IPL
escola superior de tecnologia e gestão
instituto politécnico de leiria

Distributed Applications Development

Computer Engineering	3rd Year	1st Semester	2019-20	Periodic Evaluation
Project	Limit date for results divulgation: 20 Jan 2020			
Date: 31 Oct 2019	**Delivery Date: 6 Jan 2020**			

Project – Virtual Wallet

1 OBJECTIVE

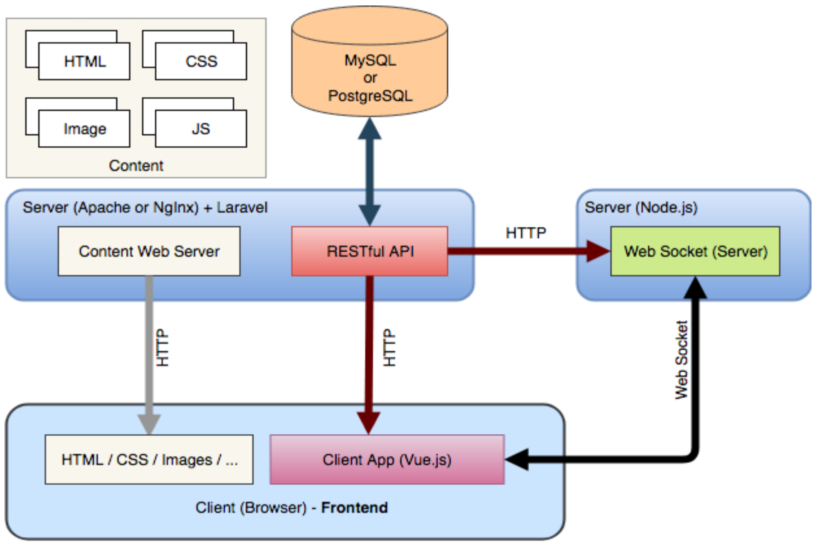
This project's objective is to implement a Single Page Application (SPA) for the "Virtual Wallet" platform, using Vue.js framework for the development of the web client.

2 SCENARIO

The main purpose of the "Virtual Wallet" platform is to provide a Web Application that replaces a physical wallet, by keeping virtual money and handling small transaction between platform users or external entities. The main concept of the platform is the virtual wallet, that behaves like a financial account, with a balance (amount of money currently on the wallet – it must be positive or zero) and multiple financial movements (or transactions), each representing either an income (a credit) or an expense (a debit). Income movements are created by payments from the users to the platform or through transfers from other virtual wallets. Expense movements correspond to payments to external entities or transfers to other virtual wallets. Each movement (either income or expense) is limited to a maximum of 5,000.00 € and increases or decreases the wallet balance – an income (credit) increases the balance and an expense (debit) decreases the balance. When a user transfer money from his virtual wallet to another virtual wallet, the platform creates two financial movements: an expense (debit – balance decreases) on the user wallet and an income (credit – balance increases) on the other virtual wallet.

3 ARCHITECTURE AND TECHNOLOGIES

The Web platform must use the Single Page Application (SPA) paradigm. It includes a Web client application developed with Vue.js framework, and on the server side, a RESTful API implemented with Laravel and a Web Socket server implemented with a Node.js server and all appropriate libraries. Static content delivery (HTML, JavaScript, CSS, images, etc.) is provided by Laravel and the main relational database must be either MySQL or PostgreSQL.



It is also possible to use other libraries, technologies or databases that complement the described architecture. For instance, “NoSQL” databases or cache technologies (e.g. MongoDB, Redis, Memcached) can be used to optimize the performance of storage operations.

4 BUSINESS MODEL

The business model of the "Virtual Wallet" platform follows the main principles described on the scenery section but adapted to the Web application to be implemented through this project. The main constraint of the Web application, when compared to an ideal scenery, is that it does not implement real financial transaction with external entities, which mean that both the payments of the users to the platform and payments to external entities are only registered on the platform without creating real financial transactions. Any financial movement registered on the platform simulates a financial transaction (although no real world transactions are made) which means that after a movement is created it cannot be removed or changed (at least not the critical financial data of the movement – secondary information within the movement might be changed).

