



Appichetto: share easily your expense

Human Computer Interaction
Prof. Andrew D. Bagdanov









Giuseppe Palazzolo
Federico Vaccaro

Introduction: Key idea

Why Appichetto?

Appichetto's idea was born from a daily problem we ourselves and our friends have encountered: the division of expenses. An android application was developed such that:

- Allow to import products / entire receipts as easily as possible;
- Split expenses without wasting time with alternative methods e.g. pen and paper, spreadsheets;
- Keep track of all debts and credits;

Your status 		
	Giuseppe Palazzolo	Debt: -0.00€ Credit: +0.00€ Total = 0.00€
	Maria Manieri	Debt: -9.95€ Credit: +3.40€ Total = -6.55€
	Pasquale Vaccaro	Debt: -252.99€ Credit: +264.85€ Total = 11.86€
	Francesco Palmieri	Debt: -97.16€ Credit: +60.72€ Total = -36.44€
<div> Status  Import ticket  Profile</div>		

Introduction: How we proceeded

The development of our application has been divided in three core phases:

1. **Needfinding:** we studied the needs of our users that had to be fulfilled; we created two Personas from our research, who could represent our archetypal users;
2. **Design of the functionalities:** starting from our two personas' context we identify and develop what functionalities the application has to offer;
3. **Usability testing:** we distributed the final application and assessed the quality of the user experience;



Needfinding

Interview

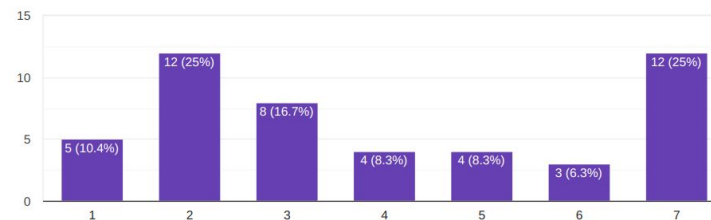
For the definition of the Personas we conducted some informal interviews with friends and colleagues, mainly off-site students; we then deepened our interviews through a **Google survey**, in order to have a clearer overview. In total, about **sixty** interviews were conducted.

The survey was divided in three step, each about:

1. *Demography*: the participants were mostly between 20-30 years old, living with friends, partner or parents;
2. Shopping (typology, frequency): most respondents go shopping more than once a week and in particular:

Se faccio la spesa, compro prodotti anche per i miei coinquilini

48 responses



3. *Split expenses*: most people use pen and paper, someone spreadsheets, other simply keep in mind. Overall many people not satisfied with their current method;

Needfinding - Personas

Francesco



He is a University **off-site student**. He goes often to the grocery store and **buys products also for his housemates**. He has not an income, he can't offers each time the meal, so he has to split the expense with the others.

Scenario

Francesco is going to the grocery store for buying the ingredients for making the Sunday launch. When he gets home he doesn't feel like doing the math and splitting expenses so you'll have to postpone it till next time. The desk and pockets were piled with receipts and maybe someone got lost. During the week happens that Francesco or his housemates ask each other if and why they owes money to someone but the answers are vague.



Needfinding - Personas

Emanuele



He is a software programmer and lives by himself. Emanuele split some services with his friends that live in other cities. He hates spending the time to do boring stuff like tracking his expense. Sometimes he tends to be confusing and forget if he gave the money to his friends, or if some of his friend paid for the services that is splitting with him.

Scenario

He is subscribed to services like Netflix and Spotify, which **shares with** two group of friend. It's also graduation time, he and his colleagues will buy a gift for one of them. To take trace of all of these expenses currently he's using a spreadsheet: while it helps, it still requires focus, so it is still error prone. Every time the spreadsheet is updated, he'll have to write to every friend to remind them of the debt. Also, it happened that he had some discussion with a friend for being too much late with the payment, and would like to avoid discussions like these in the future.



Design of the functionalities

What does the user need?

From the study of the scenarios and the needs of the personas has emerged some functionalities that they needs mainly:

- **Importing a receipt:** this action must be as fast and easy as possible. This is the action that the user procrastinates, from here the problems are created. Some supermarket provides a receipt in pdf format, the app should be capable of parsing this document or scan the items from a photo;
- **Costs computing:** the user select who participates to buy a product then the app do the rest;
- **Debits and credits summary:** the user should be able to easily check the balance with a friend, and understand how it is originated. An user should be able to pay off his debt or deleting the debt from a friend;
- **Social functions:** the user should be able to add new friend and also should be notified when there is some change about his status;

