Distributed Applications Development

Computer Engineering	3rd Year	1st Semester	2018-19	Periodic Evaluation
Project	Limit date for results divulgation: 18 Jan 2019			
Date: 2 Nov 2018	**Delivery Date: 7 Jan 2019**			

Project - Restaurant Management

1 OBJECTIVE

This project's objective is to implement a Single Page Application (SPA) for the management of a restaurant's order system, using Vue.js framework for the development of the web client.

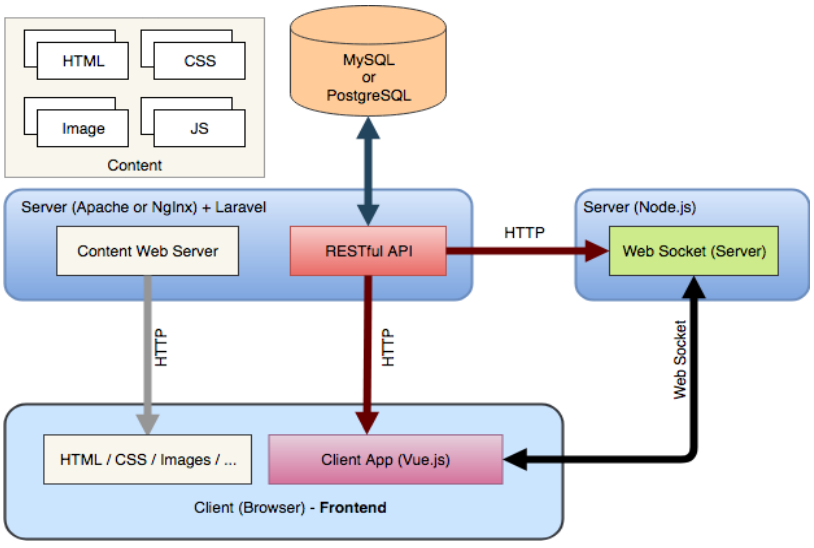
2 SCENARIO

The main purpose of the order system is to improve the efficiency of the communication between managers, waiters, kitchen personal (cooks) and cashiers of a restaurant. Each time a waiter takes an order, the required items of that order are passed on as fast as possible to the kitchen. Also, the waiter must be notified as soon as the orders are completed in the kitchen, and the cashier as soon as the meal ends and the payment is due. To ensure the maximum efficiency, the communication within the application should happen in real time.

To simplify the scenario, each order is comprised of only one item (dish/food or drink). Drinks and dishes have the same flow, they are all prepared by cooks.

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 application is accessible to the restaurant staff and to the public in general (anonymous users). The public can only view the menu, which includes all food dishes and drinks, as well as their descriptions, photos and prices. The restaurant staff may assume one of 4 distinct roles in the application: manager; cook; cashier and waiter.

The restaurant is opened 24 hours, which means that all staff works in shifts. Each person that works in the restaurant has an account that includes his full name, username, e-mail, photo and password (minimum size of the password is 3 characters). When a worker shift starts, he must enter the application with his authentication credentials (username or email and password) and declare that his shift has started (for instance, by clicking a button). The same must happen when his shift ends. The information about the active workers in the current shift (the users that are currently working in the restaurant) is fundamental to ensure that only active workers receive notifications. Other users (that are not presently working), even if they are logged in the application, should not be disturbed by any notifications.

The restaurant managers can handle the restaurant menu - add, change or delete menu items (dishes or drinks), administer application users (restaurant staff) and have access to information about the performance of the restaurant or the restaurant cooks and waiters (KPI – key performance indicators).

The cook (kitchen personal) receives dishes or drinks orders from the waiters, and after preparing each dish or drink, notifies the waiter that the order is prepared (by marking the order as "prepared"). The orders are received by all cooks in the kitchen. A cook may select the dish (i.e. the order) he is going to prepare, and that dish will be his responsibility and will be marked "in preparation". Once the cook finished the dish it marks the dish (i.e. the order) as "prepared". Alternatively, the cook may mark any order that arrives in the kitchen (i.e. "confirmed" order) directly as "prepared" - without first changing its state to "in preparation". That happens when the order refers to a pre-prepared item such as drinks or pre-prepared desert.