1.1 MOVEMENTS

The payment of the users to the platform will be registered by human operators (Virtual Wallet employees) as an income (credit) movement of the virtual wallet that will receive the money.

During the registration of these payments, the operator specifies the destination (the e-mail of the virtual wallet that will receive the money), the value (amount of money), the type of payment (cash or bank transfer), the IBAN (when the type of payment is a bank transfer) and a description ("source description") for the payment.

The payment to external entities will be registered as an expense (debit) movement by the user. All expense movements will be either a payment to an external entity or a transfer to another virtual wallet within the platform. The registration of both cases includes the type of movement (payment to external entity or transfer), the value (amount of money), the category of expense (e.g. groceries, transport, clothes, etc.) and a description. When the type of movement is a payment to external entity the registration also includes the type of payment (bank transfer or MB payment), the IBAN (for bank transfers) and for MB payments, the MB entity code (5 digits) and the MB payment reference (9 digits). When the type of movement is a transfer, the registration adds the e-mail of the destination (the e-mail of the virtual wallet that will receive the money transferred) as well as a source description.

When a transfer movement is created by an user (as an expense/debit of his wallet), the platform creates a mirrored movement (an income/credit with the same amount of money) on the destination virtual wallet – a transfer always creates two paired movements of the same value, the source movement (expense/debit) and the destination movement (income/credit). The source description added on the transfer movement (the source movement) registration, will be stored on both movements (source and destination movements) – this allows the user that transfers the money to pass some textual information to the user that receives it.

As it can be inferred from the above description, platform users do not register any income movements on their own virtual pocket. Income movements are added only by the operators (when registering a payment from the user to the platform) or as part of a transfer from another virtual pocket. However, platform users can edit any of their movements (including income movements) by adding or changing the category of income (e.g. salary, sale, etc.) or expense or the description (not the same as "source" description). Any other information on the movements is immutable.

Some considerations about the movements:

- Each movement will be identified by an ID (automatic id);
- The date of the movement is filled automatically when the movement is created (with current date and time);
- "Description" of the movement refers to textual information filled by the wallet owner;

- "Source Description" of the movement refers to the textual information filled by the operator, when the movement is a payment to the platform, or by the user that created the transfer movement (both on the source and destination movements).

1.2 BALANCE

Each virtual wallet will have a balance with the amount of money currently on the wallet, that must be positive or zero – the platform does not allow negative balance for the virtual wallets. The balance must be calculated automatically whenever a movement is added (by a user or operator).

Also, each movement must have a start balance and an end balance, with the value of the wallet balance before and after the movement has been applied. For instance, if the start balance is 855 € and the movement is an expense of 230.5 €, then the end balance will be 624.5 €.

Table 1: Example of wallet movements

Virtual Wallet: someones_wallet@mail.pt	**Current Balance:**	**674,50 €**

Movements:

Type	**Category**	**Date**	**Value**	**Start Balance**	**End Balance**
Income	Sale	14-9-2019	**50,00 €**	624,00 €	674,50 €
Expense	Clothes	13-9-2019	**230,50 €**	855,00 €	624,50 €
Expense	Food	13-9-2019	**145,00 €**	1.000,00 €	855,00 €
Income	Salary	12-9-2019	**1.000,00 €**	0,00 €	1.000,00 €

The values of the start balance and end balance of the movements, as well as the balance of the wallets (the current balance) should all be calculated automatically.

1.3 ACCOUNT

A user creates an account on the platform by registering himself on the application. During the application registration the user provides his e-mail (must be unique), name, photo, NIF (fiscal identification) and a password for authorization on the platform. When the user's account is created it will be automatically associated with a virtual wallet that has a balance value of zero.

The accounts of the platform operators and administrators are not associated with any virtual wallet and are created exclusively by an administrator. These accounts will only have a name, a photo, a password and an e-mail, which must be unique among all accounts of the platform (including the platform users). Administrators may also remove any operator or administrator account, except his own account. Also, platform users (users that own a virtual wallet) accounts can be deactivated and reactivated by any administrator. An account can only be deactivated if it has a virtual wallet with a balance value of zero.

