be stored in database and Retrives a 128 Characters Token. ... ok

-----
Ran 1 test in 2.277s

OK

```

Figure 100: Result of the Test 1 - Assert Token Length



#### 4.5.4.2 Test 2 - Sending Transaction to the GraphQL API and TypeError Exception

If the data that is sent to the database does not comply is an expected failure that can be tested so it is assured that the data sent to the API has to meet certain requirements to be able to process it. In the figure above is shown this test, called test2\_SendBadDataToAPI. In said figure can be appreciated the line of code on which the Exception TypeError is asserted. The first one is used for the test in the above section and the second one is omitting the Payment Info field, causing the API to dispatch this data and sent an empty response to the program. This triggers a TypeError Exception that ultimately makes the assertion be verified, passing the test.

```
35 | def test2_SendBadDataToAPI(self):
36 |     """Tets that a Trasaction has to have a Structure in the file data.json if not it can be sent to
37 |     API."""
38 |     # First, we load the json file that represents the transaction from the folder fixtures
39 |     with open(r"Tests\Fixtures\badData.json") as f:
40 |         message = json.load(f)
41 |
42 |     # Instance of the API connection
43 |     apiAddress = "127.0.0.1"
44 |     apiConnector = APIConnection(apiAddress)
45 |
46 |     # Sending transaction to the API
47 |     # As well as we have to access the "transaction" index in the message to be able to send that data
48 |     # to API
49 |     # If not a exeption is raised
50 |     hostAddress = "127.0.0.1" # Host ipv4 address
51 |     hostListenerPort = 31000 # Host Listening port
52 |
53 |     with self.assertRaises(TypeError):
54 |         apiResponse = apiConnector.storeTransactionInDatabase(message["transaction"], hostAddress,
55 |         hostListenerPort)
```

Figure 101: Test 2 - Assert TypeError Exception

The figure 100 below shows the result of running the command **py -m unittest -v Test.py**, verifying what was implied in the previous paragraph, if the data does not meet certain characteristics the API will not be able to store it in the database.

```
C:\Users\guerr\Dropbox\PythonCodes\MP\PythonCodes\EcoExt>py -m unittest -v Test.py
test2_SendBadDataToAPI (Test.ApplicationTests)
Tets that a Trasaction has to have a Structure in the file data.json if not it can be sent to API. ... ok

-----
Ran 1 test in 1.149s

OK
```

Figure 102: Result of the Test 2 - Assert TypeError Exception.

#### 4.5.4.3 Test 3 - Listening for a Connection coming from the Client Program

For this test it is imperative to use multi-threading in order to execute the Pi App to start listening and then the following thread is to simulate the Client that is sending the message to the Pi Application. In order to pass this, test a connection between the Client and the Pi App has to be established otherwise the test will fail. In the figure 101 the test to assert this connection is shown, called **test3\_ListenerApplication**. In that figure can be seen how, by means of multiple threads, a listening application was instantiated and a client application, allowing to test if a connection can be achieved.

```
54 | def test3_ListenerApplication(self):
55 |     """Test that the listener Application is in fact getting the message from the client script."""
56 |     with open(r"Tests\Fixtures\data.json") as f:
57 |         message = json.load(f)
58 |
59 |     # Host ipv4 address and listenig port
60 |     hostAddress = "127.0.0.1"
61 |     hostListenerPort = 31000
62 |
63 |     # ipv4 address of API - Same as host computer because the test is done in a sandbox environment
64 |     apiAddress = "127.0.0.1"
65 |
66 |     # Instance of the Main Application
67 |     app = MainApp(hostAddress, hostListenerPort, apiAddress)
68 |     # Creating a Thread of the Main Application
69 |     appThread = Thread(target = app.startApplication)
70 |     appThread.start() # Start Thread of Main Application
71 |     appThread.join() # Join this Thread with main one
72 |
73 |     # Instance of Client Object - Simulation of PoS System
74 |     clientPoS = PoS(hostAddress, hostListenerPort, message)
75 |     # Creating a Thread of this PoS System
76 |     clientThread = Thread(target = clientPoS.sendMessageToServer)
77 |     clientThread.start() # Start Thread of PoS System
78 |     clientThread.join() # Join this Thread with other ones
79 |
80 |     app.getWindows().getHomeWindow().onClosing() # This is to kill any instances of the windows that are
        still in the background
```

Figure 103: Test 3 - Assert Connection Established.

After running the command **py -m unittest -v Test.py**, the assertion was confirmed, a connection was established between the Client and the Pi App and hence passing the test. The figure 102 shows the result of running this command, clearly when connections are taking place.

```
C:\Users\guerr\Dropbox\PythonCodes\MP\PythonCodes\EcoExt>py -m unittest -v Test.py
test3_ListenerApplication (Test.ApplicationTests)
Test that the listener Application is in fact getting the message from the client script. ... Listening on ('127.0.0.1', 31000)
GUI not created for reasons known!
Starting connection to ('127.0.0.1', 31000)
Accepted connection from ('127.0.0.1', 52076).
Closing connection!
Got result: Transaction Stored Successfully in Database - GUI is down.
Closing connection to ('127.0.0.1', 31000)
Closing connection to ('127.0.0.1', 52076).
Client connection closed!
Server Connection closed!
ok

-----
Ran 1 test in 7.328s

OK
```

Figure 104: Result of the Test 3 - Assert Connection Established.

#### 4.5.4.4 Executing all the tests at once

So far, the tests have been executed isolated from each other, but the idea of running automated tests is to run them all at once so any error that may arise can be tackled and a broader picture of the application can be seen. This time the command `py -m unittest -v Test.py` is going to be run but none of the tests are going to be commented so all of them can be run at once. The figure 103 shows the result of running all these tests, getting the same answer than when they were executed on their own, all the tests passed the assertion

```
C:\Users\guerr\Dropbox\PythonCodesMP\PythonCodes\EcoExt>py -m unittest -v Test.py
test1_SendTransactionToAPITokenLength (Test.ApplicationTests)
Tests that a Transaction can be stored in database and Retrieves a 128 Characters Token. ... ok
test2_SendBadDataToAPI (Test.ApplicationTests)
Tests that a Transaction has to have a Structure in the file data.json if not it can be sent to API. ... ok
test3_ListenerApplication (Test.ApplicationTests)
Test that the listener Application is in fact getting the message from the client script. ... Listening on ('127.0.0.1', 31000)
GUI not created for reasons known!
Starting connection to ('127.0.0.1', 31000)
Accepted connection from ('127.0.0.1', 52098).
Closing connection!
Got result: Transaction Stored Successfully in Database - GUI is down.
Closing connection to ('127.0.0.1', 52098).
Closing connection to ('127.0.0.1', 31000)
Client connection closed!
ok

-----
Ran 3 tests in 6.358s

OK
Server Connection closed!
```

Figure 105: Result of the Tests Assert Token Length, Assert TypeError Exception, and Assert Connection Established.

Now, extra imports were done into our test file in order to load data or to create instances of the classes used within our test methods, these imports are shown in the figure 104 below.