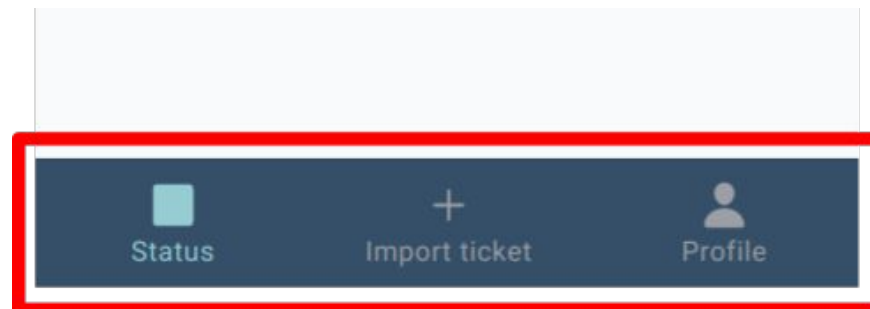


Design and structure of the application

The definition of the features, the average age of our target users and the potential that the smartphone offers led us to develop an **Android application**. This facilitates distribution, is portable and has been implemented with an interface with which the user is used to interact through the use of the **Material Design**.

We decided to structure the application in three macro modules, accessible via **tabs** pattern:

- **Status**
- **Import ticket**
- **Profile**



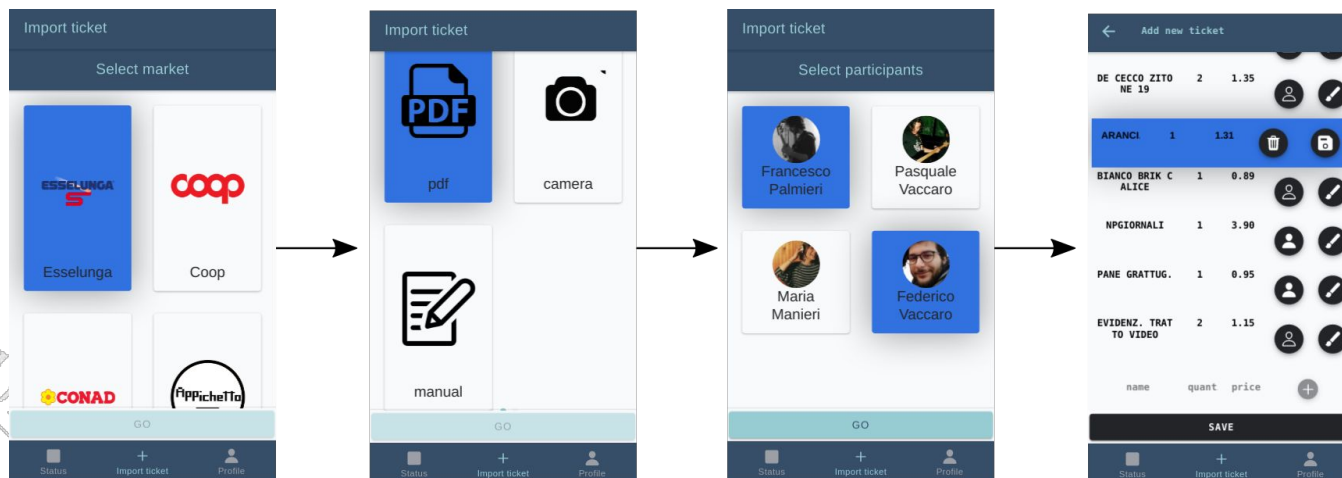
Structure of the application

Import ticket

The central entity of the app is the **Ticket**. It represents a group of one or more products to share with the friends. Importing a ticket is a three-step procedure:

1. **Select market:** this allows the correct parsing of a photo or pdf. Among the choices there is also "Generic" type;
2. **Select method;**
3. **Select participants:** who participates in the purchase of these products?

Finally, the user specifies the participants for each product and save the ticket.