1.4 STATISTIC INFORMATION

The application should provide to the platform users and administrators, statistical information visually (with graphs) and/or textually (with tables). Platform users can only access financial information about their own wallets. Administrators can access global platform usage information or global (of all users) financial information – they cannot access statistical information about individual wallets (only global information is available to the administrators).

Students are responsible for the type and quality of information extracted from the platform and the way it is presented to the users.

Just as a reference, here are some examples of information that can be extracted from the platform to be presented to the users: total expenses; total expenses through time – where the temporal dimension (years, months, weeks or days) corresponds to a graph axis; expenses by category; expenses by category through time; total income; total income through time; income by category; income by category through time; balance through time; etc.

Also, some examples of information to be presented to the administrators: number of active platform users; total accesses to the platform; total accesses through time; total movements (transactions); total movements through time; total amount of money in the platform; total amount of money transacted (moved) by platform users through time; total external income through time; total internal transfers through time; etc.

5 FUNCTIONAL REQUIREMENTS

Project's functional requirements will be presented as user stories, organized by types of users.

<u>ALL USERS</u>

US 1. As a user (anonymous or authenticated user) I want to access the application's initial page with a welcome message and information about the total number of virtual wallets;

ANONYMOUS USERS

US 2. As an anonymous user I want to create an account with an associated virtual wallet by registering myself on the platform. Registration data should include user’s name (only spaces and letters), e-mail (must be unique), password (3 or more characters), NIF (fiscal identification) and an optional photo (by uploading an image). When the user's account is created it will be automatically associated with a virtual wallet with the balance value of zero.

US 3. As an anonymous user I want to authenticate the application with valid credentials (email and password). All types of users (platform users, operators and administrators) use the same authentication process.

AUTHENTICATED USERS

US 4. As an authenticated user (platform user, operator or administrator) I want to be able to logout from the application. After logging out, I should be redirected to the application's initial page (US 1) and should not be allowed to access any other content of the application.

US 5. As an authenticated user (platform user, operator or administrator) I want to be able to change my personal account information, namely: the name (only spaces and letters); photo (upload a JPG file) and password (3 or more characters). When changing the password always confirm the new password twice and guarantee that the user known the previous password.

Platform users should also be able to change the NIF (fiscal identification with 9 digits). The e-mail of the account is immutable.

OPERATORS

US 6. As a platform operator I want to be able to register an income (credit) movement from the outside of the platform. The registration requires the e-mail of the virtual wallet that will receive the money; the value to credit on the wallet (from 0,01€ up to 5000,00€); the type of payment (cash or bank transfer); the source description and the IBAN (2 capital letters followed by 23 digits) - when the type of payment is a bank transfer.

The current balance of the virtual wallet that receives the money must be updated automatically, as well as the start balance and end balance of the movement itself.

PLATFORM USER

US 7. As a platform user (virtual wallet owner) I want to be able to view information about my virtual wallet, namely, the wallet's current balance and the wallet's movements (expenses and incomes). The movements are always sorted by date (recent movements appear first) and presented on a table with pagination. The table must include the following columns (not necessarily by this order): the ID of the movement; type (expense or income); transfer e-mail (the source or destination e-mail – only if the movement is part of a transfer between 2 virtual wallets); type of payment (only if movement is not a transfer); category; date; start balance; end balance and value.

The remaining information about the movement, namely: description; source description; IBAN; MB entity code; MB payment reference and photo (only for a transfer - photo of the user associated to the pair movement of the transfer), should be easily accessible from the movement row, either on one or more columns of the table or through a quick interaction from the user (e.g. a window or panel popup after a mouse click).

US 8. As a platform user I want to be able to filter my virtual wallet movements by ID, type (expense or income), date interval, category, type of payment or by transfer e-mail (source or destination e-mail of a transfer). When combining multiple filter criteria (e.g. type, category and date interval) I want to apply the logical operator "and" between all filter criteria (e.g. type is "expense" and category is "groceries" and date is "between 1/1/2018 and 31/12/1018").

US 9. As a platform user I want to be able to register an expense (debit) movement of my virtual wallet. The movement requires the type of movement (payment to external entity or transfer); the value (from 0,01€ up to 5000,00€); the category of expense and a description.