```
3 import json
4 from threading import Thread
5 from multiprocessing.pool import ThreadPool
6
7 from RaspAppPi.Model.DatabaseConnectors.APIConnection import APIConnection
8 from Tests.Client.Pos import Pos
9 from RaspAppPi.MainApp import MainApp
```

Figure 106: Test 3 - Assert Connection Established.

## 4.5 Automated testing conclusions

As reported, Python is a very friendly programming language in which performing automated tests is very easy, one just need to sit down and think about which way is the best to test your

algorithm. As a system that has several parts is important to find the main parts of each unit to test them so proper functionality can be assured. That is why it was important to test that in order to store a transaction in the database the API is waiting for a particular format, and if this is received then it has to be checked if the token, that represent said transaction, has the characteristics of the design that was set for it when it was develop the algorithm that creates it, a base64 string of 128 characters.

Finally, it is very important to ensure that the Pi Application is able to establish connections with the Client Program in order to process the transaction of the PoS System.

## **Chapter 5: Conclusion**

### **5.1 Project success and results**

The initial aim of the project was to design an application that ensures a user is able to scan a QR Code in order to gather a digital receipt where they would be able to save it into the application and do not need a physical paper receipt.

Research, planning and design of the application, implementation of the system and testing and evaluation reports were conducted so that the final prototype of the proposed application could be achieved.

The success of the application is when above all, the paper receipt is not needed to the final user of the application and by using the system, the user is able to save the digital receipt and control the expenses transactions into their digital purses.

Future features to be added in the subsequent versions of the applications were also considered and below we point out all of them.

### **5.2 Suggestions for further work**

#### **5.2.1 Cloud storage**

Cloud computing is a new version of the internet and it has such a relevance into our project as we need to access data and information with high speed and accuracy.

As a further work suggestion, we are proposing to use one of the three main cloud storage providers on the market now. Being them, Azure, Google Cloud or Amazon Cloud.

#### **5.2.2 Continuous Integration**

The practice of merging all developer working copies to a shared mainline several times a day is another tool used on our project where each check-in is then verified by an automated build, allowing teams to detect problems early. Another further work suggestion is to use the tool used to help us with the continuous integration, which is: Travis-CI. This tool allows to connect with GitHub and it makes the process of verifying more agile.

### 5.3 Features planned to be added in the future

- Premium Membership  
Could include features shown below:  
Attachment - upload the picture - a physical receipt
- FAQ - A help guide
- Budget - to set an estimate of income and expenditure for a set period of time.
- Planned Payment - to set up direct debits such as electric or phone bills
- Loyalty Card - to provide the option to any establishment to use the app as their digital Loyalty card.
- Debts - to manage your debts
- Shared Purses - to share a purse with one (1) or more users.
- Machine Learning - to retrieve all the items from a receipt straight to the app.
- Suggestions - based on what you consume on a regular basis, the app will be able to give suggestions. E.g. "You haven't bought the dog's food this week".

## 6. Bibliography

A. Bauer, B. (2016). Tips to reduce your exposure to BPA. [online] Mayo Clinic. Available at: <https://www.mayoclinic.org/healthy-lifestyle/nutrition-and-healthy-eating/expert-answers/bpa/faq-20058331> [Accessed 12 Oct. 2018].

Akela, A. (2016). An Introduction to Docker: The Performance Perspective – Part 1 - DZone Performance. [online] dzone.com. Available at: <https://dzone.com/articles/an-introduction-to-docker-the-performance-perspect> [Accessed 10 Dec. 2018].

AltexSoft. 2019. Functional and Non-functional Requirements: Specification and Types. [ONLINE] Available at: <https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/>. [Accessed 08 May 2019].

Andrey Herasymenko. 2019. UX design for beginners: what is UI/UX Design? – UX Collective. [ONLINE] Available at: <https://uxdesign.cc/ux-design-for-beginners-what-is-ui-ux-design-89bc4da54cbf>. [Accessed 08 May 2019].

Bonnie, E. (2018). Project Management Basics: 6 Steps to a Foolproof Project Plan. [online] Wrike.com. Available at: <https://www.wrike.com/blog/foolproof-project-plan/> [Accessed 27 Sep. 2018].

Bryansryan.ie. (n.d.). Till Rolls | Discount Till Rolls | Thermal Rolls | Receipt Rolls. [online] Available at: <https://bryansryan.ie/office-supplies/paper-products/till-rolls/> [Accessed 8 Nov. 2018].

BT.com. (2016). Could it be the end of the paper receipt?. [online] Available at: <http://home.bt.com/tech-gadgets/tech-news/tesco-contactless-receipts-phone-11364108340323> [Accessed 19 Oct. 2018].

Cashdrawers.ie. (n.d.). Epson TM-T20II Receipt Printer | Cash Drawers Ireland. [online] Available at: <https://www.cashdrawers.ie/-Epson-TM-T20-USB-Receipt-Printer> [Accessed 8 Nov. 2018].

Chapter 2. System design. 2019. Chapter 2. System design. [ONLINE] Available at: [http://moodle.autolab.uni-pannon.hu/Mecha\\_tananyag/szoftverfejlesztési\\_folyamatok\\_angol/ch02.html](http://moodle.autolab.uni-pannon.hu/Mecha_tananyag/szoftverfejlesztési_folyamatok_angol/ch02.html). [Accessed 08 May 2019].

ConferenceCall.co.uk blog. (2018). What is an e-receipt? And should your business use them? - ConferenceCall.co.uk blog. [online] Available at: <https://www.conferencecall.co.uk/blog/what-is-an-e-receipt/> [Accessed 14 Nov. 2018].

COTE, A. (2018). The Beginner's Guide to Using Project Planning Software. [online] Paymo. Available at: <https://www.paymoapp.com/blog/project-planning-software/> [Accessed 7 Nov. 2018].

da Silva, M. (2016). Why Going Paperless Can Help Your Retail Business (And How to Do It). [online] Retail Marketing Blog - Retail News, Trends, Store Tips, and More by Shopify. Available at: <https://www.shopify.com/retail/why-going-paperless-can-help-your-retail-business-and-how-to-do-it#> [Accessed 19 Oct. 2018].

Dataprotection.ie. (n.d.). Guidance for Retailers issuing e-receipts - Data Protection Commission - Ireland. [online] Available at: <https://dataprotection.ie/docs/Guidance-for-Retailers-issuing-e-receipts/1677.htm> [Accessed 14 Nov. 2018].

Dictionary, C. (2018). CONSUME | meaning in the Cambridge English Dictionary. [online] Dictionary.cambridge.org. Available at: <https://dictionary.cambridge.org/dictionary/english/consume> [Accessed 7 Dec. 2018].

Digital Check. (2018). How Much Does Your Receipt Printer Cost You? - Digital Check. [online] Available at: <https://www.digitalcheck.com/receipt-printing-costs/> [Accessed 19 Oct. 2018].

Edwards, L. (2010). BPA from thermal paper receipts passes through the skin. [online] Medicalxpress.com. Available at: <https://medicalxpress.com/news/2010-11-bpa-thermal-paper-receipts-skin.html> [Accessed 12 Oct. 2018].

