Walletize

Dario Carolla mat. 807547 [1], Federico Manenti mat. 790032 [2]
Project supervisor: Professor Fabio Mercorio

Abstract
Digital payments evolution is in continually increasing. In the last year, it has reached its peak. Is it possible to simplify more the digital payments method? The idea behind Walletize is to facilitate the exchange of information between users. This enables easy and practical access to the information required for the transactions of money. The development of Walletize is based on the use of a NoSQL database to employ its potential. Also, APIs have been implemented to allow third-party applications to use Walletize services.

Keywords
Big Data in Public and Social Services – Bank System – IBAN shortner

[1,2] *Department of Informatics, Systems and Communication, Data Science, Università degli Studi di Milano-Bicocca*
1 - d.carolla@campus.unimib.it
2 - f.manenti3@campus.unimib.it

Contents

Introduction

Walletize idea was born, for necessity, during the Covid-19 emergency: a lot of charities and hospitals needed large amounts of money to face the ongoing crisis. [1] To solve these needs, many donation campaigns have been advertised, most of which required the use of bank transfers which, to date, remain one of the main digital payment methods. To be done the bank transfer needs several data, the most important is the IBAN: a long code that uniquely identifies the beneficiary's bank account. Because of its length, the communication of the IBAN became an extremely tough task. For these reasons, the idea of an IBAN shortener dedicated exclusively to charities was born. The shortener would consist of associating a unique Token to each charity campaign that contains all the information necessary to donate. So the donator, within own home banking, would have inserted only the Token relating to the chosen charity campaign. Therefore the banks, thanks to the APIs, would be able to obtain useful information for the donation. During the feasibility study, the biggest problems were two. First was the accreditation of charities: it's very difficult to ensure the real existence of the latter avoiding becoming a vehicle of scams. The second concerns the use of APIs by banks, as they would hardly change their system and this would make the application completely useless.
The first problem could be solved by asking charities for different documents during registration, including:

- Statute of the association;
- Constitutive Act;
- Minutes of the board appointing the President;
- President's identity document.

In this way, one could attest to the existence of the association and identify the legal representative. The second problem, on the other hand, depending on the banks, cannot be solved.
For these reasons, Walletize has evolved into a system for collecting and sharing different digital payment methods no longer dedicated to charities, but usable by private individuals.
The idea behind it is very similar to the previous one:

each user has a unique Token to which the different payment methods entered by him correspond. Every user will be able to share, one or all, his payment methods using a simple Token or a link. In the same way, the users will have the option to search via a graphical interface another user's data, employing his Token or the link previously shared.
Therefore Walletize allows users to simplify the communication of their payment methods summarizing all the information within a univocal Token. Moreover, the data provided by users can be integrated into third-party application systems through the use of APIs.

1. State-of-the-art

A critical step for the development of an idea is the *state-of-the-art study*: the search for similar, or even equal, services that try to solve the same problems. Indeed it is important, if a similar application exists, to understand the differences and analyze the strengths and weaknesses of each app. After various research, two applications similar to Walletize were found. The main characteristics are shown in the table 1.

Table 1. State of the art

Feature \ Service	IBAN.run	IBAN.im	Walletize
Different payment methods	×	×	✓
Storage	×	✓	✓
Shareability	✓	✓	✓
UX	✓	×	✓
API	×	×	✓

IBAN.run [2] is a simple and intuitive shortener that allows transforming an IBAN into a shareable link, without requiring registration. Therefore this website allows sharing only one payment method. Moreover, there isn't a database for storing data and either APIs for exposing the service.
Instead, *IBAN.im* [3] is a GitHub project still under development. The ultimate goal is a digital wallet where each registered user can store several IBANs and share them with other subscribed users. Even in this case, the only accepted payment method is the IBAN and there aren't usable APIs. It is better than IBAN.run due to the presence of a database for storing the data, but it is worse concerning the *user experience/interface* because the website isn't simple and great impact like IBAN.run.
Walletize grants the storage of different payment methods (to date IBAN, PayPal, and Satispay) to registered users and the sharing with anyone. Moreover, another big difference is the implementation of APIs which can be used in the future by third-party-services. The result of the state-of-the-art study shows that there is no equal or very similar competitor to Walletize that tries to answer the same problems.

2. Architecture

Walletize architecture is based on the use of *Flask* [4], a micro web framework written in Python. As can be seen in figure 1, the architecture is divided into four main blocks:

- Flask
- Storage
- Front-end
- API

The first one, as mentioned, is Flask. It is the back-end of the application, through which all the interactions with the other blocks are managed. The second one is data storage. For it is used two databases, one relational and another one document-based. The front-end, instead, allow the users to interface with the application by GUI created using the traditional *HTML*, *CSS* and *Javascript* languages. Finally, the last block is constituted by APIs which, through HTTP requests, allow to have access to the functionality of the application and in particular to the contents of that.

2.1 Storage

As previously mentioned, data storage has been managed using two separate databases, one relational and the other document-based. In particular, for the first one was used *SQLAlchemy* [5] a Python SQL toolkit. The relational database consists of an individual table. It was employed to manage the information provided by users during registration: *username*, *email*, *password* and the *Token*. The relational database design is shown in table 2.