If the type of movement is a payment to an external entity, the registration must also include the type of payment (bank transfer or MB payment). When the payment uses a bank transfer the registration also requires the IBAN (2 capital letters followed by 23 digits). For MB payments the registration also requires an MB entity code (5 digits) and the MB payment reference (9 digits).

If the type of movement is a transfer, the registration must also include the e-mail of the destination wallet and a source description – application must guarantee that the destination e-mail is associated to a valid virtual wallet from another user.

The current balance of the associated virtual wallet, as well as the start balance and end balance of the movement, must be updated automatically.

US 10. As a platform user, when a transfer movement is created (US 9), I want the application to automatically create the mirrored (pair) movement (an income) on the destination wallet. The data on the mirrored movement (destination movement) must reflect the data on the source movement –field values are mostly the same, except the type (source = expense; destination = income), the description and category (both are empty on the destination movement). Also, the application must add to both movements (source and destination) a reference to the other one.

The current balance of the virtual wallet associated to the destination movement, as well as the start balance and end balance of the destination movement, must be updated automatically (as should happen with the source movement – US 9).

US 11. As a platform user I want to edit any movement (expense or income) of my virtual wallet. When editing a movement, I can only change the category and the description. All the remaining information of the movement is immutable.

US 12. As a platform user, I want to be notified when an operator adds an income movement to my virtual wallet (US 6) or another user transfers money to my virtual wallet (by creating a transfer movement US 9, US 10). If at that exact moment (when movement was created) I'm using the application, I want to receive the notification instantly within the application itself (using popup messages or similar functionalities). If I'm not using the application, I want to receive an e-mail notification.

US 13. As a platform user, if I'm viewing the information about my virtual wallet (US 7), I want that information (including the wallet's current balance and the wallet's movements) to be automatically and instantly updated when an operator adds a movement to my virtual wallet (US 6) or another user transfers money to my virtual wallet (by creating a transfer movement US 9, US 10).

US 14. As a platform user I want to view statistical information about the movements of my virtual wallet. The type and quality of information extracted from the platform and the way it is presented to the users (with charts, tables or other alternative method) is defined by the students (*it will affect the evaluation grade for this user story*).

<u>ADMINISTRATOR</u>

US 15. As an Administrator of the platform I want to create operator and administration accounts. These accounts will only have a name (only spaces and letters), a photo (upload a JPG file), a password (3 or more characters) and an e-mail, which must be unique among all accounts of the platform (including the platform users).

US 16. As an Administrator of the platform I want to manage all user accounts (platform users, operators and administrators) of the platform. For this I have to access and filter the list of users. The list of users should be presented on a paginated table with the user type (platform user, operator or administrator), name, photo and e-mail (not necessarily by this order). For the platform users, the table must also include the account status (active or inactive) and information about the balance of the associated wallet (either "empty" or "has money"). Please note that the administrator should not know the exact balance of any wallet on the platform – he only knows if the wallet is empty (balance = 0) or not (balance > 0).
User accounts presented on the table can be filtered by the user type, name, e-mail and status (for platform users). The logical operator "and" should be applied when combining multiple filter criteria.

Also, as an administrator I want to remove any operator or administrator account (except my own account) and deactivate or reactivate platform users – an account can only be deactivated if it has a virtual wallet with a balance value of zero.

US 17. As an Administrator of the platform I want to view statistical information about the platform usage information or global financial. The type and quality of information extracted from the platform and the way it is presented to the administrators (with charts, tables or other alternative method) is defined by the students (*it will affect the evaluation grade for this user story*).

6 CONSTRAINTS

Project's **<u>mandatory constraints</u>** are the following:

C 1. Application must use exclusively the Single Page Application (SPA) paradigm.

C 2. Application must use the Vue.js framework for the client-side code.

C 3. Applications' web API must be implemented with Laravel framework.

C 4. Applications' web socket server (if implemented) must be based on Node.js.

C 5. Main relational database must use either MySQL or PostgreSQL.

C 6. Application must be published and accessible on the web through a desktop browser.

C 7. E-mail messages sent by the application (for notifications) must use a real e-mail server (application cannot use mailtrap.io or similar services). This means that e-mail notification can be tested using a real e-mail address.