En.wikipedia.org. (n.d.). Dot matrix printing. [online] Available at: [https://en.wikipedia.org/wiki/Dot\\_matrix\\_printing](https://en.wikipedia.org/wiki/Dot_matrix_printing) [Accessed 25 Oct. 2018].

En.wikipedia.org. (n.d.). Electronic receipt. [online] Available at: [https://en.wikipedia.org/wiki/Electronic\\_receipt](https://en.wikipedia.org/wiki/Electronic_receipt) [Accessed 14 Nov. 2018].

En.wikipedia.org. (n.d.). Thermal paper. [online] Available at: [https://en.wikipedia.org/wiki/Thermal\\_paper](https://en.wikipedia.org/wiki/Thermal_paper) [Accessed 12 Oct. 2018].

En.wikipedia.org. (n.d.). Thermal printing. [online] Available at: [https://en.wikipedia.org/wiki/Thermal\\_printing](https://en.wikipedia.org/wiki/Thermal_printing) [Accessed 19 Oct. 2018].

Entity Relationship Diagram (ERD) - What is an ER Diagram? . 2019. Entity Relationship Diagram (ERD) - What is an ER Diagram? . [ONLINE] Available at: <https://www.smartdraw.com/entity-relationship-diagram/>. [Accessed 08 May 2019].

Geeta T. (n.d.). 6 Stages of Software Development Process. at: <https://www.synapseindia.com/6-stages-of-software-development-process/141>. [Last Accessed on the 4th of April 2019].

Green America. (2018). Skip the Slip Report: Environmental Costs & Human Health Risks of Paper Receipts with Proposed Solutions. [online] Available at: <https://www.greenamerica.org/report-STs> [Accessed 12 Oct. 2018].



Green America. (2018). “Skip the Slip” Report: Toxic Paper Receipts Jeopardize Health of Millions, Waste 12 Million Trees and 13 Billion Gallons of Water Annually. [online] Available at: <https://www.greenamerica.org/press-release/skip-slip-report-toxic-paper-receipts-jeopardize-health-millions-waste-12-million-trees-and-13-billion-gallons> [Accessed 12 Oct. 2018].

Gunhan Sancar/Mobile Application Developer. (2015, 10 29). Retrieved 02 2019, from Best Practice to Instantiate Fragments with Arguments in Android: <https://gunhansancar.com/best-practice-to-instantiate-fragments-with-arguments-in-android/>

Gupta, M. (2018). What is a true cost of a paper receipt to a retailer?. [online] Quora. Available at: <https://www.quora.com/What-is-a-true-cost-of-a-paper-receipt-to-a-retailer> [Accessed 12 Oct. 2018].

Harned, B. (n.d.). How to Write a Good Project Plan in 10 Steps | Project Management Guide. [online] Teamgantt.com. Available at: <https://www.teamgantt.com/guide-to-project-management/how-to-plan-a-project> [Accessed 27 Sep. 2018].

Hines, W. (2013). Going Paperless: The Hidden Cost of a Receipt. [online] HuffPost. Available at: [https://www.huffingtonpost.com/will-hines/going-paperless-the-hidde\\_b\\_3008587.html](https://www.huffingtonpost.com/will-hines/going-paperless-the-hidde_b_3008587.html) [Accessed 12 Oct. 2018].

Huntoffice.ie. (n.d.). Till Rolls - HuntOffice.ie Ireland. [online] Available at: <https://www.huntoffice.ie/till-rolls.html> [Accessed 8 Nov. 2018].

IBM Knowledge Center Error . 2019. IBM Knowledge Center Error . [ONLINE] Available at: <https://www.ibm.com/support/knowledgecenter/en/SSPJVK/DigitalRecommendations/UserGuide/examplebusinessrules.html>. [Accessed 08 May 2019].

Kablo. (n.d.). Top 5 Negative Side Effects Of BPA - And How To Avoid It. [online] Available at: <https://shopkablo.com/blogs/the-reformist/dangerous-side-effects-of-bpa-plastic> [Accessed 12 Oct. 2018].

Namdeo, P. (2018). What is the importance of a database management system? [online] Quora. Available at: <https://www.quora.com/What-is-the-importance-of-a-database-management-system> [Accessed 10 Dec. 2018].

Nisbets.ie. (n.d.). Till rolls, thermal rolls and ink rollers | Nisbets Catering Equipment. [online] Available at: [https://www.nisbets.ie/tableware-and-bar-supplies/cash-handling/till-rolls-and-ink-rollers/\\_/a33-3](https://www.nisbets.ie/tableware-and-bar-supplies/cash-handling/till-rolls-and-ink-rollers/_/a33-3) [Accessed 8 Nov. 2018].

Panda Paper Roll. (n.d.). Paper Receipts: Why Do We still Use It in the Digital Age? | Panda Paper Roll. [online] Available at: <https://pandapaperroll.com/why-we-use-paper-receipts-in-digital-age/> [Accessed 25 Oct. 2018].

PaperlessKitchen.com. (2014). Reseed Program for Eliminating Paper Receipts. [online] Available at: <https://www.paperlesskitchen.com/blogs/news/11815625-reseed-program-for-eliminating-paper-receipts> [Accessed 12 Oct. 2018].

Peter Alexander. 2019. Data Storage Solutions for Your Business. [ONLINE] Available at: <https://www.entrepreneur.com/article/172226>. [Accessed 08 May 2019].

Pope, C. (2018). Ask for a receipt; no legal obligation on shop to give you one. [online] The Irish Times. Available at: <https://www.irishtimes.com/news/consumer/ask-for-a-receipt-no-legal-obligation-on-shop-to-give-you-one-1.3443281> [Accessed 12 Oct. 2018].

Porter, B. and Temsamani, A. (2018). Environmental Costs & Human Health Risks of Paper Receipts with Proposed Solutions. [online] Greenamerica.org. Available at: <https://www.greenamerica.org/sites/default/files/2018-08/Skip%20the%20Slip%20Report%20-%20Green%20America.pdf> [Accessed 12 Oct. 2018].

Psychology Today. 2019. Why We Consume So Much | Psychology Today. [ONLINE] Available at: <https://www.psychologytoday.com/us/blog/the-human-beast/201805/why-we-consume-so-much>. [Accessed 07 May 2019].

Recommended Data Repositories | Scientific Data. 2019. Recommended Data Repositories | Scientific Data. [ONLINE] Available at: <https://www.nature.com/sdata/policies/repositories>. [Accessed 08 May 2019].

Riant Soft. (2013). 6 basic steps of software development process. at: <https://www.slideshare.net/RiantSoft123/6-basic-steps-of-software-development-process>. [Last Accessed on the 4th of April 2019].

Robins, D. (2017). Hybrid: A new project management approach. [online] CIO. Available at: <https://www.cio.com/article/3222872/project-management/hybrid-a-new-project-management-approach.html> [Accessed 14 Nov. 2018].

