Nacho’s Library

Documentation to manage the library

Nacho Gómez Buenaventura

Índex

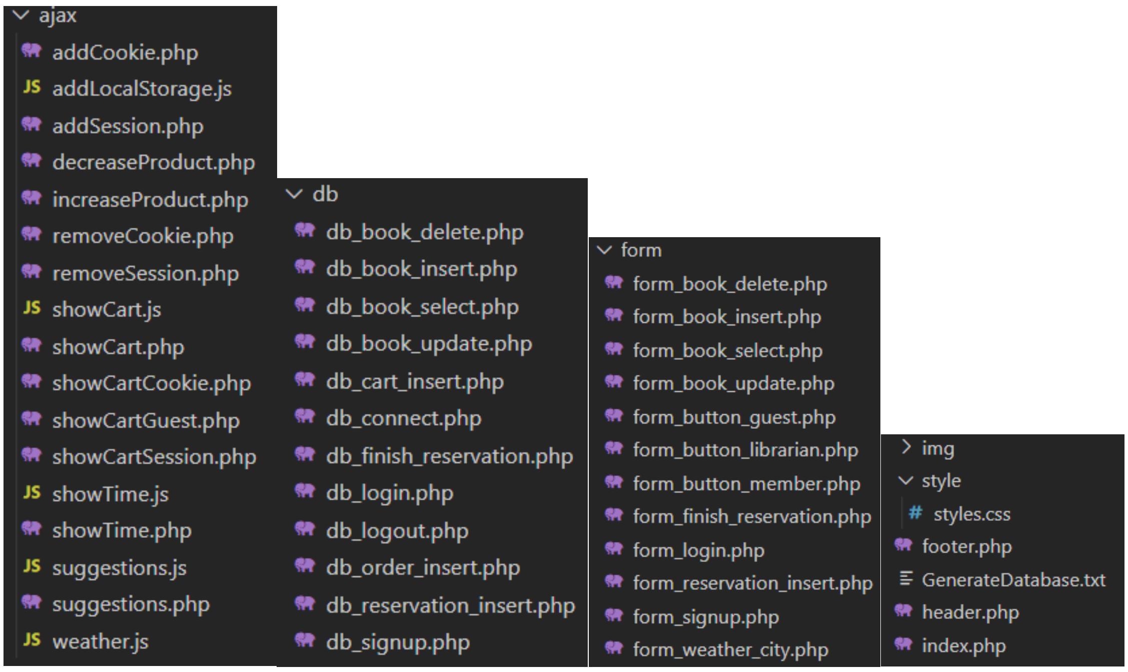
Customer requirements

- Nice appearance with personal HTML & CSS (frameworks such as bootstrap or materialize also accepted).
- HTML forms to enter data.
- SQL database to store data.
- PHP files to capture information from the HTML forms and connect to the SQL database
- Automatic calculation required: when returning a book x days late, you should not be allowed to borrow a new book before x days.
- Add products from the online shop
- Modify the product quantity within the cart
- Make the product disappear from the cart if quantity is 0, or include button/icon to delete it.
- Include a button or similar to process the order.

Users privileges

- librarian: access to books (search, insert, update or delete), members (search, insert, update or delete), reservations (search, insert, update or delete).
- member: books (search), reservations (insert).
- guest: books (search), members (insert=autoregistration).

Folder structure



```
ajax
  addCookie.php
  addLocalStorage.js
  addSession.php
  decreaseProduct.php
  increaseProduct.php
  removeCookie.php
  removeSession.php
  showCart.js
  showCart.php
  showCartCookie.php
  showCartGuest.php
  showCartSession.php
  showTime.js
  showTime.php
  suggestions.js
  suggestions.php
  weather.js
db
  db_book_delete.php
  db_book_insert.php
  db_book_select.php
  db_book_update.php
  db_cart_insert.php
  db_connect.php
  db_finish_reservation.php
  db_login.php
  db_logout.php
  db_order_insert.php
  db_reservation_insert.php
  db_signup.php
form
  form_book_delete.php
  form_book_insert.php
  form_book_select.php
  form_book_update.php
  form_button_guest.php
  form_button_librarian.php
  form_button_member.php
  form_finish_reservation.php
  form_login.php
  form_reservation_insert.php
  form_signup.php
  form_weather_city.php
img
style
  styles.css
footer.php
GenerateDatabase.txt
header.php
index.php
```

Project files

- index.php → have all the includes that we need for each member.