C 8. The database of the published application must be seeded with data that simulates real life usage – both in size and content. Apply the provided seeder (Laravel seeder) - with the option "1-full" - to fill data on the published database.

7 NON-FUNCTIONAL REQUIREMENTS

Project's non-functional requirements are the following:

NF 1. The client application code and structure must follow the principles and good practices of Vue.js framework.

NF 2. The server-side code and structure of the web API (implemented in Laravel) must follow the principles and good practices of Laravel framework.

NF 3. The web API must follow the principles and good practices of a RESTful service.

NF 4. Visual appearance and layout should be consistent through the entire application and adapted to the applications' objective and usage context.

NF 5. All implemented features must be correctly integrated and work consistently throughout the application.

NF 6. Application should have the best usability possible. This implies that information and available operations are clearly presented to the user, and that user interaction is very simple, uses standard procedures, and requires a minimum of interaction from the user to obtain the required outcomes.

NF 7. Data inputted by the user (any type of user) should always be validated on the client and on the server, according to rules extrapolated from the business model, database and functional requirements.

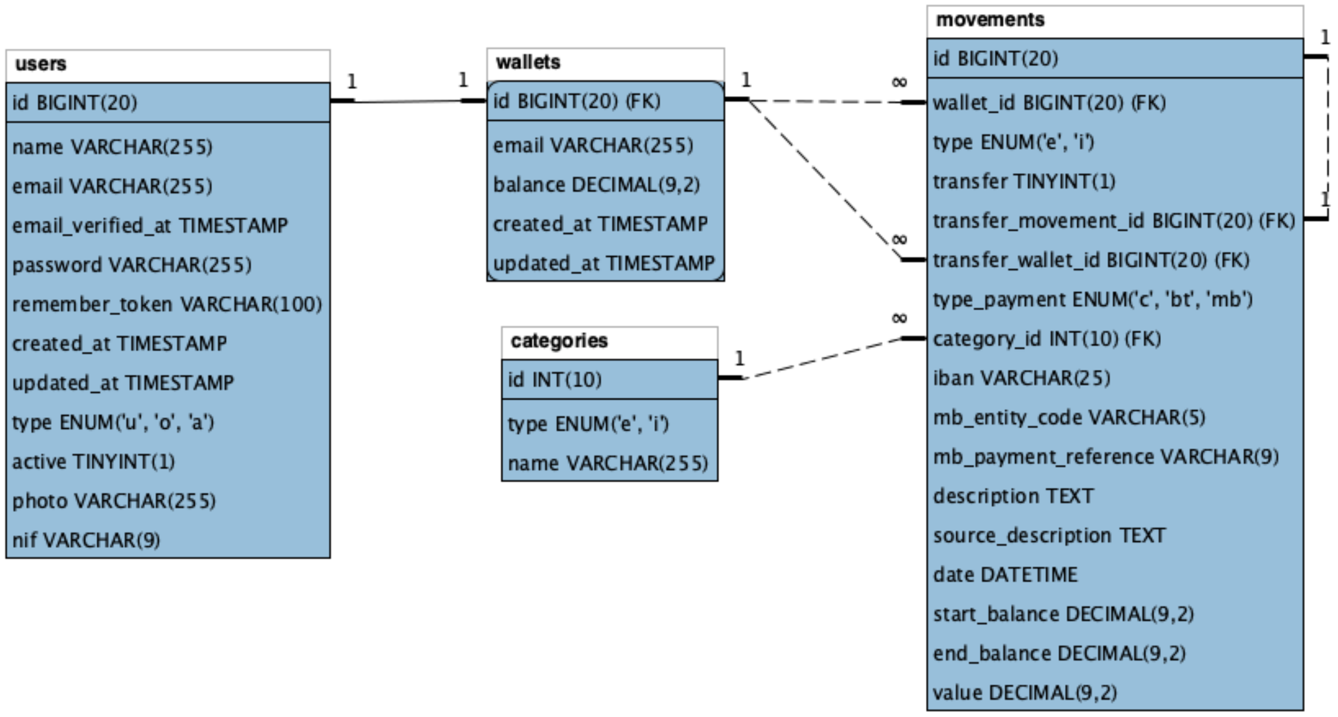
NF 8. Application should be reliable and have an optimized performance. For instance, performance can be improved by applying a mix of techniques to minimize internet bandwidth (reduce the number of HTTP Requests and the size of HTTP Responses) and client memory (limiting the size of datasets on the client) while maximizing the performance of data navigation and filtering on the client (if data is already on the client, navigation and filtering data can be very fast using JavaScript code on the client). Other techniques can be applied to the Vue.js code to improve the performance on the client, and to the Laravel and MySQL code to improve the performance on the server.