Rouse, M. (2013). What is QR code (quick response code)? - Definition from WhatIs.com. [online] WhatIs.com. Available at: <https://whatis.techtarget.com/definition/QR-code-quick-response-code> [Accessed 10 Dec. 2018].

Ross, S. (2018). 5 Surprising Benefits of Tracking Your Spending. [online] Lifehack. Available at: <https://www.lifehack.org/482228/5-surprising-benefits-of-tracking-your-spending> [Accessed 12 Dec. 2018].

Satoamerica.com. (n.d.). Thermal Transfer vs. Direct Thermal: Five Key Considerations | SATO America. [online] Available at: <https://www.satoamerica.com/resource-library/learning-center/dt-vs-tt.aspx> [Accessed 25 Oct. 2018].

Shop4rolls.ie. (n.d.). Till Rolls, Thermal Rolls, Credit Card & Paper Rolls - Shop4Rolls.ie. [online] Available at: <https://www.shop4rolls.ie/> [Accessed 8 Nov. 2018].

System Implementation - SEBoK. 2019. System Implementation - SEBoK. [ONLINE] Available at: [https://www.sebokwiki.org/wiki/System\\_Implementation](https://www.sebokwiki.org/wiki/System_Implementation). [Accessed 14 April 2019].

Usborne, S. (2016). The real reason shops want you to sign up for e-receipts. [online] the Guardian. Available at: <https://www.theguardian.com/travel/shortcuts/2016/oct/16/shops-sign-up-e-receipts-proof-of-purchase> [Accessed 14 Nov. 2018].

Vikingdirect.ie. (n.d.). Paper Rolls | Printer Paper | Viking Direct IE. [online] Available at: <https://www.vikingdirect.ie/en/paper-printing-c-108/printer-paper-c-10804/paper-rolls-c-10804008> [Accessed 8 Nov. 2018].

Weisbaum, H. (2014). Paper or email? Pros and cons of digital receipts. [online] CNBC. Available at: <https://www.cnbc.com/2014/01/23/paper-or-email-pros-and-cons-of-digital-receipts.html> [Accessed 12 Dec. 2018].

Westland, J. (2016). What Is Hybrid Methodology? - ProjectManager.com. [online] ProjectManager.com. Available at: <https://www.projectmanager.com/blog/what-is-hybrid-methodology> [Accessed 14 Nov. 2018].

Zebra Technologies. (n.d.). Direct Thermal Labels | Thermal Transfer | Zebra. [online] Available at: <https://www.zebra.com/gb/en/resource-library/getting-started/direct-thermal-thermal-transfer/direct-thermal-faq.html> [Accessed 25 Oct. 2018].

## Appendix A – Android Class Diagram

A class diagram for the android application was designed, although the documentation did not fit the previous mentioned. So, in this Appendix the below figure shows the class diagram.

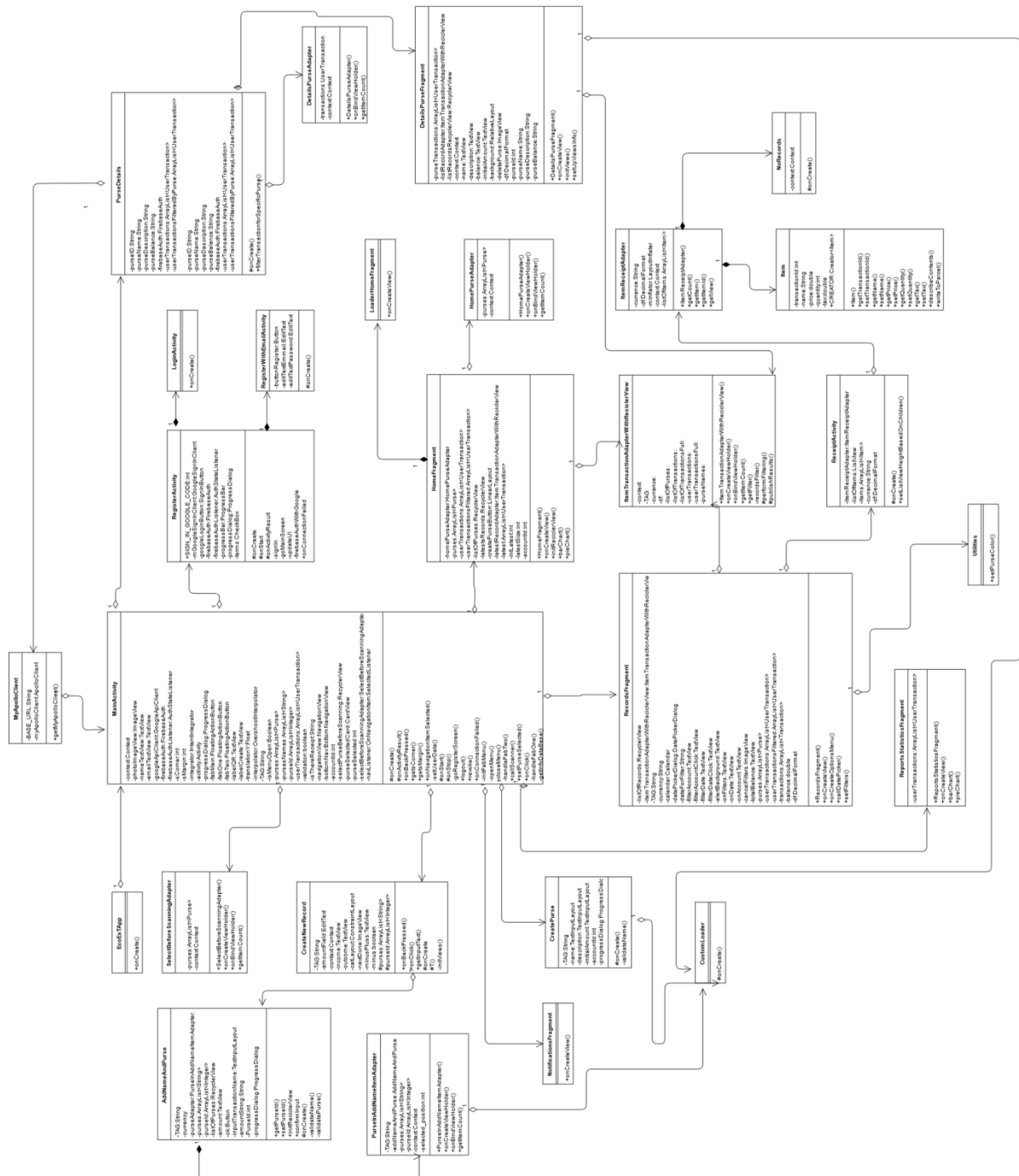


Figure 107: Android Class Diagram

## Appendix B - Project Planning

Activities related to project planning began to be carried out in order to structure our project in the best way possible. The planning necessary for activities to be carried out was divided in some steps.