The order will have information about the cook responsible for the preparation of the item, even if it is a pre-prepared item (the same user that marked the order as "prepared").

The waiter is the main responsible for the relationship with the customers and handles their orders as fast as possible, until they end their meals. In the context of this application, one meal is associated to one table, and may include several orders, where each order is relative to one dish or one drink. One meal (with one or more orders) will generate one invoice.

Waiters are not responsible for specific tables, however, each meal (one table with orders) will always have one and only one waiter responsible for it (i.e. once a meal starts, there will be only one waiter serving the table).

When a new customer or customers arrive to a table, one of the waiters starts a new meal (in the application) associating it with the table number. Afterwards, the waiter will take note of the dishes and drinks (orders) that customers order. During 5 seconds the orders are considered as "pending" orders, and during that time, the waiter can cancel the orders and delete them from the database. After 5 seconds (without any intervention from any user), orders are considered as "confirmed" and cancellation is no longer possible. At this point the kitchen staff (cooks) is notified of the orders - each order refers to one item (food dish or drink).

After the cook finishes an order the application will notify the waiter that is responsible for the order (the waiter that registered the order initially). At that point, the order is considered as "prepared", and the waiter can pick it up at the kitchen and deliver it to the customer. After delivering it, the waiter marks (within the application) the order as "delivered".

When the waiter confirms that the customers have ended their meals, he marks the associated meal as "terminated". On that moment, a pending invoice is created for that meal, and the cashiers will receive a notification with information about that invoice. The invoice includes the information about the meal's date and table number, the total price of the meal and the detailed information about all items consumed (quantity, item name, unit price, sub-total price). Orders that were not delivered ("pending", "confirmed", "in preparation" or "prepared") should be closed automatically (state changes to "not delivered") and not considered as a consumed item. In contrast, orders with the state of "delivered" will be passed on to the invoice as "invoice items", merging all orders of a single item type to a single invoice item. For instance, 5 orders of the same item (e.g. 5 "waters") will generate a single invoice item with the quantity value of 5.

Cashier will then receive the payment for the invoice (out of the context of the application), fill the NIF and customer's name within the invoice (both are required) and closes the invoice (invoice state changes from "pending" to "paid"). As soon as the invoice is closed (i.e. state is "paid"), the meal's state changes from "terminated" to "paid" (the associated orders state is maintained - at this point, orders are either "delivered" or "not delivered") and the invoice is made available for download as a PDF file to any cashier or manager. Also, at this exact moment, restaurant managers are notified that a previously "terminated" meal is now closed and considered as "paid", helping them to check if someone leaves the restaurant without paying what is due.

Besides all automatic notifications due to the typical workflow described above, waiters, cooks or cashiers may send, at any time, a notification to the managers of the restaurant with a text describing a problem or situation that the managers need to be aware of (only managers that are "working" will receive the notification).

Also, to avoid inconsistent and indefinitely opened orders, meals or invoices, the managers may change manually the state of meals and invoices to "not paid". When changing the state of a meal or invoice to "not paid", the associated invoice (if any) or meal must also change its state to "not paid". Also, the application must guarantee that all orders from a "not paid" meal that are not in "delivered" state, change their state to "not delivered".

Summary of order states:

List of all order states:

- "pending"
- "confirmed"
- "in preparation"
- "prepared"
- "delivered" – final state (order was handled and delivered to the customer)
- "not delivered" – final state (order was not delivered to the customer)

List of all meal states:

- "active" – meal is currently active, no payment is due
- "terminated" – meal has terminated but not paid yet - payment is due
- "paid" – final state (customer has paid for the meal)
- "not paid" – final state (customer has left without paying the meal)

List of all invoice states:

- "pending" – cashier is filling NIF and customer name
- "paid" – final state (invoice data was filled by the cashier and the invoice was paid)
- "not paid" – final state (customer has left without paying the invoice's meal)

Note: Do not forget that all notifications referred on the business model are sent **only** to the workers (users) that are currently working on the restaurant (making the **current shift**).

5 FUNCTIONAL REQUIREMENTS

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

1 ALL USERS

✓ US 1. As a user (anonymous or authenticated user) I want to access the menu of the restaurant, which includes all food dishes and drinks, as well as their descriptions, photos and prices.

