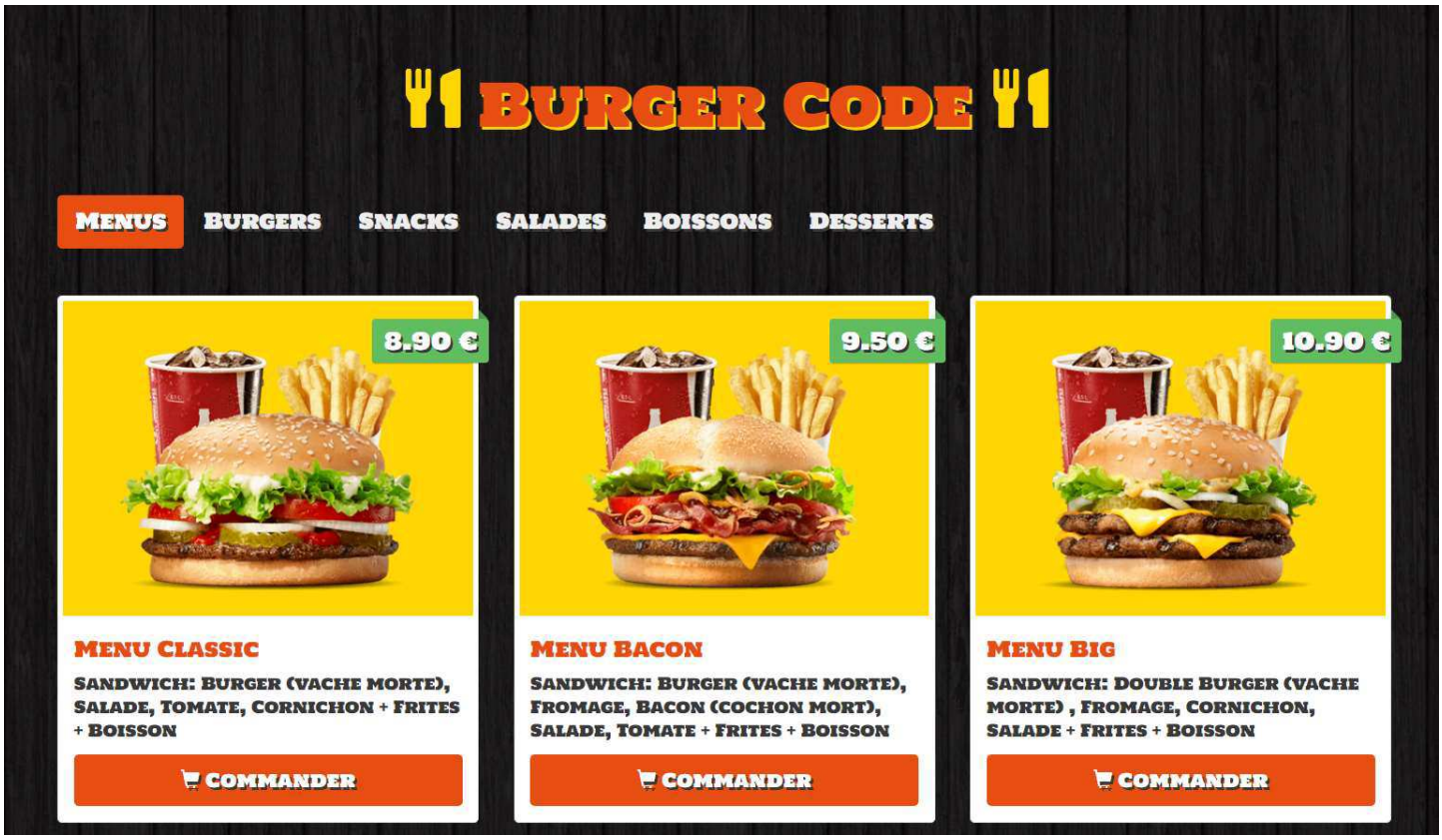
Fiche de Synthèse : Burger Code *Mai 2020*

Développement :
Burger Code est un site dynamique de commande de Burger et accompagnements, basé sur une base de données MySQL.
Visible à cette adresse : http://burgercode.lencodage.fr



Cadre :
Il s'agit d'un TP proposé dans le cadre du cours " Apprendre Php et MySQL et créer un site Web dynamique ", dans le cadre de la formation en ligne Udemy " Formation complète Developpeur Web ".

Support :
HTML/CSS, MySQL et Php, à l'aide de l'IDE NetBeans.