Table 2. User table

Name	Type	Constrain
Id	Integer	NN, CP
User	Varchar	U
Email	Varchar	U
Password	Varchar	
Token	Varchar	U

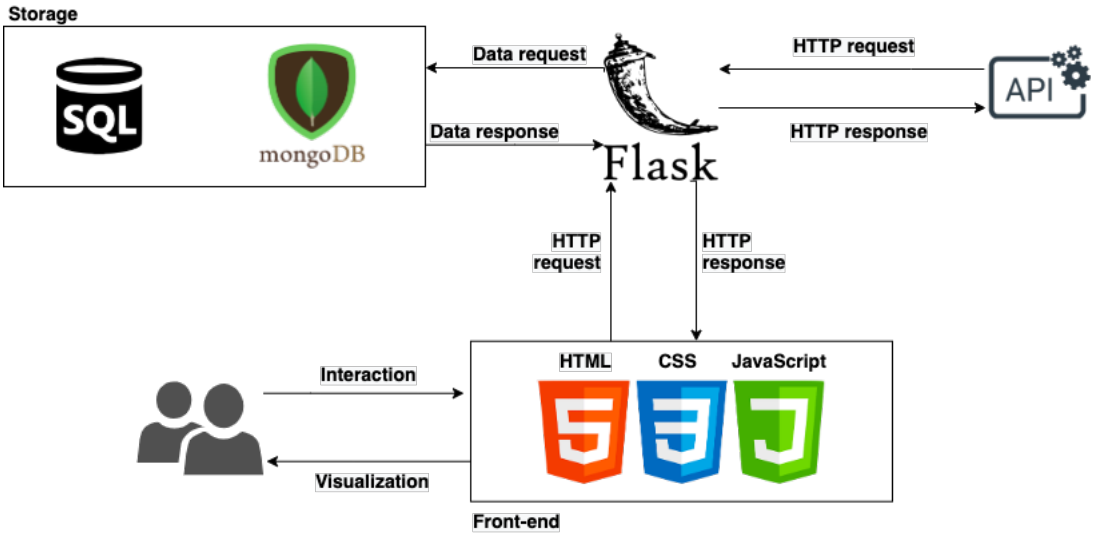

Figure 1. App Architecture

In the *user* table, the id of a user is the primary key. For the columns *email* and *Token*, instead, has been used the *UNIQUE* constraint, which ensures that all values in a column are different. The user password, for higher security, is additionally encrypted through the *SHA256* hash algorithm before being stored in the database.
The second database is MongoDB [6]. It is a document-based database with powerful scalability and flexibility. It is employed to manage the payment methods data provided by the users. The payment methods that can be inserted are IBAN, PayPal and Satispay. These three methods require very different information, in particular:

- IBAN: payee, the purpose of the payment and the IBAN;
- PayPal: PayPal link;
- Satispay: mobile number.

For this reason, it was employed the *schemaless* property of NoSQL databases. The database is composed of a single collection where each document denotes a user. Below is represented a structure of a document in the collection:

```
{
    _id: 1
    user: "user_name"
    token: "user_token"
    metodi: Array
}
```

Figure 2. User document

User and *Token* fields, of course, is the same contained in the relational database. The field *metodi*, rather, is a document array that includes the payment methods created by the user that have the following structures:

```
{
    _id: 0
    type: "IBAN"
    beneficiario: "payee"
    causale: "payment_reason"
    IBAN: "IBAN_numeber"
}
```

Figure 3. IBAN

```
{
    _id: 1
    type: "PayPal"
    link: "PayPal_link"
}
```

Figure 4. PayPal

```
{
    _id: 2
    type: "Satispay"
    cellulare: "mobile_number"
}
```

Figure 5. Satispay

The fields *_id* are used to uniquely identify documents in the array. The field *type*, rather, represents the payment method.