### 1. Identification and needs in the market

We approached few students at CCT College in order to identify the interest, needs and expectations of the application before creating our planning. We have got an ethical approval to conduct the research and the questionnaire carried out on the first chapter of this documentation. Please see copy of the Ethical Approval below:



Figure 108: Ethical approval from CCT College to conduct research

## 2. Definition of objectives and priorities

Once we had a list with all the needs and goals to be achieved, we prioritized these and in the same way we have set the main goals for the project, outlining the project objectives that we should achieve.

## 3. Defining deliverables

We identified the main dates to be delivered and, in this way, we created a reasonable timeline that allows us to be guided to see how the process was being done. Below we have images of two different timelines, the project in general and the documentation in complement:

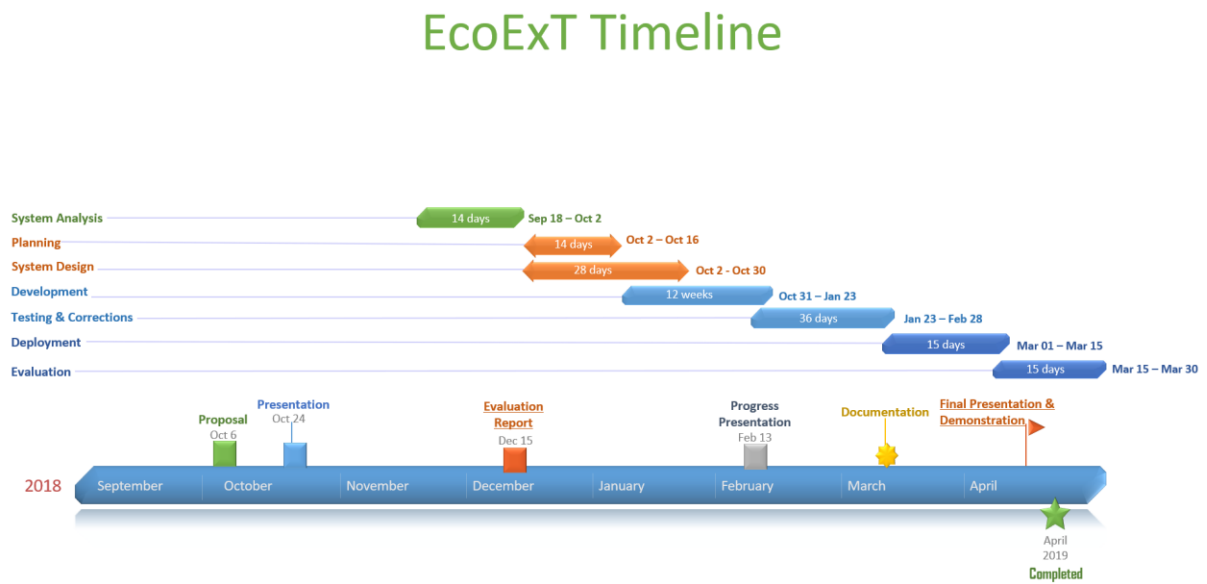


Figure 109: EcoExt project timeline.

## Documentation Timeline

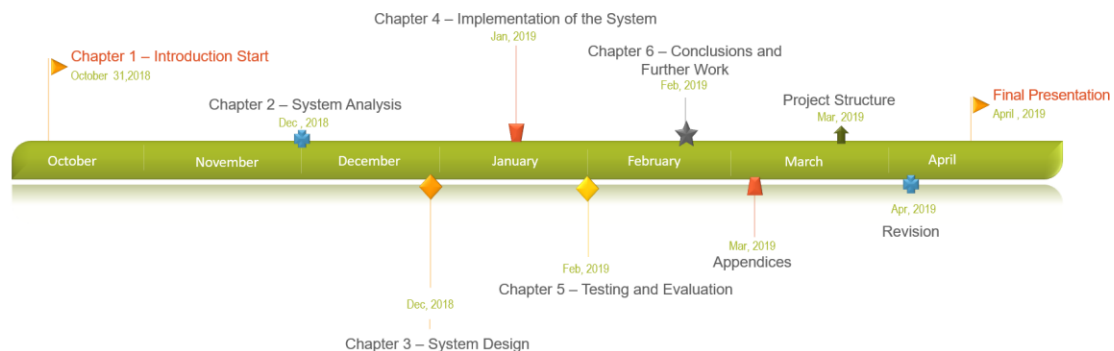


Figure 110: EcoExt documentation timeline.

Although, the first deliverables and the timeline planning could not be achieved and because of that we had to change the approach and define new deliverables so that we could meet the final deadline. Below you can find the updated versions of the two different images timelines, the project and the documentation:

## EcoExT Timeline

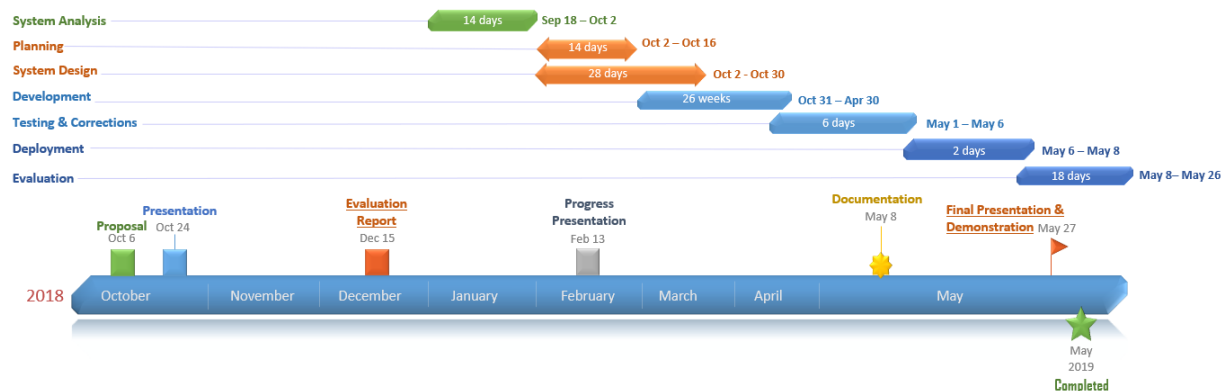


Figure 111: EcoExt project timeline updated.

## Documentation Timeline

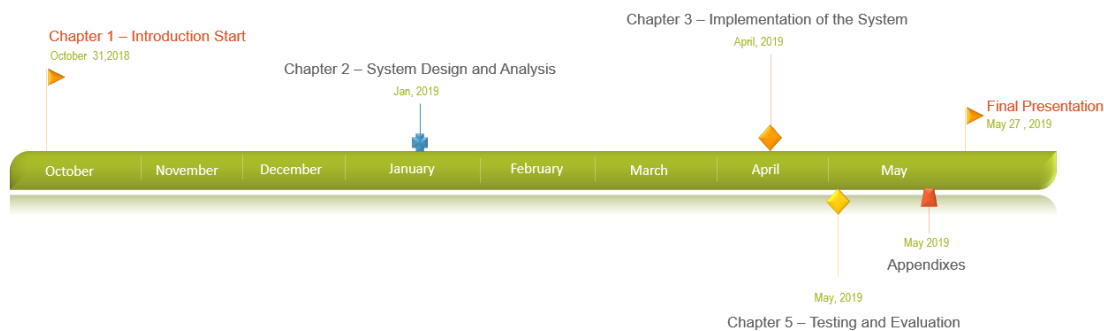


Figure 112: EcoExt documentation timeline updated.

