Uliège eShop
e-commerce and e-business project

Beguin Mathias - Lejoly Loic
Github access: `https://github.com/MTHSBGN/e-commerce`

November 2018

1 Introduction

For this project, we decided to work on the proposition given in the statement (Uliège eShop). As asked in the statement the goal is to provide a website where objects related to the University can be sold. To achieve that, we decided to use a relational database to store products and data related to customers. Concerning the back-end's website NodeJS and SQL are used. NODEJS is a cross-plateform JavaScript run-time environment. That means Javascript is used outside client browser. This cross-platform is young compare to PHP but more and more companies tend to use it due to its design and scalability. Concerning the front-end part as usual HTML,CSS and JS are used.

2 Database architecture

To model the database of the Uliège eShop website a relational database architecture was chosen because the product catalogue is not large and because it can be hard to have a good NO-SQL design. On the web you can find some debate that discuss the fact to use relational DB or no-SQL DB. The advantage of the no-SQL DB is that it is easier to manage your product catalogue (product variations) but the drawback of no-SQL DB is the reliability (can be solved with DB clustering e.g: foundationdb, couchDB,...). Concerning the reliability relational DB architectures are, for the majority, ACID compliant. These two types of architectures are not mutual exclusive. Most of the time, e-commerce websites use of combination of SQL and No-SQL DBs (a paper discussing that topic `https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1026&context=sais2013`).

2.1 entity-relationship model

Figure 1 depicts the entity-relationship model based on the minimal requirements given in the statements. We decided to split the product table proposed in the statement into multiple tables to avoid data replications and to provide a scalable approach. Indeed, in the product table proposed, all data linked to a product is packaged into this table. The result of that is it is difficult to manage products with different properties (e.g colors, size,...) without replications. That is why we decided to split this table into several tables that are *Sku (stock keeping units) Variant (gives the properties of a product), Image(image paths related to a product), Description (Give the product description), and Category (specify the product category).*

Concerning the transaction part we decided to split that part into two related steps. The first step establishes a relation between the sku ID, the order ID (defined in the second step) the current price of the product during the order and the quantity needed. This step is achieved with the table Order_details. The second step consists in summarizes the the order of the client by referencing the ID of the order, the delivery address, the total price of the order, and the user ID.

2.2 Relational model

1. Customer(client_id, mail, username, password, delivery_address, firstname, lastname, cookie_id)
2. Client_order(order_id, #client_id, date, total_price, ship_address)
3. Order_details(#order_id, #sku_id, quantity, price)
4. Product(product_id, #category_id, #image_id, #description_id, name)
5. Image(image_id, #sku_id, filename)
6. Sku(sku_id, #produt_id, price, available, sold)
7. Variants(variant_id, attribute, value)
8. Description(description_id, french, english)
9. Category(category_id, #description_id, name)

2.3 model to SQL tables

The translation of the model depicted on Figure 1 can be translated into SQL tables as Figure 2 shows. If you want a more interactive view of these tables, you can load the sql file in *phpmyadmin* for instance.