NF 9. Application should be as secure as possible and it should guarantee the protection of users' data and privacy, ensuring that no unauthorized user can view or change any data or document it should not have access to. Authentication and authorization must follow the principles and good practices for authentication and authorization of web APIs.

NF 10. Performance of the Web Socket full duplex communication should also be optimized. This implies minimizing the size of the messages and sending messages exclusively to the clients that really require it (avoid broadcasting messages to all clients if they do not need to receive them).

8 DATABASE

The main relational database must use either MySQL or PostgreSQL (it can be complemented by any Non-SQL database, created by the students, if they so decide). The **database structure is provided** (as a Laravel Migration) and cannot be modified. A **database seeder is also provided**, which must be used to preload the database with data that simulates the application usage through several months. The database structure is as follows:



1.5 SUMMARY

Summary of the most important database entities (tables) and attributes (columns):

- **users** – user account – including platform users, operators and administrators
  - **id** (automatic) – identification of the user account;
  - **type** – ("u", "o", "a") type of user – "u" = platform user (owners of the virtual wallets); "o" = operator; "a" = administrator;
  - **active** – indicates whether the user is active or not– only platform users can have an inactive account (active = 0). For operators and administrators, accounts are always active (active = 1) – in these cases accounts can be removed;
  - **name** – full name of the user;
  - **email** – user's e-mail (unique);
  - **password** – password hash;
  - **photo** (optional) – file name of the photo of the user – null if no photo is available;
  - **nif** (optional / mandatory for platform users) – fiscal identification number (9 digits);
- **wallets** – virtual wallet associated to a platform user
  - **id** – internal identification of the virtual wallet (should not be visible to the users) – it is the same as the id of the user that owns the virtual wallet (foreign key that references the user);

    This id is not an automatic id
  - **email** – identifies (publicly) the virtual wallet (must be the same as the email of the associated user account);

    Note: although the email of the wallet could be accessible through the "users" entity, it is duplicated to simplify development, improve performance and due to the fact that users identify wallets by email (and not by id)
  - **balance** – current balance of the virtual wallet;
- **movements** – financial movement associated to a virtual wallet
  - **id** (automatic) – identification of the movement;
  - **wallet_id** – the virtual wallet that the movement belongs to;
  - **type** – ("e" ,"i") type of movement, either an expense (debit) movement ("e") or an income (credit) movement ("i");

- **transfer** – (boolean) indicates whether the movement is a part of a transfer or not. When movement is a transfer (transfer = 1) a pair movement must be created and referenced by the value of transfer_movement_id. Both movements of the transfer (pair movements) must have the transfer value = 1. The source movement is the expense movement, and the destination movement is the income movement.

  When a movement is not a transfer (transfer = 0) it is considered to be a payment. For expense movements, payment refers to a payment to an external entity and for income movements, payment refers to a payment from the outside into the virtual wallet (register by an operator);
- **transfer_movement_id** (optional/mandatory for transfer movements) – the id of the pair movement of a transfer. When the current movement is an expense (source movement), the pair movement will be the destination movement. When the current movement is an income (destination movement), the pair movement will be the source movement;
- **transfer_wallet_id** (optional/mandatory for transfer movements) – the id of the wallet associated to the pair movement of a transfer;

  > *Note: although the wallet_id could be accessible through the "transfer_movement_id", it is duplicated to simplify development and improve performance*
- **type_payment** (optional/mandatory for payments – null for transfers)