- form_FUNCTIONALITY.php → all the files that start with "form" is a form to enter data for a specified functionality, for example, if the functionality is book_insert the file will have a form about the data that we can store in the database from a book.

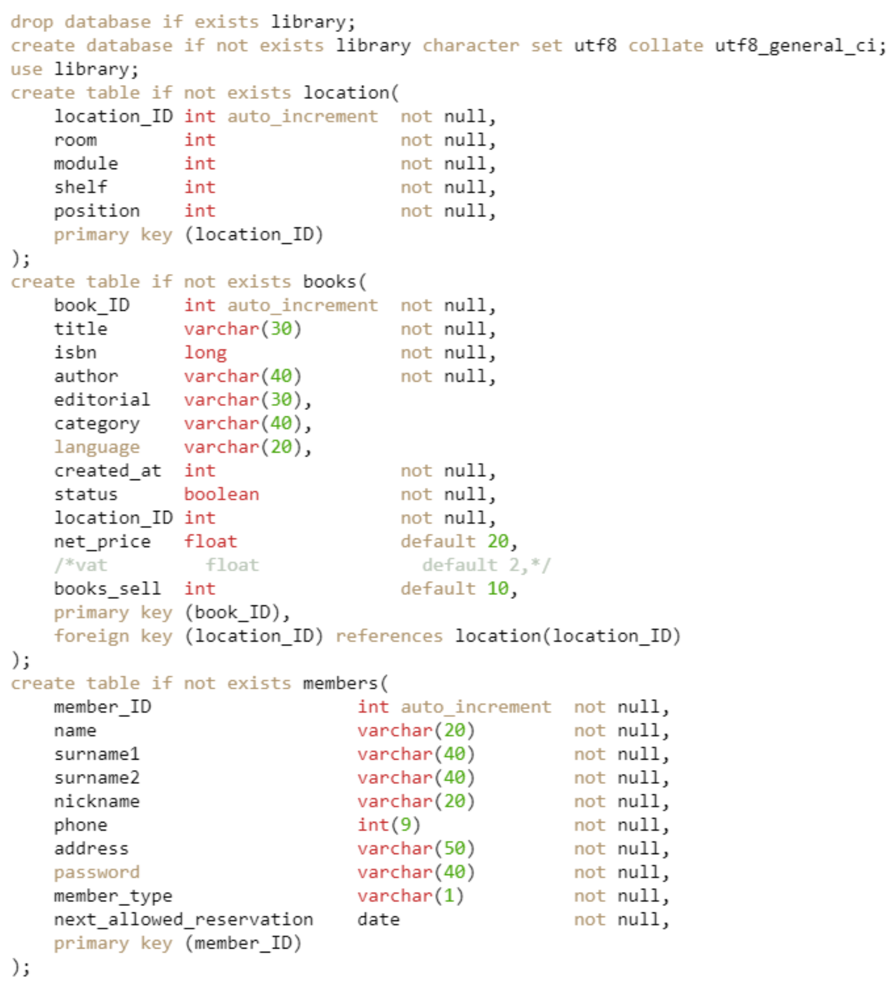
- db_connect.php → Is the file that content the query to connect to our database. I have done it like this because I need to do a connection every time to the database if I want to do any action. Is more simple to me to do an include from the file than write several lines every time I wanna connect to the database.

- db_FUNCTIONALITY.php → all the files that start with "db" is the next step from the forms. The function from this files is get the data from the forms and do some actions to add,update,delete books, members or reservations.

- Ajax folder → all the files have an specific solve of a requirement. The names are predictable their function.