Contraintes / Organisation :
Il ne s'agit pas d'un vrai site bien sûr.
L'objectif de ce TP est d'être capable de manipuler une base de données MySQL, mais aussi la création d'un back office pour une gestion simplifiée de la base de données (et donc du site).

Difficultés rencontrées :
Pas de difficultés particulières.

Description résumée du développement

Le projet a été développé selon les étapes suivantes :
- création du site statique (html / css)
- création de la base de données
- connexion à la base de données
- création de l'admin
- admin : afficher un item

- admin : ajouter un item
- admin : modifier un item
- admin : supprimer un item
- rendre le site dynamique
- mise en ligne du site

1. Création du site statique (html / css)

Le préalable : avant que ça devienne dynamique, il faut d'abord le développer de manière statique, afin d'appliquer la mise en page adéquate.
Afin qu'il soit responsive (qu'il s'adapte à la taille de l'écran de l'utilisateur), Bootstrap 3 a été utilisé (la version 3 à cause des Glyphicons bien pratiques qui ont disparus à la version 4).
Pour la police d'écriture, Google est, comme souvent, notre ami en nous proposant un large choix, gratuitement et facilement : il suffit d'insérer le lien adéquat dans l'en-tête du site.

```
<head>
    <title>Burger Code</title>
    <meta charset='UTF-8'>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <!-- Latest compiled and minified JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <link href="http://fonts.googleapis.com/css?family=Holtwood+One+SC" rel="stylesheet" type="text/css">
    <link rel="stylesheet" href="css/styles.css">
```

Côté css, rien de particulier, un petit extrait :

```
body {
    background:  url(../images/bg.png);
}
.site {
    font-family: "Holtwood One SC", sans-serif;
}
.text-logo {
    font-family: "Holtwood One SC", sans-serif;
    color: #e7480f;
    text-shadow: 2px 2px #ffd301;
    font-size: 50px;
    padding: 40px 0px;
    text-align: center;
}
.text-logo .glyphicon {
    color: #ffd301;
    text-shadow: 0px 0px #ffd301;
}
.nav {
    margin-bottom: 40px;
}
.thumbnail img {
    background: #ffd301;
}
```

2. Création de la base de données

La création de la base de données s'est faite directement via PhpMyAdmin : 2 tables (catégories et items) avec simplement une clé étrangère dans la table items (Menu classique, fondant chocolat, etc.….) qui correspond à l'id de la catégorie (Menu, Burgers, … Desserts)

Les tables nous ont été fournies, il n'y a eu qu'à les importer.
Par la suite, quand il a été possible de modifier les items je me suis permise de détailler un peu plus les descritpions (mais c'est beaucoup plus tard...)

```
SELECT * FROM `items`
```

Profilage [Éditer en ligne] [Éditer] [Expliq

1 ▾ > >> | Tout afficher | Nombre de lignes : 25 ▾ Filtrer les lignes: Chercher dans cette table Trier sur l'index: Aucun(e)

Options

	id	name	description	price	image	category
Éditer Copier Supprimer	1	Menu Classic	Sandwich: Burger (vache morte), Salade, Tomate, Co...	8.9	m1.png	1
Éditer Copier Supprimer	2	Menu Bacon	Sandwich: Burger (vache morte), Fromage, Bacon (co...	9.5	m2.png	1
Éditer Copier Supprimer	3	Menu Big	Sandwich: Double Burger (vache morte) , Fromage, C...	10.9	m3.png	1
Éditer Copier Supprimer	4	Menu Chicken	Sandwich: Poulet Frit (cadavre plongÃ© dans une fr...	9.9	m4.png	1
Éditer Copier Supprimer	5	Menu Fish	Sandwich: Poisson (mort Ã©videmment, probablement ...	10.9	m5.png	1
Éditer Copier Supprimer	6	Menu Double Steak	Sandwich: Double Burger (double vache morte ?), Fr...	11.9	m6.png	1
Éditer Copier Supprimer	7	Classic	Sandwich: Burger (morceaux de cadavre de vache), S...	5.9	b1.png	2
Éditer Copier Supprimer	8	Bacon	Sandwich: Burger (morceaux de cadavre de vache), F...	6.5	b2.png	2
Éditer Copier Supprimer	9	Big	Sandwich: Double Burger (vache morte), Fromage, Co...	6.9	b3.png	2
Éditer Copier Supprimer	10	Chicken	Sandwich: Poulet Frit (poulet mort), Tomate, Salad...	5.9	b4.png	2

3. Connexion à la base de données

C’est là qu’arrive les choses sérieuses.
Un objet Database a été créé, afin de pouvoir se connecter et se déconnecter de la base de données.