2 RESTAURANT WORKER (MANAGER, COOK, WAITER, CASHIER)

✓ US 2. As a worker of the restaurant, I want to receive a confirmation e-mail when a manager registers my account on the system. The e-mail should include a hyperlink, that opens a web page where I can define my account password. At the end of that process, the account is considered a confirmed account.

✓ US 3. As a worker of the restaurant, after confirming the account and defining my password (US 2), I want to authenticate with the application using the credentials username+password and/or email+password. Also, I want to logout of the application.

✓ US 4. As a worker of the restaurant, after confirming the account, I want to be able to change my password at any time.

✓ US 5. As a worker of the restaurant, I want to view and change the details of my account. I should be able to change the username, full name and photo, and to view the e-mail. The e-mail cannot be modified by the user itself.

✓ Missing time since last shift

US 6. As a worker of the restaurant, I want to be able to join (declare the start) and quit (finish of) a shift by clicking on an element in the page (ex: button, image, link, etc). Also, I want to be able to view (preferably, within all the applications' views) if I'm currently working (on a shift) or not. When working, the application should display the time when the shift has started and the total time that has passed since. When not working, the application should display the time when the last shift has ended (if the worker has already terminated a shift) and also the total time that has passed since.

US 7. As a worker of the restaurant, I want to be able to view all my notifications (according to the business model described previously) on a dedicated area or with popup elements. Either solution must guarantee that the notifications always appear instantaneously and in all the applications' views. Each notification should have an element (ex: button, link, image, div, etc.) that when clicked will show the related application area. For example, if a cook is viewing his account details and receives a notification of a dish order, when he clicks on the referred notification element, the application will jump to the list of orders (where he would view the order associated with the notification).

US 8. As a worker (cook, waiter, cashier and manager) I want to be able to send a "manual" notification to all restaurant's manager (that are active on the current shift) with a text describing a problem or situation that the managers need to be aware of – the text must be filled by me.

3 COOKS

US 9. As a cook I want to access the list of all orders to prepare or that I am preparing, ordered by the time which they "arrived" to the kitchen. Earliest orders should appear on top and latest orders should appear on the bottom. Orders "in preparation" (my own "in preparation" orders) should appear first, followed by the confirmed orders, and should be presented with a distinct visual characteristic (for example, a different color).

Pending or prepared orders, as well as orders in preparation by other cooks should not appear on the list - only "confirmed" orders or my own "in preparation" orders appear on the list.

US 10. As a cook I want that the list of all orders to prepare or that I am preparing (US 9) to be instantaneously refreshed as new orders arrive, as orders change its state or are removed from the list.

US 11. As a cook I want to be able to declare an order as "prepared" or "in preparation" by clicking an element (button, image, link, etc.) within the order in the list of all orders.

4 WAITER

✓ US 12. As a waiter I want to start a new meal (create a meal record with the state "active") associating it with a table number. At any given moment, each table should only have one active meal.

US 13. As a waiter I want to take note of the dishes and drinks orders (create new orders) for a specific meal/table (each table should be associated to only one active meal). For this I'll have to select the meal and the dish or drink to order. The meal should be selected from the list of active meals that I'm responsible for (that I have created and with state "active"), and the dish or drink should be selected from the complete list of items in the menu.

US 14. As a waiter I want to view all orders that are "pending" or "confirmed" of the active meals for which I am responsible for. "Pending" orders visual representation should be distinct from the "confirmed" orders representation, for instance using a distinct color or text format.

US 15. As a waiter I want to be able to cancel (delete the record from the database) any "pending" order (of my active meals). I can do it up to 5 seconds after the order has been created.

US 16. As a waiter I want that the list of all "pending" and "confirmed" orders related to my meals to be instantaneously refreshed as they are removed from that list (change state to "in preparation", "prepared", "delivered" or "not delivered"), or as they change state from "pending" to "confirmed".

US 17. As a waiter I want to view all "prepared" orders of my meals and declare any of these orders as "delivered".

US 18. As a waiter I want that the list of all "prepared" orders related to my meals to be instantaneously refreshed as new "prepared" orders arrive from the kitchen, or as they are declared as "delivered" or "not delivered".

US 19. As a waiter I want to view a summary (table number and current total price) and the details (name and price of dish or drink) of all orders for a given meal. The meal should be selected among the list of all active meals for which I'm responsible.