Database

Now you can see the lines to generate the tables of the DB and some lines to have data stored when you generate it. This is only a photo but you can see in the same folder a file named GenerateDatabase.txt with the code.



```
drop database if exists library;
create database if not exists library character set utf8 collate utf8_general_ci;
use library;
create table if not exists location(
    location_ID int auto_increment  not null,
    room        int                 not null,
    module      int                 not null,
    shelf       int                 not null,
    position    int                 not null,
    primary key (location_ID)
);
create table if not exists books(
    book_ID     int auto_increment  not null,
    title       varchar(30)         not null,
    isbn        long                not null,
    author      varchar(40)         not null,
    editorial   varchar(30),
    category    varchar(40),
    language    varchar(20),
    created_at  int                 not null,
    status      boolean             not null,
    location_ID int                 not null,
    net_price   float               default 20,
    /*vat         float               default 2,*/
    books_sell  int                 default 10,
    primary key (book_ID),
    foreign key (location_ID) references location(location_ID)
);
create table if not exists members(
    member_ID                   int auto_increment  not null,
    name                        varchar(20)         not null,
    surname1                    varchar(40)         not null,
    surname2                    varchar(40)         not null,
    nickname                    varchar(20)         not null,
    phone                       int(9)              not null,
    address                     varchar(50)         not null,
    password                    varchar(40)         not null,
    member_type                 varchar(1)          not null,
    next_allowed_reservation    date                not null,
    primary key (member_ID)
);
```

```sql
create table if not exists reservations(
    reservation_ID  int auto_increment  not null,
    book_ID         int unique          not null,
    member_ID       int                 not null,
    initialDate     date                not null,
    finalDate       date                not null,
    realFinalDate   date                not null,
    primary key (reservation_ID),
    foreign key (book_ID) references books(book_ID),
    foreign key (member_ID) references members(member_ID)
);
create table if not exists reservations_log(
    reservation_ID  int                 not null,
    book_ID         int                 not null,
    member_ID       int                 not null,
    initialDate     date                not null,
    finalDate       date                not null,
    realFinalDate   date                not null,
    primary key (reservation_ID)
);
create table if not exists cart(
    member_ID   int                 not null,
    book_ID     int                 not null,
    quantity    int                 not null,
    created_on  date                not null,
    primary key (member_ID,book_ID),
    foreign key (book_ID) references books(book_ID),
    foreign key (member_ID) references members(member_ID)
);
create table if not exists orders(
    order_ID    varchar(30)         not null,
    member_ID   int                 not null,
    book_ID     int                 not null,
    quantity    int                 not null,
    price       float               not null,
    order_date  datetime            not null,
    primary key (member_ID,book_ID,order_date),
    foreign key (book_ID) references books(book_ID),
    foreign key (member_ID) references members(member_ID)
);
```

```sql
insert into location(room,module,shelf,position) values(1,1,1,1);
insert into location(room,module,shelf,position) values(1,1,1,2);
insert into location(room,module,shelf,position) values(1,1,1,3);
insert into location(room,module,shelf,position) values(1,1,1,4);
insert into location(room,module,shelf,position) values(1,1,1,5);
insert into location(room,module,shelf,position) values(1,1,2,1);
insert into location(room,module,shelf,position) values(1,1,2,2);
insert into location(room,module,shelf,position) values(1,1,2,3);
insert into location(room,module,shelf,position) values(1,1,2,4);
insert into location(room,module,shelf,position) values(1,1,2,5);
insert into location(room,module,shelf,position) values(1,1,3,1);
insert into location(room,module,shelf,position) values(1,1,3,2);
insert into location(room,module,shelf,position) values(1,1,3,3);
insert into location(room,module,shelf,position) values(1,1,3,4);
insert into location(room,module,shelf,position) values(1,1,3,5);
insert into location(room,module,shelf,position) values(1,1,4,1);
insert into location(room,module,shelf,position) values(1,1,4,2);
insert into location(room,module,shelf,position) values(1,1,4,3);
insert into location(room,module,shelf,position) values(1,1,4,4);
insert into location(room,module,shelf,position) values(1,1,4,5);


insert into books values(1,"luces de bohemia",255,"ramon del valle inclan","coleccion austral","drama","español",2342,1,1,20,10);
insert into books values(2,"don quijote",324254,"juan de la cuesta","francisco de robles","novelas de aventuras","español",1615,1,2,20,10);
insert into books values(3,"don quijote2",324254,"juan de la cuesta","francisco de robles","novelas de aventuras","español",1616,1,3,20,10);
insert into books values(4,"don quijote3",324254,"juan de la cuesta","francisco de robles","novelas de aventuras","español",1617,1,4,20,10);
insert into books values(5,"don quijote4",324254,"juan de la cuesta","francisco de robles","novelas de aventuras","español",1618,1,5,20,10);

insert into members values(1,"Paco","Perez","Pons","pacopons",658568587,"Ramón y Cajal","pacopons","m",'');
insert into members values(2,"Pere","Perez","Pons","perepons",786676568,"Ramón y Cajal","perepons","l",'');
insert into members values(3,"Nacho","Perez","Pons","nacho",845268751,"Ramón y Cajal","nacho","l",'');
insert into members values(4,"Nacho","Perez","Pons","nacho1",452156985,"Ramón y Cajal","nacho1","m",'');
```