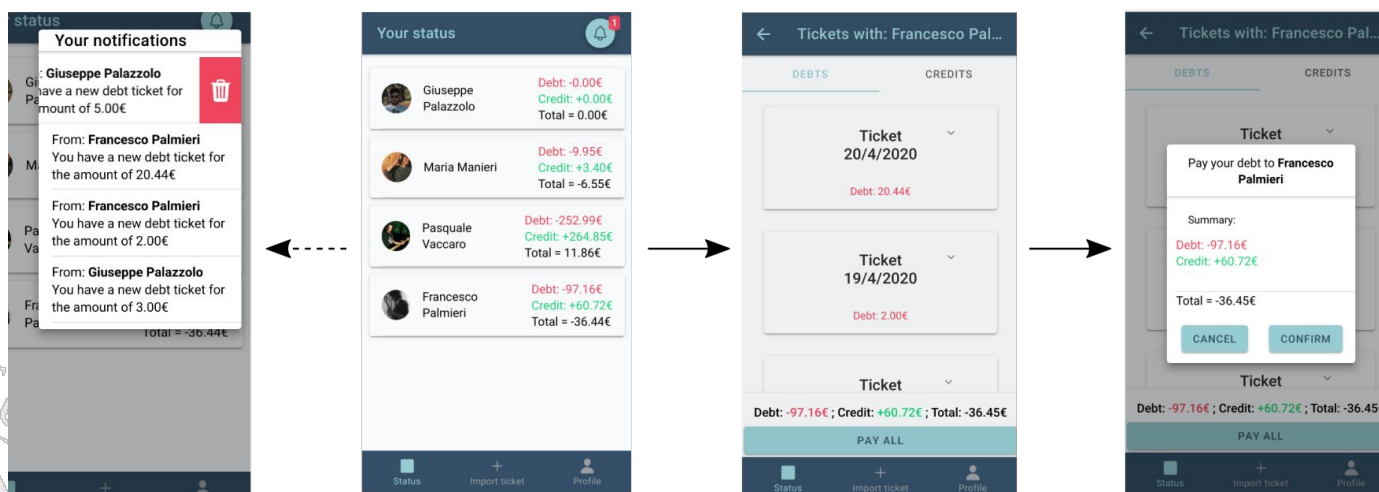


Structure of the application

Status

In the Status view is shown a recap about each user's **accountings**. With a system of *cards* and *slides* have been represented the sections that the user can access to review and mark as paid his **debts/credits**.

Any action of payment or importing of a ticket will be notified to the receiving user through a red number in the **notification** button, clicking on which will show what's new.

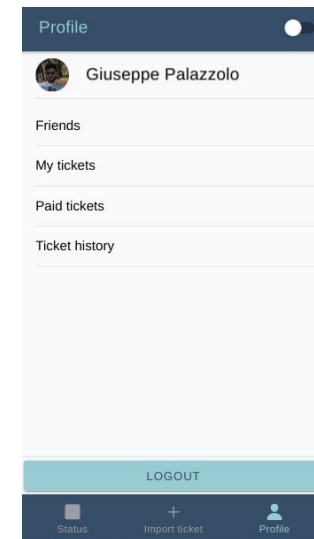


Structure of the application

Profile

In the Profile view, the user can performs generic actions such as:

- Add and remove friend;
- Logout;
- Review his ticket;
- Review his paid ticket;
- Review the tickets that he bought;
- Change theme to dark/light through toggle;



App development

Client

The client side was developed in Typescript using the **Ionic 5.0** framework for Angular. Ionic has provided us with most of the components based on Material Design e.g. card, button, fab button, footer, header, etc.



Cloud DB

All the API are called from the client, to develop server side we use **Firebase**, that provided us with a noSQL real time database.



Usability testing

After the development of the application, we compiled the **apk** file, installable on Android devices and distributed it to some friends and people who participated to our survey that reflected our **target audience**. Sometimes they tried the application from a PC browser and streamed to us their view.

We asked to perform 7 actions:

1. Add a friend;
2. Add a new receipt;
3. Add/update/delete elements in the receipt;
4. Check the balance with a friend;
5. Inspect the tickets shared with a friend;
6. Make the payment of a debt to a friend;
7. Inspect the paid tickets;



Usability testing

Answer

Finally we asked them to fill out 15 questions, consisting of a discrete number spanning from 1, if the tester strongly disagree with the statement, to 7 otherwise and we left one last optional open-answer question for a free thought or suggestion about the application.

N	Question	μ	σ
1	The application is visually pleasant	6.07	0.95
2	The meaning of the buttons and the elements of a page is clear	6.15	0.80
3	The application results smooth and fluid	6.07	0.75
4	The application results hard to use	1.92	0.64
5	Inserting a new ticket is challanging	1.46	0.77
6	It is easy to understand the functionalies of the application	6.23	0.83
7	It is hard to understand the origin of a debt from a friend	1.69	0.63
8	The positioning of the buttons is wrong and illogical	1.61	0.87
9	Overall, the system is enjoyable	6.38	0.76
10	The application has every functionality I need	6.00	1.08
11	I learned to use better the application after a while	4.69	1.10
12	I ignored the notifications	1.76	1.23
13	It is easy to add or remove friends	6.53	0.66
14	It is easy to pay a debt	6.46	0.77
15	I am overall satisfied	6.15	0.68

Conclusion, user suggestion and further developments

Thanks to the phases and principles of the **HCI**: needfinding, development and usability testing, we managed to produce an user interface that felt natural and friendly.

Our testers were after all satisfied with the application, as the results of the tests tell.

Some of them gave us some suggestions like:

- Scheduled ticket insertion, to use for services such as Netflix, Spotify, etc.;
- Choice of currency to use, to use on vacation;
- Inserting a *group* feature where each *circular debt* would be automatically solved;

We also have in mind some improvements to implement:

- Integrate a payment system like PayPal;
- Integrate a push notifications system in addition to the current notification system;
- Improving the OCR algorithm;





Download and use Appichetto

