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 *Node.js* and SQL are used. *Node.js* is a cross-platform 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 its scalability. Concerning the front-end part as usual HTML,CSS and JavaScript are used.

2 Database architecture

To model the database of the Uliège eShop website a relational database architecture was chosen. The reason of this choice is justified by 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 debates that discuss the fact to use relational DBs or no-SQL DBs. 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 DBs 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

?? 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) and Image(image paths related to a product).*

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 OrderDetails. 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, session_id, update_at, created_at)
2. CustomerOrder(order_id, #client_id, date, total_price, ship_address, , update_at, created_at)
3. OrderDetails(#order_id, #sku_id, quantity, price, update_at, created_at)
4. Product(product_id, name, update_at, created_at)
5. Image(image_id, #sku_id, filename, update_at, created_at)
6. Sku(sku_id, #produt_id, price, stock, sold, update_at, created_at)

2.3 model to SQL tables

The entity-relational model depicted on **??** can be translated into SQL tables as **??** shows. If you want a more interactive view of these tables, you can load the SQL file in *phpmyadmin* for instance.

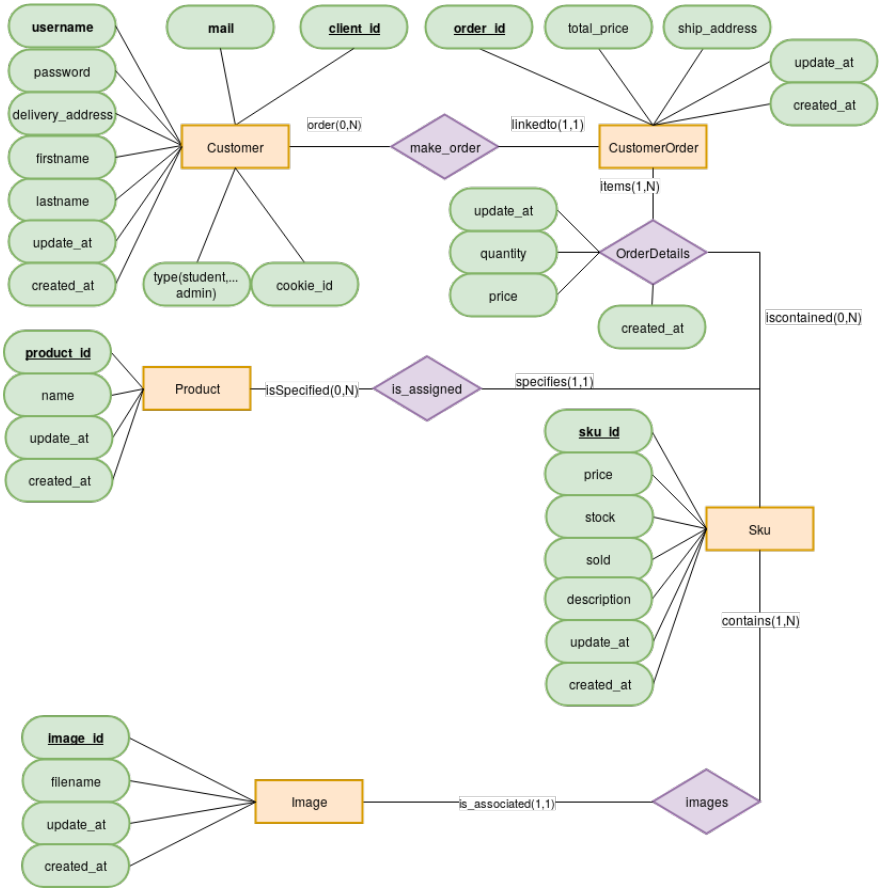


Figure 1: eShop entity-relationship model

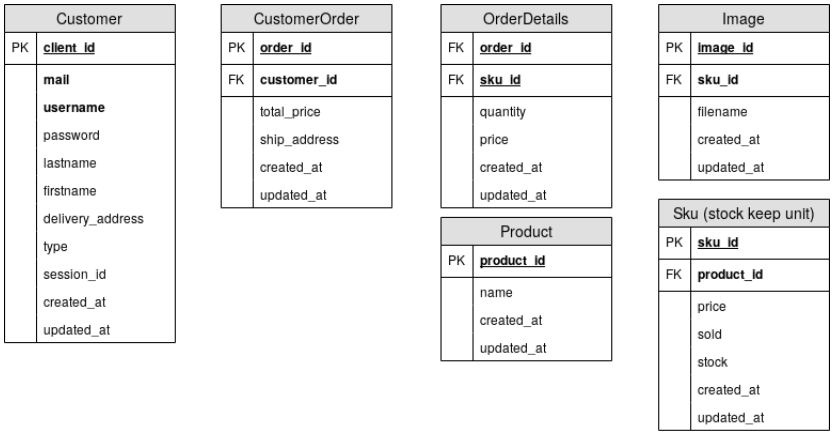


Figure 2: eShop entity-relationship model

3 Website

The website is divided into section. The first section is dedicated to the admin of the website itself. This panel 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.

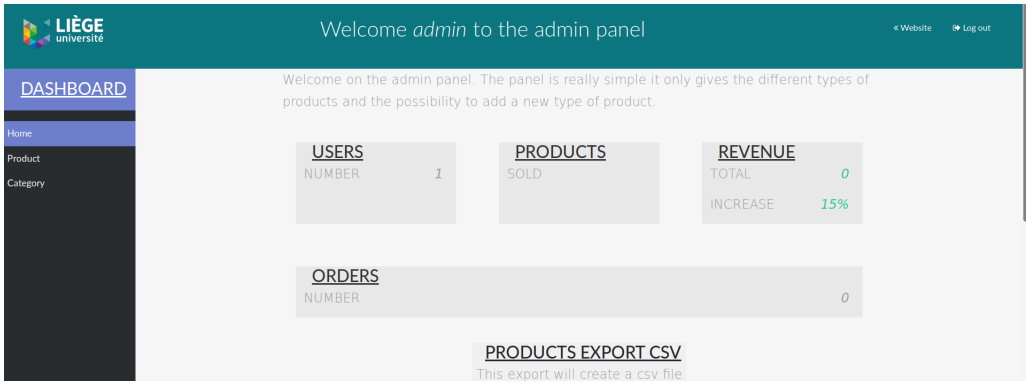
3.1 Admin panel

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

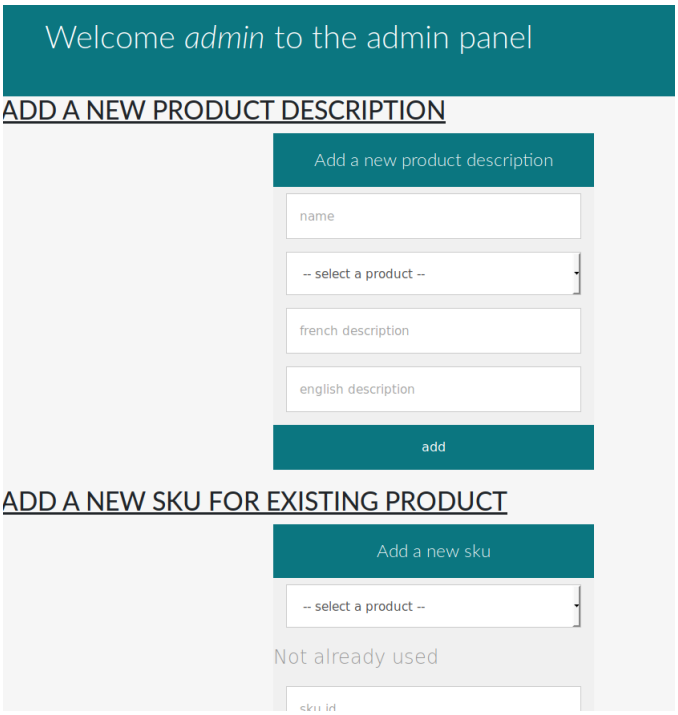
users have an extra button to allow them to access to the admin panel.

The dashboard is simple, on the admin homepage you have a summary of your stock. You also have a page to add a new product inside your database. You can also export your files in a csv format that is the standard format to export data.