Server

- PHP version → PHP Version 7.3.9
- Apache version → Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.3.9
- Server version → 10.4.6-MariaDB

Functionalities

I have divided the project in two big sections and for each section I have three mini-sections:

PHP

PHP Project - Bronze

Ugly appearance, no styling

All HTML forms to enter data

All PHP files to capture information from the HTML forms and connect to the SQL database

Reservations with manual date introduction.

index_guest.php, index_member.php, index_librarian.php to give each user the appropriate capabilities

Action buttons can all be on the same page.

PHP Project - Silver

Basic styling with personal CSS or CSS libraries (bootstrap or materialize)

HTML & PHP code injection protection

Reservations with automatic date calculation (initial date, final date, x days penalty for returning books x days late)

Login validation form and opened sessions for users, to give them the appropriate capabilities.

Action buttons can be distributed on different pages.

PHP Project - Gold

Fine styling with personal CSS or CSS libraries (bootstrap or materialize)

Incorporate ebooks to the library.

Include images of the book covers.

Include images for book_types (paper, ebook).

Possibility of downloading ebooks with a maximum of 10 ebooks per user.

Possibility of borrowing a maximum of 3 books on paper per member.

Possibility of uploading one or multiple files by the librarian, through a form, to a chosen directory.

Action buttons distributed in a logical way (in the right context).

Autofill capabilities for update forms, so the user only has to modify the desired fields.

New functionalities unexpected by the customer.

Improved UI & UX (User experience)

AJAX

AJAX Project - Bronze

Put a clock on the main web page with the server time, updating it every second. Create the files "showTime.php" (to generate the clock data) and "showTime.js" (to present the clock data on screen). Integrate them into the "index.php" main page, under the <section id="clock">Clock goes here<section>.

Implement a shopping cart with the described functionality.

Implement a suggestion field for searching books by title or by author.

Update the technical manual, the user manual and the installation manual to incorporate the new functionalities.

AJAX Project - Silver

To avoid losing a potential customer, all the information contained within the shopping cart must be temporarily stored until the user finally processes the order, something that might happen on a different day. As an exercise, develop 4 different versions of the shopping cart storing its content with EACH of the following options:

- The PHP global variable $_SESSION

- The PHP global variable $_COOKIE

- The local storage in the browser

- A table in the SQL database

A practical application of this approach is to allow guests to add items to the shopping cart (storing items with $_COOKIE or LocalStorage). Then, when they log in, we can move those items into the $_SESSION variable or SQL Database.

AJAX Project - Gold

Guests can start adding items to the shopping cart before logging in with user/pwd (or autoregistration). Once logged in, they can process the order.

Every time the shopping cart is shown, prices in the shopping cart are refreshed from the original SQL tables (normally 'books') to make sure that the order is processed with the actual book price.

Implement the AJAX technique using the fetch API (async & wait).

Install GIT (with commander, integrated in Visual Studio Code, or as you prefer) and create at least three commits during your software development (bronze, silver, gold).

Write a document explaining the good and bad practices within your code, suggesting improvements. Imagine you are giving advice to yourself on how to develop the software of a similar project in the future.

***All the information in red are the functionalities that I couldn't finish on time. But are not difficult to implement if you want it in the future.**