EcoExt project timeline (see [image 4](#)) is divided into seven (7) different phases:

- **System Analysis** - where we study and research about what we are willing to develop in order to accomplish them in a most efficient way.
- **Planning** - where we define goals and manage tasks creating schedules and reporting the progress of each tasks. The whole scope idea defined in a most appropriate method with the aim to complete the project environment.
- **System Design** - where we define and create the architecture, modules, diagrams for the system to satisfy the specific requirements.
- **Development** - the phase of actually developing the whole system.
- **Testing & Corrections** - where we test from an end-to-end perspective in order to correct what have been done wrongly and also make sure we have the required functionalities of the system.
- **Deployment** - where we assembly or transform the EcoExt product from a temporary or development state to a permanent or desired state.
- **Evaluation** - where we make sure that the whole system is performed well and in market conditions. Also, where we present the working prototype of the whole developed EcoExt system.



## 4. Teamwork

Defining and delegating tasks for each member of the group was decided between all of the members. On the Appendix B (Individual Contribution) you can find with more details what each one of us is responsible to do so.

With the concept of collaborative effort to help each other in order to achieve the evaluation of the whole system, we have been working together towards a common goal to make sure we meet the requirements.

### 4.1 The Team Members

The team members worked together in order to achieve the best version of EcoEXT prototype. Each member had their tasks and responsibilities on completing the system.



Figure 113: Team members sprint. From left to right, Asmer, Miguelantonio, Carina, Gabriel and Eduardo.

## 5. Methodology

A research was conducted in order to find the best approach of methodology to be used for EcoExt system. Nowadays few methodologies are used in order to have a fast and redundant development.

According to Jason Westland to Project Manager website: *“Agile Hybrid Methodology is the poster child for hybrid mythology in that it was developed for a highly flexible environment with an*

*openness to change and non-hierarchical forms of leadership. Agile is a big tent that holds a lot of methodologies, such as Scrum, Kanban and Lean.”*

Hybrid comes to EcoExt to combine the best of Agile and the work breakdown structure, the system development life cycle divides the development cycle into short term deliveries called “sprints”.

Every week EcoExt team conduct “sprints” on how the process of the project is going so far, also to set goals for the next “sprint”.

Hybrid can handle requirement changes and deliver products in stages, that was the better approach for EcoExt as we have few stages to conclude to be able to proceed to the next step. In hybrid, the planning is done using the waterfall approach. The execution and delivery are handled by the Agile method. The image below shows how the methodology is being practically adapted on a weekly basis using one of the online tools called “Trello”:

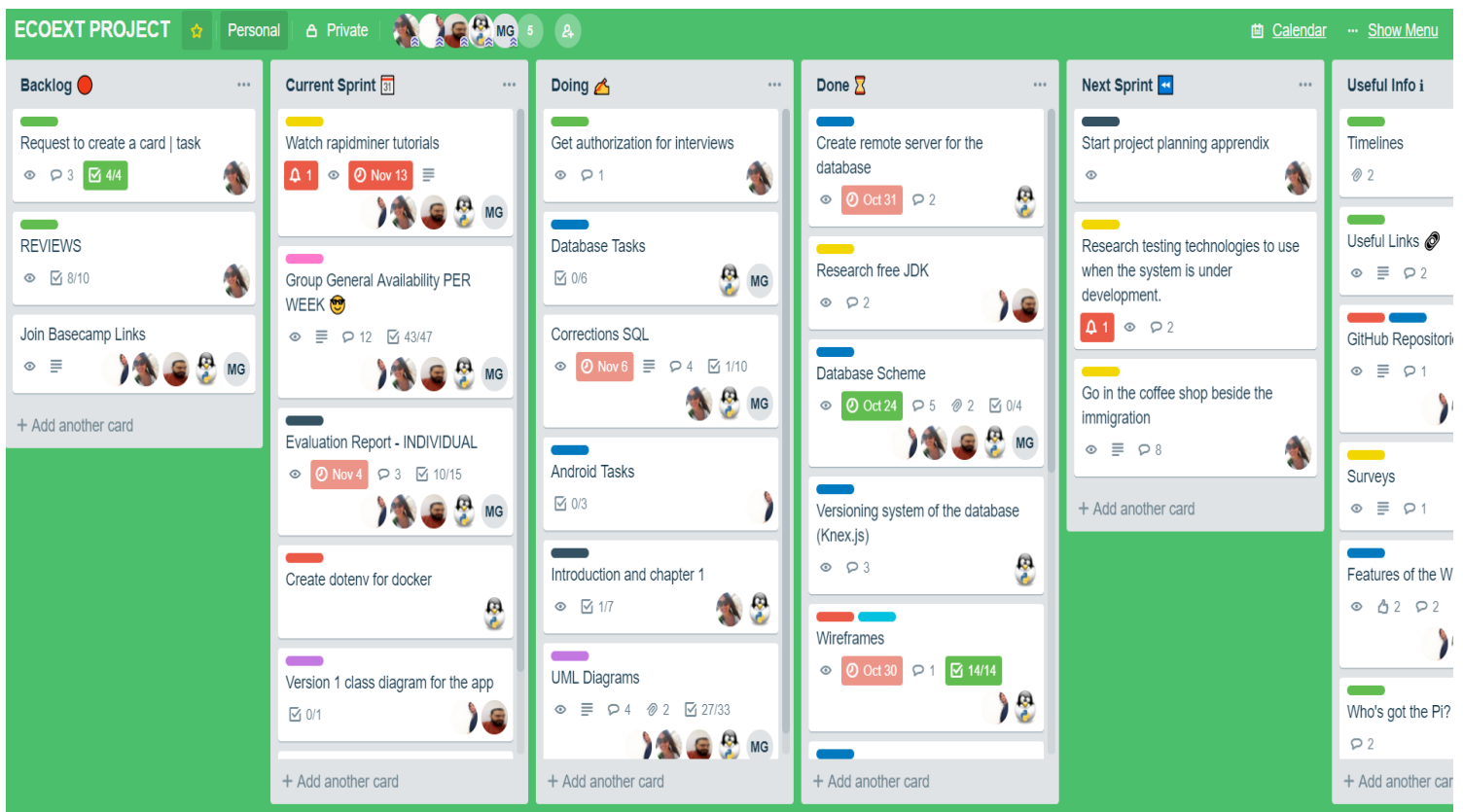


Figure 114: Trello in use as a tool to manage tasks.

- The Backlog represents every single task that has to be done.
- The Current Sprint represents the tasks carried out on the same week.
- And Next Sprint represents the tasks that we will have to deal on the following week every single “sprint” that we have are based on a weekly approach.

## Appendix C – Self Reports – Group Journal



**Carina Lins**  
Team Manager

---

In charge of the management of the team, Carina has been dealing with the agenda as well as managing the communication platforms such as Trello and Google Drive; Responsible for setting up meetings and monitoring the tasks that have been carried out by each member of the team. Responsible for the documentation and deliverables of all the project requirements. Responsible for the Testing of the application and ensuring quality of the product.