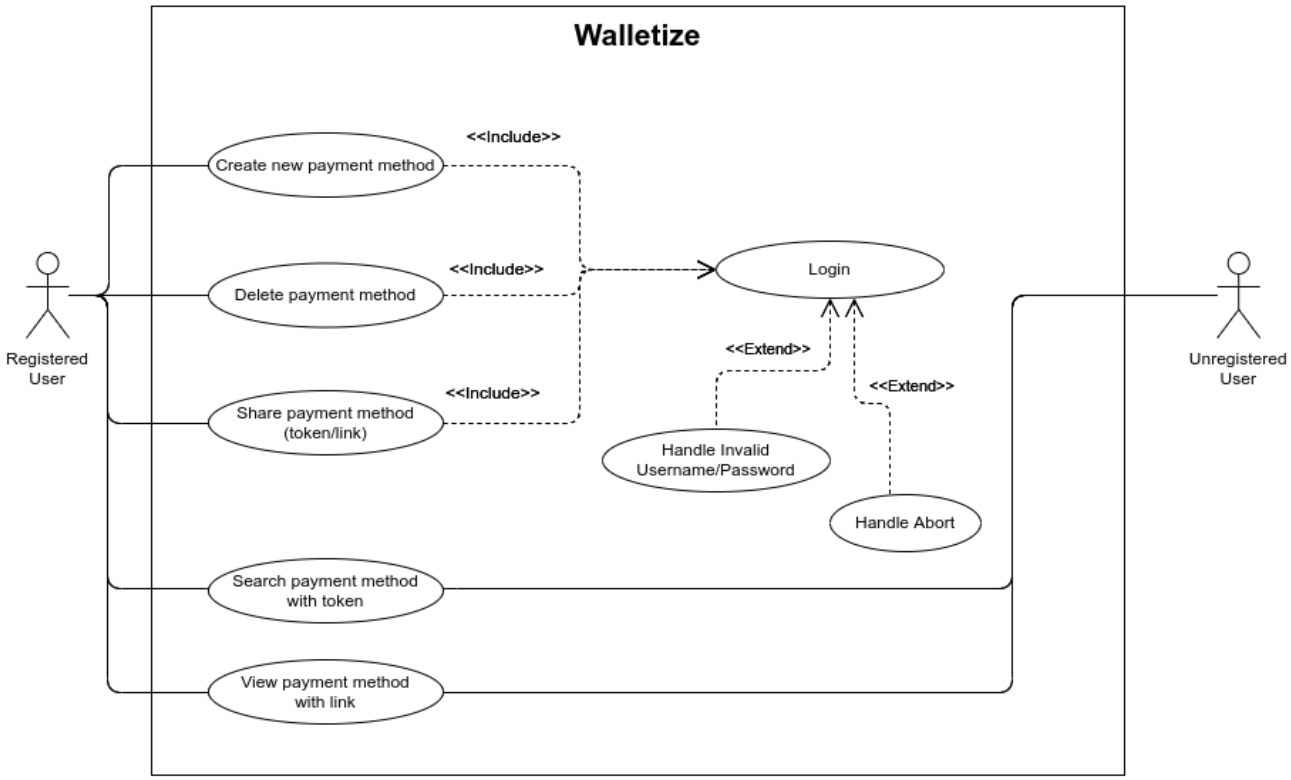

Figure 6. Use Case Diagram

2.2 API

The web APIs are frequently used to provide an interface for access to data to web sites and application client. Web APIs use the HTTP protocol to provide CRUD operations (Create, Read, Update and Delete). These operations can be performed on a resource to which corresponds a URL (Uniform Resource Locator) that uniquely identifies it. For the Walletize project, specific APIs have been created. With these, through HTTP requests, is possible to obtain the data stored into the MongoDB database. Table 3 presents and describes the URIs needed to use the APIs:

Table 3. API description

Type	URL path	Description
GET	/api/all_token	Get all Tokens
GET	/api/random_user	Get all information of a random user
GET	/api/payment_method/<token>	Get all payment methods for the user with <token>
GET	/api/payment_method/<token>/<type>	Get all payment methods <type>for the user with <token>

To use APIs it is required to be registered, because, to make a request it is required a token with the following format: *user_name:user_password*.

3. Use case

Walletize is designed to facilitate the sharing of different payment methods among private individuals. The actors involved are two: registered users whose purpose is to make their information is known and the users, both registered and non-registered, that want to find this information.
A new user who wishes to share their payment methods must register to the website by providing a username, a mail, a univocal Token (used for sharing), and a password to log in.
The picture 6 represents the use case diagram for the two actors involved.
A registered user, after a correct login into the platform, can:

- Create a new payment method (IBAN, PayPal, or Satispay). It is also possible to create more than one per type;
- Delete an old payment method previously added;
- Share all its payment methods through the Token or a link;
- Share one payment method through a link.

Furthermore, a user, whether he is registered or not, can:

- Search into the website the payment information of another user, thank the personal Token;
- With a previously shared link find one or all payment methods of another user.

4. Conclusions and future developments

The use of Walletize will allow an easy exchange of information between different users and eliminates the difficulty of communicating an IBAN. It will also make easier the sharing of different payment methods in case the user wants to provide more option for an exchange of money. Moreover, thanks to the use of MongoDB and its scalability, it will not be difficult to add new payment methods in the future. At the same time, through the use of APIs, it will be possible to integrate into a third-party app the usage of the Token as a way of communicating bank data or other payment methods. In conclusion, Walletize potential is manifold.
Possible future developments are different. The first one is adding other payment methods. Also, the possibility to modify a payment method can be added. Finally, it may be necessary to create a system to tokenize the individual payment methods, which are currently only sharable via links.

References

[1] *Italy donation for Covid-19*. https://italianonprofit.it/aiuti-coronavirus/dataviz/.

[2] *iban.run*. https://iban.run/.

[3] *iban.im*. https://iban.im/.

[4] *Flask Documentation*. https://flask.palletsprojects.com/en/1.1.x/.

[5] *SQLAlchemy*. https://www.sqlalchemy.org/.

[6] *MongoDB*. https://www.mongodb.com/.