US 20. As a waiter I want to declare a meal as "terminated". The meal should be selected among the list of all active meals for which I'm responsible. If there are any opened order that were not delivered (state different from "delivered"), the application should confirm the operation and allow it to be cancelled. If the operation is confirmed, all orders that were not delivered should be closed (i.e. state changes to "not delivered") and should not be included in the total price of the meal.

US 21. When the meal is terminated (US 20), the application will create a new "pending" invoice with all associated consumed items (invoice_items) for that meal. All data of the invoice

and invoice_items, except the NIF and customer name, should be automatically filled by the application according to the rules described on the business model.

5 CASHIER

US 22. As a cashier I want to access the list of all "pending" invoices - that are not paid yet. Within this list, each invoice should include at least the table number, the waiter that was responsible for the associated meal and the total price to pay.

US 23. As a cashier I want that the list of all "pending" invoices to be instantaneously refreshed as new "pending" invoices are created (when meals are "terminated") or as "pending" invoices are closed (paid or declared as not paid by the manager).

US 24. As a cashier I want to access the detailed information about any invoice, which includes the date, table number, waiter that was responsible for the meal, total price and a detailed list of all items consumed, including quantity, name (of dish or drink), unit price and sub-total price (quantity * unit price) of each item.

US 25. As a cashier, I want to fill the NIF and name of the customer of any "pending" invoice. NIF must have 9 digits and the name must have only letters and spaces.

US 26. As a cashier, after filling the NIF and customer's name (both are mandatory), I want to close an invoice by declaring it as paid (state = "paid"). The associated meal should also be closed automatically (state = "paid").

US 27. As a cashier I want that any paid invoice (i.e. state = paid) to be available for download as a PDF file. The PDF file should only be available for cashiers or managers.

6 MANAGER

US 28. As a manager I want to be able to handle the restaurant tables and restaurant menu - adding, changing or deleting tables and menu items (dishes or drinks). Each menu item should include its type (dish or drink), a description, photo and price. If for some reason the table or menu item cannot be deleted from the database, a soft delete should be used instead of a real database delete operation.

✓ US 29. As a manager I want to be able to add or change applications users, as well as view and filter the list of all applications users (including blocked and soft deleted users). Each user will correspond to a restaurant worker (either a cook, waiter, cashier or manager) and includes the full name, username, e-mail and photo. The username and the e-mail should be unique among all users.

Note that only a manager can create new users or change users' e-mail.

US 30. As a manager I want to be able to block, unblock or delete any user, except myself. A blocked user cannot access the application. If for some reason the user cannot be deleted from the database, a soft delete should be used instead of a real database delete operation.

US 31. As a manager I want to access a dashboard where I can observe at real time a summary of the information about "pending" invoices, "active" and "terminated" meals and all orders associated with these meals ("active" and "terminated"). For the "pending" invoices, and "active" and "terminated" meal, it is important to view the associated table number, responsible waiter and current total price. For the orders it is important to know to state of the order and the name of the dish or drink.

US 32. As a manager, from within the dashboard (US 31) I want to be able to declare "pending" invoices or "terminated" meals as "not paid" invoices or meals (i.e. change its state to "not paid"). On both cases, the associated invoice or meal must also change its state to "not paid". Also, the application must guarantee that all orders from a "not paid" meal that are not in "delivered" state, change their state to "not delivered".

US 33. As a manager I want to be notified when any meal is "terminated" or "paid" (i.e. the associated invoice is "paid").

US 34. As a manager I want to be receive "manual" notifications from cooks, waiter and cashiers.

US 35. As a manager I want to be view and filter the list of all meals being served or already served on the restaurant. By default, I want to view "active" and "terminated" meals only, but I want to be able to filter by state ("active", "terminated", "paid" and "not paid"), by date and by the waiter responsible for the meal.

US 36. As a manager I want to view a detailed information for any given meal, including the meal state, date, the waiter responsible, the total price of the meal and a detailed list of all items ordered, including the type (dish or drink), name, price and order state.

US 37. As a manager I want to be view and filter the list of all invoices created on the restaurant. By default, I want to view "pending" invoices only, but I want to be able to filter by state ("pending", "paid" and "not paid"), by date and by the waiter responsible for the meal.