---

### From beginning up to evaluation:

- Organization and preparation of weekly sprints gathering the planning structure and the tasks to be completed for each individual team member.
- Insertion of related tasks in Trello, managing time, deliverables and delegating the task with the team according to the needs of the project timeline.
- Decision of which technologies are useful and helpful to the project.
- Proposal documentation and Slides to be presented in class.
- Helped with Use Case diagrams in order to plan and design the project.
- Research about methodologies used nowadays, such as waterfall, agile, extreme programming, hybrid.
- Team work and decisions to achieve a database table design along with diagrams.
- Self-study how GitHub works and how to use and apply the tool to the project.
- Conducted internal surveys on how likely are potential participants and companies to adopt the system.
- Research about papers, printers, ink cartridge, ribbons, electronic receipts, new GRDP rules for data protection, environment and costs.
- Carried out interviews and researches in order to determine criteria and project planning. The costs, the environment impact and the usage of paper receipts were also part of my contribution for the first chapter.
- Guidelines, graphs and images were created by me to use in the first chapter along with the research done.
- All the content written in the documentation was done by me with the Implementation Chapter helped by the developers and I readapt some ideas that the team have had to contribute to the first chapter.

- How EcoExt was born and the benefit of adopting our system was also part of my contribution.
- The technologies that are being used and the methodologies adapted to our project.
- Revision of all the work done by the team if it aggregates to the project guidelines.
- Meetings with the project supervisor with the aim to investigate concerns and how we can follow up with the idea.
- Competition of all chapters of the documentation – Introduction, System Analysis and Design, Implementation of the System, Testing and Evaluation, Conclusion and Appendixes.
- Working along with the team to format the final document of the system.
- Completed of Appendix and group reports.
- Testing of the running application.
- Writing and creating different scenarios to be tested.
- Assuring quality of the product, that the application after tested meet the requirement's criteria of the design.
- Responsible for concluding all the chapters of the documentation and also the research within it.
- Ensuring the whole project meets the criteria that have been requested on the handbook and that the team members deliver the product on time.



## **Gabriel Santos**

Back end Developer

---

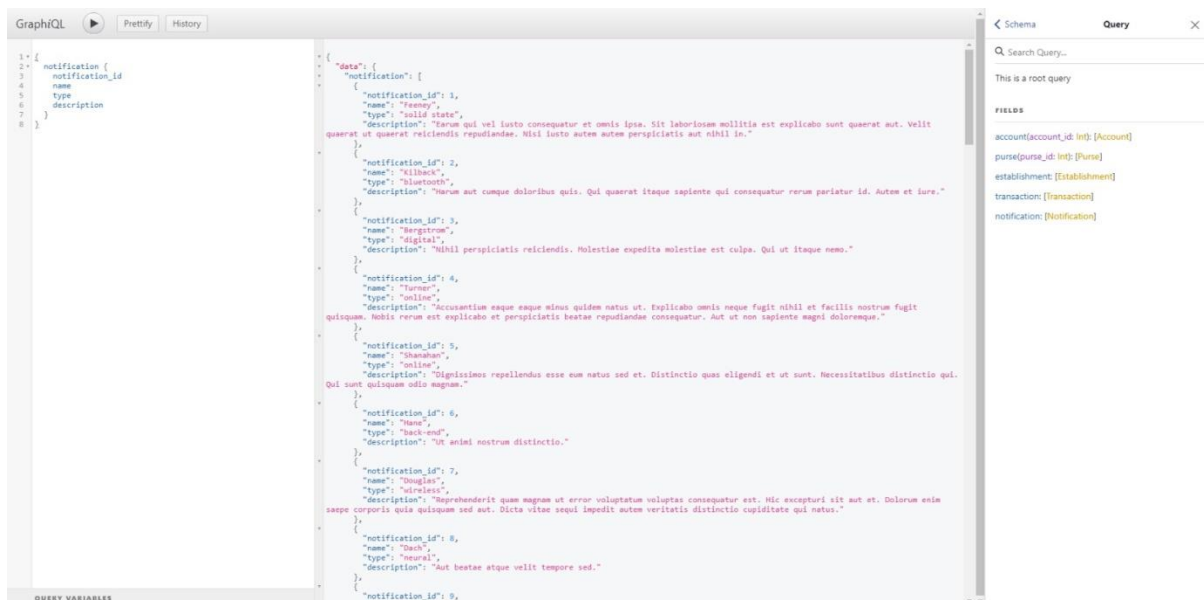
In charge of the application from the back-end perspective, Gabriel had worked technologies that will allow the system to have a full working behaviour with efficiency and performance. Responsible for setting up the environment such as Docker, GraphQL and Database structures and designs. Responsible for building the API and connecting it to the frontend application.

---

### **From beginning up to evaluation:**

- Contributed with the design of the database
  - Applied normalizations to the database
  - Reviewed the ERD
- Implemented the GraphQL Server
  - Researched about the design of the schemas
  - Researched about how to implement GraphQL using a relational database
- Implemented the database versioning using Knex.js
  - Tested the database design by feeding data to id
  - Created versions of the database so we could follow our changes
  - Tested the integrity of the relations by querying data from it
- Implemented Docker for the backend development
  - Managed to put together Docker containers running:
    - MySQL
    - Node.js webserver running GraphQL
    - Contributed to Raspberry Pi container
- Helped in the design the wireframes for the android app
  - Register
  - Login
  - Home
  - Home/Menu
  - Records
  - New transaction
  - Labels
  - Labels/Create
  - Scan Code
  - Profile
  - Add Purse
  - Categories
  - Categories/Delete
- Researched on how paper receipts impact on the environment
- Helped in the design of the surveys
- Researched about technologies to use in the project

- Implemented the API using GraphQL
- Adapted the design of the database to fix missing relations and add new ones to suit the needs of the application
- Implemented Sequelize.js ORM in order to manage GraphQL relation
- Implemented Knex.js to do the versioning of the database
- Improved the performance of the Docker image and containers
- Pushed Docker image to docker hub
- Tested on continuous integration and deployment using Travis-ci and Heroku
- Tested on how to implement an online database using AWS EC
- Tested API using Insomnia
- Helped advising on the implementations of the Android application
- Helped advising on the implementation of the Raspberry Pi application.



GraphQL API running over Docker feeding from a MySQL database that also runs over Docker (by Gabriel)



## **Miguelantonio Guerra**

Back end Developer

---

In charge of the developing the raspberry pi interface, frontend, backend and the connectivity between all the different devices that are going to be interacting within the system, from the raspberry pi to and the mobile application. From the frontend perspective the design of the QR code using Python modules and how the server is going to control how the data flows.

---

### **From beginning up to evaluation:**

- Contributed with the design of the database by presenting:
  - Database requirements
  - Final ER diagram
  - Relational Model using the 7 steps algorithm
  - Normalised the database relational model up to 4NF
  - Created a data dictionary for all relations
  - Created the physical model using MySQL create statements
- Research of the tools, technologies and programming languages to use to generate the interaction between the raspberry pi, the server, the mobile app, and the data exchange between them.