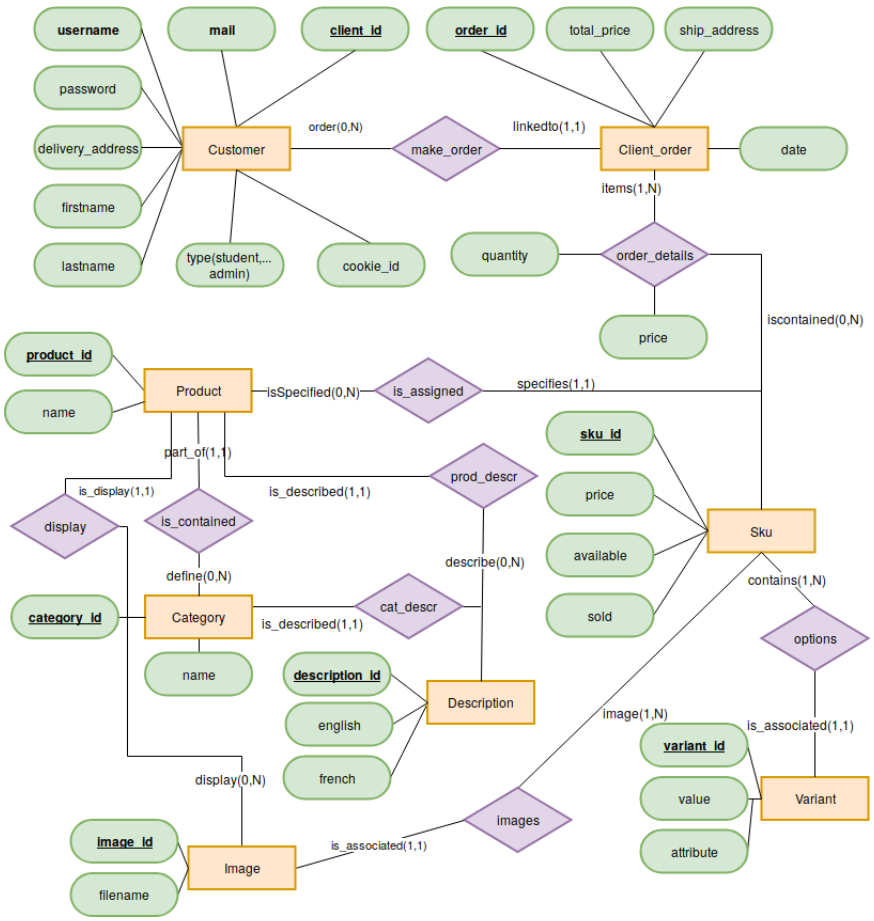


Figure 1: eShop entity-relationship model

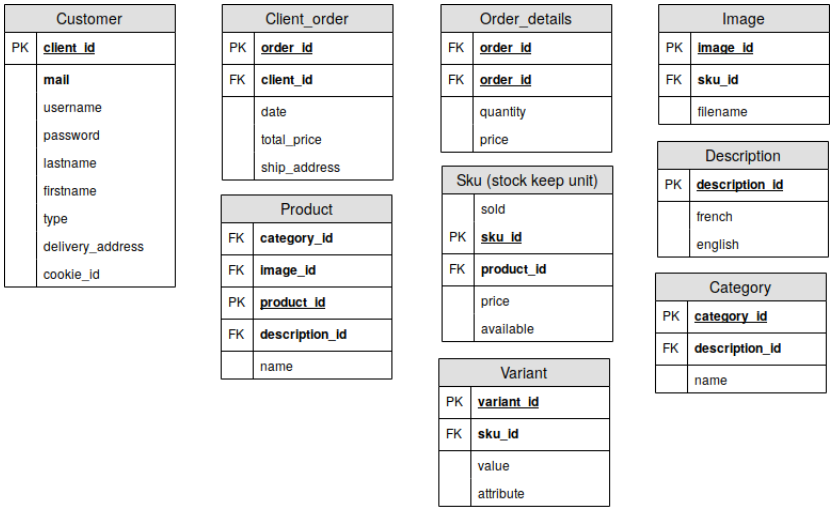


Figure 2: eShop entity-relationship model

3 Website

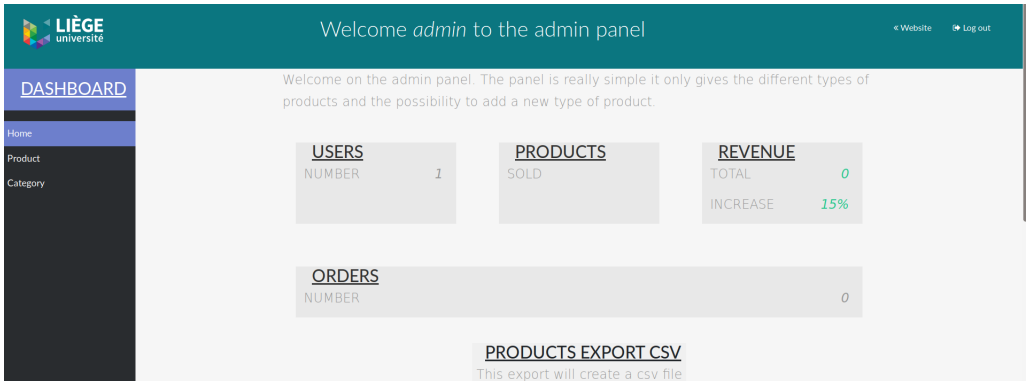
The website is divided into section. The first section is dedicated to the admin of the website itself an can only be accessed by users with the right privileges. The second second is dedicated to customer where they can buy products. For the development of the website we applied minimal protections like SQL injection managed by the node.js server.