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

If you are a regular customer, you can just have access to the e-commerce webpages. On the homepage you can see the different products that are sold by the University. On top of the website at right you have several buttons. One to create an account, one to have access to your basket and one which is the about page. To buy products a user needs to create an account and this account must be active.

A user can scrolls the homepage that contains all products and when he is interested by a product, he can click on that product and he will be redirected that the detailed product page.

Like common e-commerce websites, users can search for products by using search-bar or simply by walking through the website. Concerning this website, we do not received enough information concerning products (i.e:

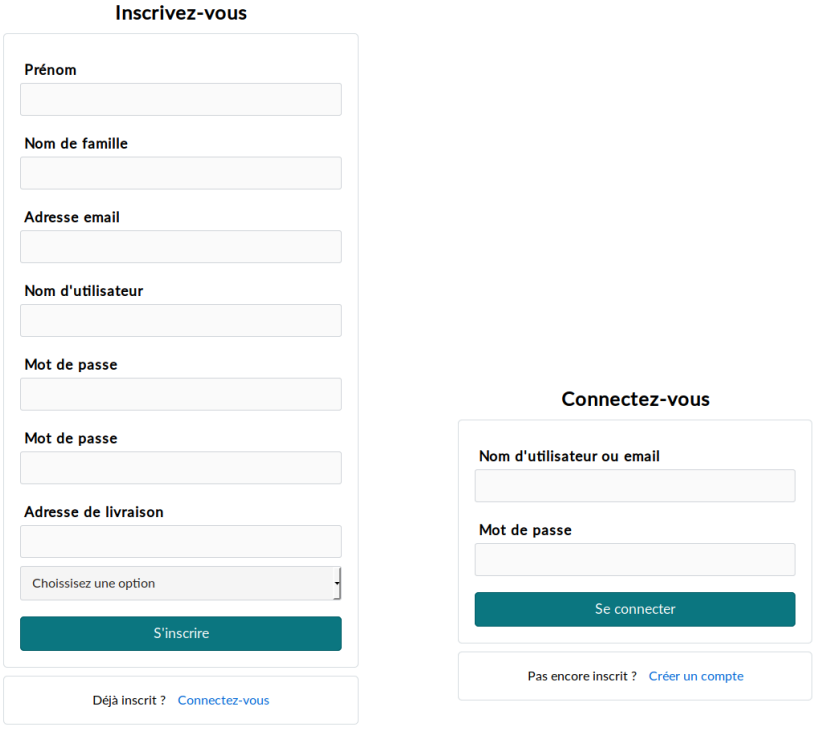
categories, etc.) so we decided to display categories since the number of products to display is rather small. Nonetheless, products has a category as you can see on the database schema. Administrators have thus the choice to use it or not.

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 information. To achieve that he simply clicks on the button made for that purpose or is going to the appropriate url. If information given is correct he will be redirected on the homepage.



(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

To store the basket of a user, HTTP cookies are used. When a user clicks on its basket, he obtains a summary of his products. This summary gives the possibility to modify product quantities or to delete some of them. When the user is in accordance with his order summary, he can validate it only if he has an account and if he is connected. The validation process send information to the payment procedure where all element are stored and where the order becomes effective.

3.3.1 Payment

Since none payment procedure like PayPal etc. is set for this project, 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,...

4 Market model

Concerning the target of the website it is obvious. It concerns essentially Alumni, employees and students of ULiège. The eShop provides a bunch of goodies related to the university and the city of Liège. Prices given are affordable for student and the aim of the eShop is not to be competitive with others websites that sell similar products. The aim is to provide a website where university community can buy university products easily. The website can also be used by student associations to sell their pulls for instance an provides a legal framework to sell them.

5 Improvements

Concerning the website some improvements can be done. First of all concerning the design, we tried to have a flat design user interface but we are not web designer nor user experience designer (UX). After that, security aspects must be improved and not neglected (DDoS attacks,XSS,etc.). Finally, we also developed the schema **??** of a more complex database that has several detailed added. This schema is not implemented for this project but can be used as a basis.