US 38. As a manager I want to view a detailed information for any given invoice, including the invoice state, date, the waiter responsible, the total price and a detailed list of all items consumed, including the type (dish or drink), quantity, name, unit price and sub-total.

For "paid" invoices I want to be able to download the invoice as a PDF file (US 27).

US 39. As a manager I want to view statistical information about the performance of the cooks and waiters, namely: the average number of orders handled by day for each cook and waiter; the average number of meals handled by day for each waiter (e.g. 23.4 meals per day).

US 40. As a manager I want to view statistical information about the performance of the restaurant organized by month (for each month), namely: the total number of meals and orders handled; the average time it takes to handle each meal (time between the moment a meal is created until the meal is closed – "paid" or "not paid"); the average time it takes to handle each order (time between the moment an order is created until the order is closed – "delivered" or "not delivered") organized by item of the restaurant's menu (i.e. for each month the average time to handle each of the items in the menu).

6 CONSTRAINTS

Project's **mandatory constraints** are the following:

C 1. Application must be published and accessible on the entire web through a desktop browser (tests will be made on google chrome).

C 2. Application must use exclusively the Single Page Application (SPA) paradigm.

C 3. Application must use the Vue.js framework for the client-side code.

C 4. Applications' web services (Restful API) must be implemented with Laravel framework.

C 5. Applications' web socket server must be based on Node.js.

C 6. Main relational database must use either MySQL or PostgreSQL

7 NON-FUNCTIONAL REQUIREMENTS

Project's non-functional requirements are the following:

NFR 1. When displaying data as a list or table, the total number of potential records should be considered. This implies that for all but small datasets, lists and tables should implement server-side pagination and sortable columns. Also, when possible filtering the dataset should also be considered.

NFR 2. Form's data should always be validated on the client and on the server, according to rules extrapolated from the business model and functional requirements.

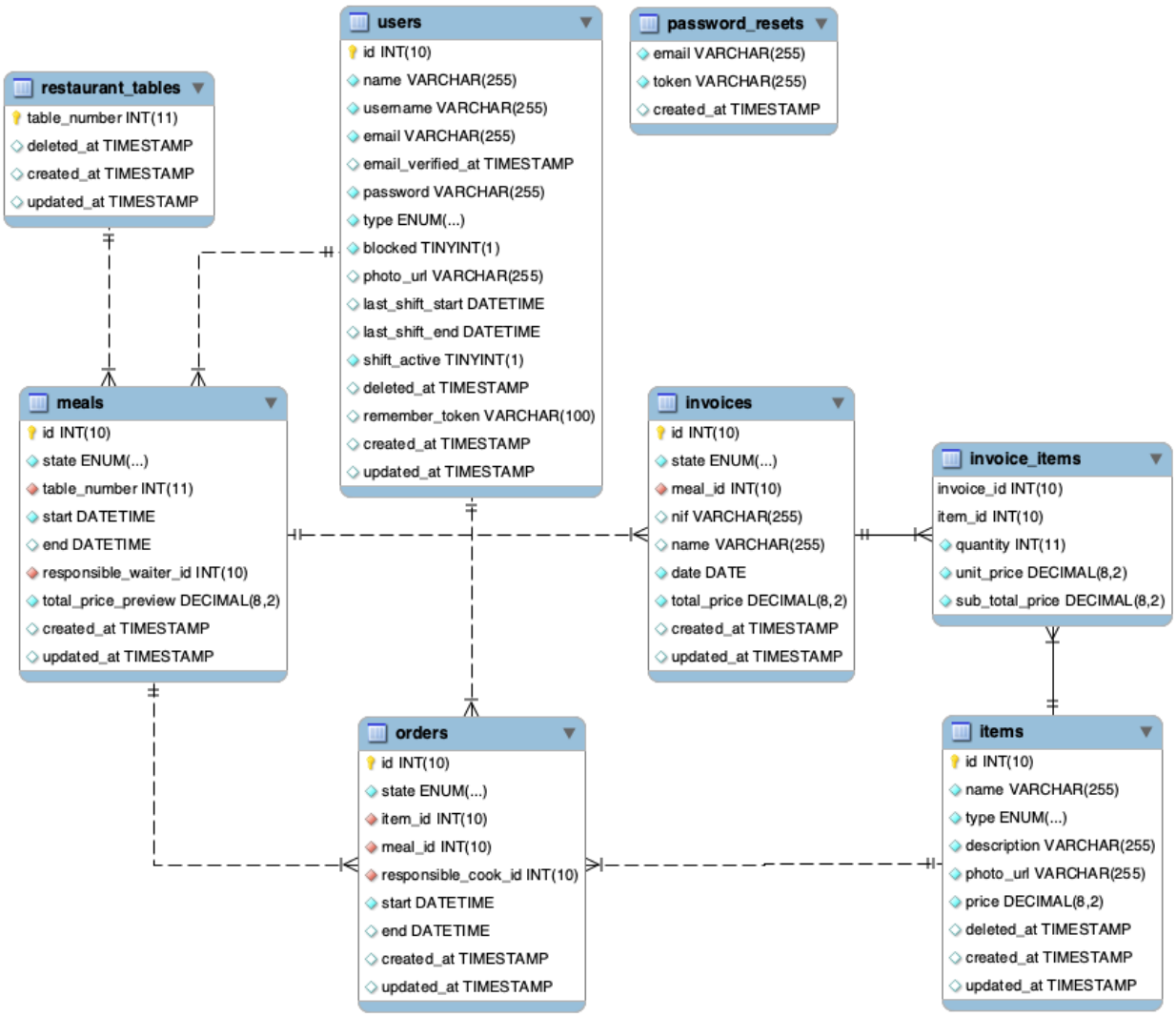
NFR 3. Visual appearance and layout should be consistent through the entire application, and adapted to the applications' objective and usage context. Graphical information should be used to enrich statistical data and dashboards.