- Created, alongside with his other teammates, diagrams such as:
  - Use Case
  - Activity diagrams
  - Class Diagrams
  - User Stories
  - Dataflow Diagram
  - Sequence Diagram
- Drew a system overview diagram where it shows a simplified view of the system's inner workings.
- QR code design using existing Python 3 modules.
- Pseudocode for a centralised system where the server is the one that controls what the raspberry pi is going to show and whether the user is scanning a valid QR code.
- Generating a token that represents a transaction id using Python 3 modules for encrypting and encoding. Given that the main idea was to make the server the master of the system, for performance purposes, these codes were translated to JavaScript, the language that the API is built on.
- Connecting raspberry pi with database using available Python 3 modules for performing requests.
- Research on the effects of a paper receipt in the environment.
- Research on technologies and tools for frontend and backend in Python 3.



- Report on QR code generator based on the Standard ISO 18004 and how to generate them using Python 3 and the tests to find the most optimal configuration.
- Report on Database design that outlines the requirements, ER diagram, relational model, normalisation, data dictionary, and MySQL create statements.
- Report on encryption and encoding in Python 3 for creating a distinct token for the transaction id.
- Exploratory testing on how the raspberry pi application works from a user point of view.
- Automated testing, unit and integrated, in order to assert the main functionality of the raspberry pi application.
- Contributed with the documentation of the project by providing reports on the system analysis and requirements, logical and visual design, development and implementation, and testing of the raspberry pi application.
- Built utility classes in JavaScript to perform communication with the pi app form the API and to generate a unique token per transaction.
- Development of the raspberry pi application
- Working along with the team to generate the graphs in the mobile application.
- Working along with the team to format the final document of the system.

Validation in the server side of an EcoExT QR Code (by Miguelantonio)



## **Asmer Bracho**

Front end Developer

---

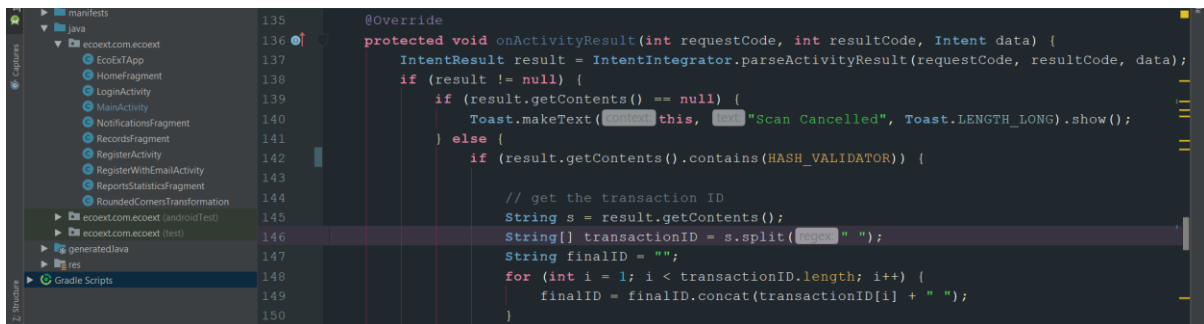
In charge of the front-end development for the application, Asmer has worked with Android Studio and developing the User Interface and User Experience design. Responsible for setting up the environment and dependencies with third parties API's, in order to have a fully working application, as well as, implementation and testing of EcoExT API making sure the UI/UX meet the requirements previously designed.

---

### **From the beginning up to evaluation**

- Decision making: contributed with the scheme of the database by presenting a first version of an ER Diagram.
- Impeditive factors discussion of the database in relation to the tables how we overcame the issue.
- Integration and understanding of tools such as:
  - Git
  - Docker
  - Android Studio
  - GraphQL
- Decision making: contribution to the design of wireframes for the Android Application.
- Designing of 3 different version of EcoExT logo for the Application and System in general and deciding the final version of it.
- Contributed with diagrams such as Use Cases diagrams, Activity diagrams, Class diagrams and so on.
- Development of Front-End Android application as follow:
  - Integration of dependencies for Google, Facebook and Firebase as Authentication means for the Login Section.
  - Login to the application with Google Account.
  - Development of layouts such as Login, Registration, Home, Adding and deleting purses, Sidebar, QR Code Scanning, Adding and deleting transactions, View statistics and all the front and visual implementation that follows the application.
  - Interaction among the menu items and their specifics fragments within the application.
  - Integration of external libraries for designing aspects of the application, such as backgrounds, round corner in pictures and so on.
  - Token Validation for QR Code Scanner to ensure the scanning of it just applies to the users of EcoExT.

- Development of the system and connection the UI/UX to the API in GraphQL by meanings of providers such Apollo compliant client.
- Implementation of navigation menu and sidebar.
- Graph and Visual Report implementation.
- Implementation of filters and retrieving the right information when these are applied.
- Carried out research about front end technologies to be used as part of the project, and possible outcomes, to get the consensus of switching from web design application to native Android Application.
- Written a report of how the implementation of the system was carried out and the problems encountered along the development process and working with the team in order to have a fully working system, reporting the work done to the conclusion of the documentation.
- Decision making: contributed to the overview of the system design.
- Team work: responsible for formatting and designing of the individual contribution section along with the team manager to ensure the quality of the documentation.
- Responsible for ensuring that the bugs found on the testing scenarios were fixed and if not, planning to do in a future version of the application.
- Implementing new features for the application to meet the product requirements.
- Testing and evaluating the testing scenarios to ensure the application does not crash.



*EcoExT Front-end Development Project, where it can be seen:*

*On the left some of the structures and classes implemented up till now.*

*On the right a client-side validation for the QR Scanner that allow user been able to scan only QR generated by EcoExT Raspberry Pii (This is a double side validation, having back-end validation thought API with GraphQL implemented as well) (by Asmer).*



## **Eduardo Firino**

---

In charge of the design of the system business rules and ideas, focused on researching all the impeditive factors and the information needed for the project to carry on. Acting as a wild card, assist the members in different types of tasks, such as database designing and infield research. Responsible for designing the class diagrams and also sharing ideas on how to design the main diagrams required for the application.

---

### **From Beginning up to the evaluation:**

- Researching on how similar applications apply user experiences in order to visualize a better design to EcoExT.
- Research of paper receipt and printer providers.
- Decision making: Graphical user interface wireframes for the mobile app.
- Decision making: Which technologies to be used in the whole project, such as task management tools like Trello, programming languages like Java for Android, etc.
- Decision making: Definition of business rules, such as how the validation is going to take place when the user wants to make a payment using more than one purse/account.
- Decision making: Definition of features for the app for now and also for the future.
- Development of questionnaire for possible users and managers in a survey format.
- Partial development of system diagrams such as Use Case, Activity Diagrams, Entity-relationship model and Class Diagrams of android and Raspberry Pi.
- Defining the features that will be implemented in future versions of the application.