3.1 Admin panel

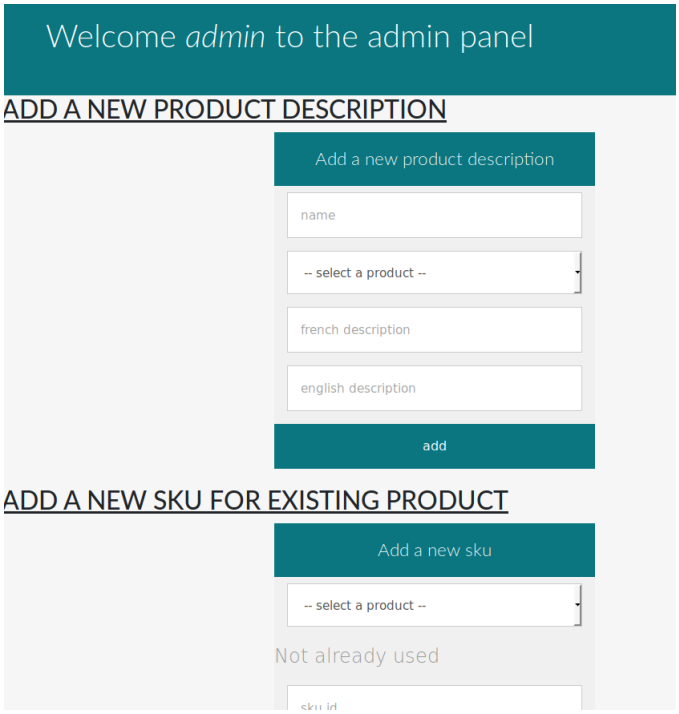
The admin panel is a dashboard provided to the administrator of the website. The users that can use this panel have special privileges. To detect this kind of users the field type of customer table is set to zero. As results, a admin user needs to follow the regular registration phase to create an account and then the super user that has a direct access to the database can change the type field to promote a user as admin user. Admin users have an extra button to allow them to access to the admin panel.

The dashboard is simple at left a menu can help the user to navigate through the different webpages. As you can see on the Figure 3 the panel only contains 3 pages (home,product,category). The homepage summarizes different parameters, you can also export some data. The product and Category pages give the possibility to add new elements to the database easily.

Even if you are on the dashboard you can go back to the e-commerce pages just by clicking on the website button.



(a) admin homepage



(b) product manager

Figure 3: admin pages

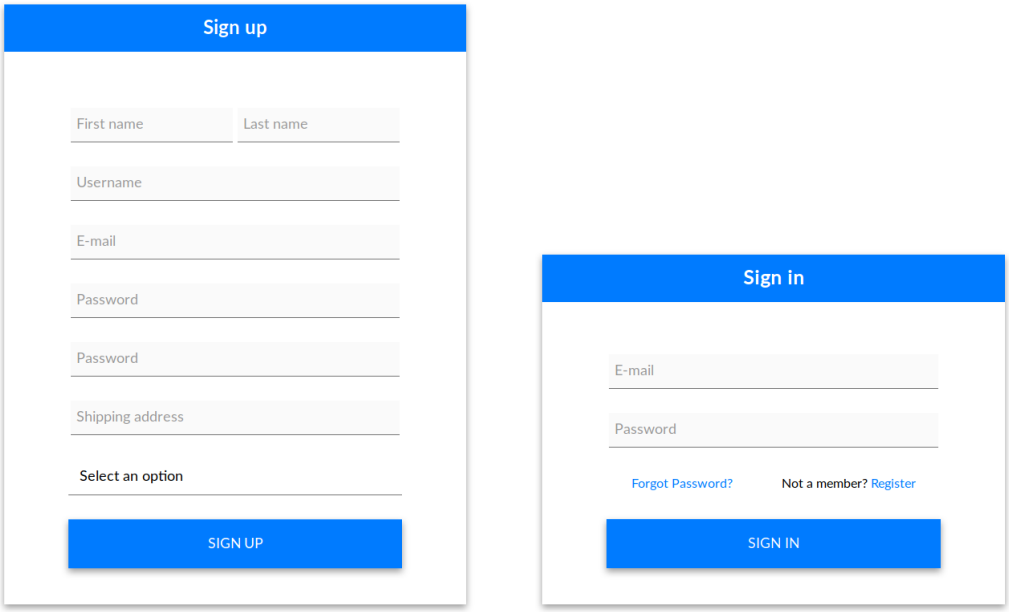
3.2 Customer View

3.2.1 Registration

To purchase products, a customer needs to create an account on the website to do so, he needs to click on the button dedicated to that purpose or simply follow the appropriate url. On that page the user must fill fields with correct informations and then validate his account. If everything is correct, he is redirected to the login page.

3.2.2 Log in

After the creation of an account a user can connect to the website by using his account informations. To achieve that he simply clicks on the button made for that purpose or going to the appropriate url. If the information given are correct he will be redirected on the home page.



(a) register page (b) login page

Figure 4: connection pages

The login page is the same for the admin, since the admin has more privileges when he is logged, he can access more pages such has the admin panel.

3.3 Basket

3.3.1 Payment

Since none payment procedure like PayPal etc. is set, a simple payment page is used to update the database. Even if it is a simple page, a transactional commit phase is established to be sure that all tables are updated correctly.

3.4 Security aspects

Concerning the security of the website, some common guidelines are applied such as SQL injections and transactional commits. But to provide a robust website some other security rules must be applied such as directory permissions, redirection rules by catching different errors,...