NFR 4. Application should be reliable and have an optimized performance. This implies that the application's database will be seeded with data that simulates real life usage (if the database is small, then reliability and performance optimizations cannot be tested).

NFR 5. 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.

8 DATABASE

The main relational database must use either MySQL or PostgreSQL, but can be complemented by any Non-SQL database. Students are free to define the database structure, but can use or adapt the following database structure (provided as a Laravel Migration):



Also, when using previous database structure, a **database seeder is also provided**, which can be used to preload the database with data that simulates the application usage through several months.

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 should use data that simulates real life application usage. 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 service 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 by UC teachers through the Moodle platform, as well as databases backups and 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".
- db_bak_NNN.zip" – zip file with all backups required to restore a complete copy of all databases used (MySQL or PostgreSQL and Redis, Memcached, MongoDB, etc.)
- 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 by the teacher.

Note: replace NNN with your group number.

10 EVALUATION

The evaluation (avaliação) of the project will consider the resources delivered, compliance with the mandatory constraints (C1 … C6), the implementation of the Functional Requirements (US1 … US38), compliance with Non-Functional Requirements (NFR1 … NFR5) and the quality and structure of the project's source code.

Resources Delivered

Students must deliver the code, database backups and **report** (correctly filled) on moodle link. Not delivering these resources implies that the project classification will be zero.

Mandatory Constraints (C1…C6)

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 (US1…US40)

Each User Story (functional requirement are specified as Users Stories) will be valued **2%** of the total classification of the project (80% total). For the classification of each User Story, it will be considered the usability, the compliance with the business model, the reliability and the integration of the User Story with the application, as well as some non-functional requirements that are relevant for that particular User Story.

Non-Functional Requirements (NFR1…NFR5)

Each non-functional requirement will be valued **2%** of the total classification of the project (10% total). For the classification of NFR3 and NFR4 non-functional requirements, it will be considered an informal evaluation of the general compliance of these non-functional requirements through the entire application. For the classification of NFR1, NFR2, NFR5 and part of NFR4 (performance), it will be considered a test of one selected (by the teacher) use case of the application where the compliance with the non-functional requirement is important.

Quality and structure of the project's source code

The quality and structure of the project's source code will be valued **10%** of the project. This includes all the projects components, such as the REST API, Vue.js client, Web socket server, database structure, as well as the quantity and quality of the sample data of the published application. For the classification of this criteria, it will be considered an informal overview of the structure of the source code and components used, as well as a small set of partial analysis of sections of the source code - the teachers will select some features of the project, and analyze how they were implemented in the student's project.