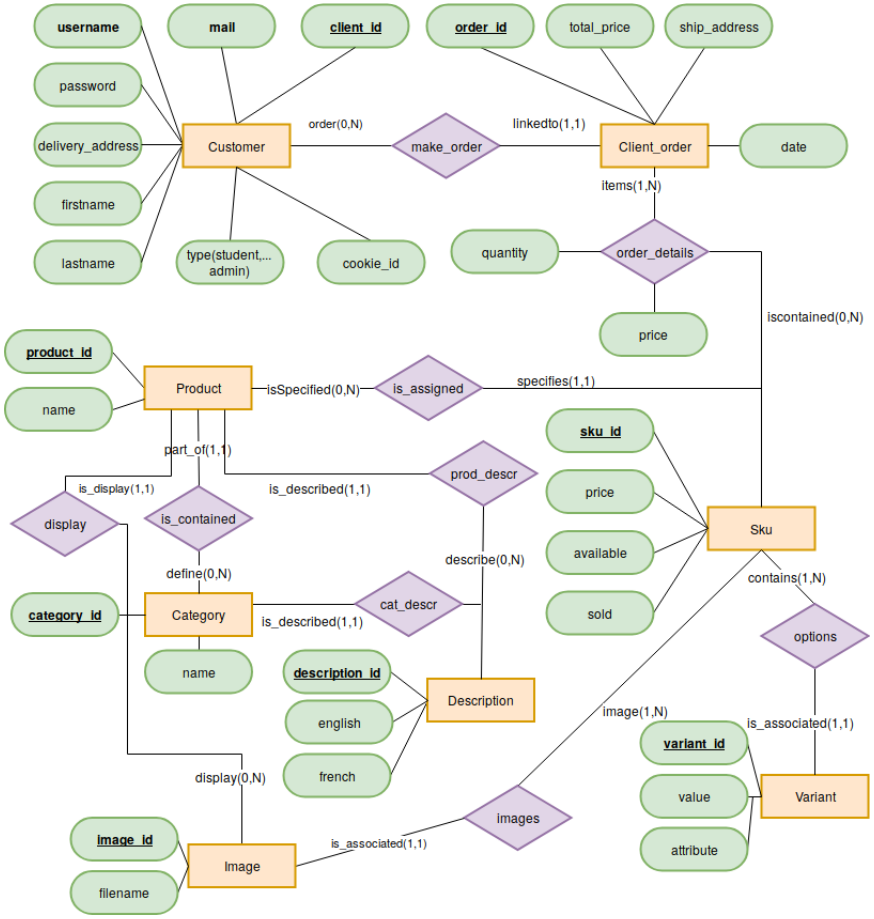


Figure 5: eShop entity-relationship model

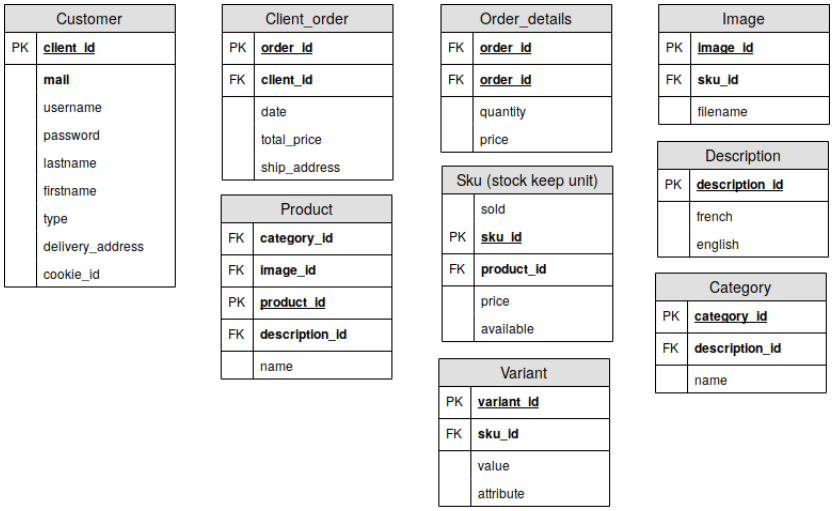


Figure 6: eShop entity-relationship model