  ("c", "bt", "mb") – cash (c), bank transfer (bt), "Multibanco"/MB payment (mb). If type of the movement is a transfer, type of payment should be null. Otherwise, type of payment can be cash (for income movements only), bank transfer (both income or expense) or MB payment (for expense movements only);
- **category_id** (optional) – the category of the expense or income. The value null represents a movement that is not categorized yet;
- **iban** (optional/mandatory when the type of movement is a bank transfer) – the IBAN (2 capital letters followed by 23 digits) of the bank transfer;
- **mb_entity_code** (optional/mandatory for expense movements with type of movement = MB Payment) - the MB entity code (5 digits);
- **mb_payment_reference** (optional/mandatory for expense movements with type of movement = MB Payment) - the MB payment reference (9 digits);

    - **description** (optional) – informal textual description about the current movement, added by the owner of the associated wallet;
    - **source_description** (optional/available for income movements only) – informal textual description about the current movement, added by the operator (for an income payment) or the owner of the wallet associated to the source movement (for an income transfer). When the movement is an expense movement, the value of the source_description is always null. For income movements, value is optional;
    - **date** – date and time when the movement was created;
    - **start_balance** –balance of the associated wallet before the movement is registered;
    - **end_balance** - balance of the associated wallet after the movement is registered;
    - **value** – the value of the movement;
- **categories** – category of a movement (credit or debit)
    - **id** (automatic) – identification of the category of the expense or income movement;
    - **type** – ("e", "i") either an expense category ("e") or an income category ("i");
    - **name** – name (text description) of the category.

9 DELIVERY AND PUBLISHING

The application must be published in an external service (no home servers allowed) and accessible on the entire web via a desktop browser (tests will be made on google chrome). **The functional correction of the application will be made exclusively in the published version, so publication is not optional.**

When publishing the project, all functionalities and configurations should be as close as possible to the application final production stage. Publishing should include all performance optimizations and must use data that simulates real life application usage (apply the provided database seeder). Also, don't forget to use a real mail server so that when testing the application, it will send real e-mails to the selected e-mail addresses.

Students are free to choose the provider to use, however some suggestions are presented:

- Amazon Free Tier (https://aws.amazon.com/free/)
- Digital Ocean with the GitHub Education Pack (https://education.github.com/pack)
- RedHat Openshift (https://www.openshift.com/)

The project report, whose model (Excel file) will be provided on the Moodle platform, as well as the complete source code of the project, have a mandatory delivery.

The files to delivery (upload) through the link available on the page of the Course in Moodle are:

- code_NNN.zip – zip file that includes a copy of all the project's folders and files, except the folders "vendor" and "node_modules".
- report_NNN.xlsx – the report file (Excel file) that includes group elements identification and information about the project implementation. The model for the report will be provided.

Note: replace NNN with your group number.

10 EVALUATION

The evaluation (*avaliação*) of the project will consider the compliance with the mandatory constraints (C 1 … C 8), Functional Requirements (US 1 … US 17) and Non-Functional Requirements (NF 1 … NF 10).

Mandatory Constraints (C 1…C 8)

These are mandatory constraints. If the project does not comply with all these constraints (including application publishing), then it will have a **classification of zero**.

Functional Requirements (US 1…US 17)

For the classification of each User Story, it will be considered the compliance with the business model, the usability, the reliability and the integration of the User Story with the application, the quality and structure of the project's source code to implement it, as well as the compliance of non-functional requirements applicable to the user story.

The classification of all user stories is valued **70%** of the project. Individual values for each user story are specified on the table with the weight of all evaluation criteria.

Non-Functional Requirements (NFR 1…NFR 10)

Non-functional requirements are valued **30%** of the total classification of the project. These requirements are evaluated as a whole (there is no individual classification for each non-functional requirement) through the entire application. The evaluation of these requirements is made in parallel with the evaluation of the functional requirements, by analyzing the code and structure of the project and by testing (manually or automatically) chosen Web API endpoints.

Weight of all evaluation criteria:

US 01	US 02	US 03	US 04	US 05	US 06	US 07	US 08	US 09
1%	4%	3%	1%	4%	5%	5%	5%	5%

US 10	US 11	US 12	US 13	US 14	US 15	US 16	US 17	NFR*
5%	3%	5%	5%	5%	4%	5%	5%	30%

** – all Non-Functional Requirements*