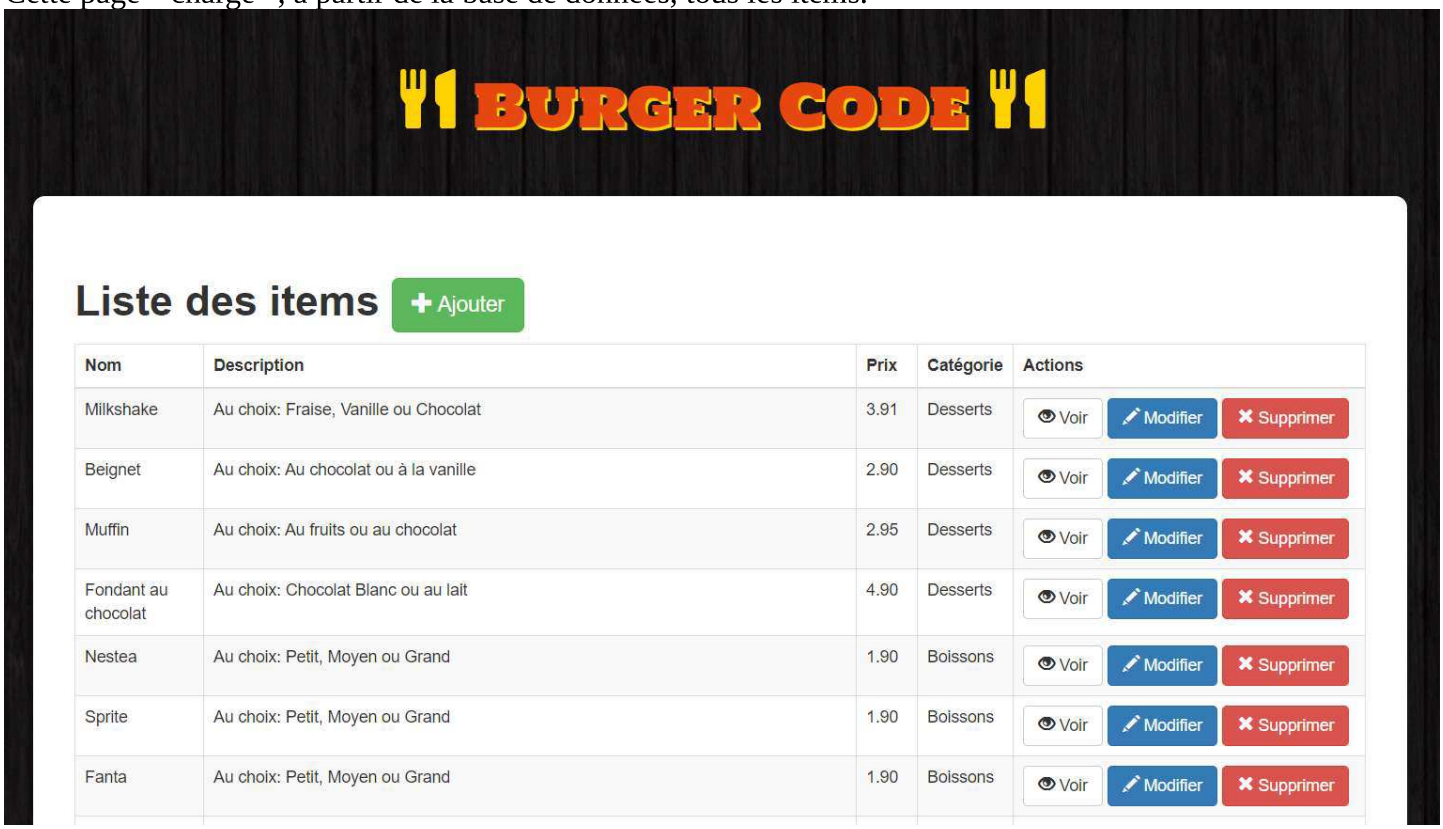
```php
<?php

class Database
{
    private static $dbHost = "localhost";
    private static $dbName = "burgercode";
    private static $dbUsername = "root";
    private static $dbUserpassword = "";

    private static $connection = null;

    public static function connect()
    {
        if(self::$connection == null)
        {
            try
            {
              self::$connection = new PDO("mysql:host=" . self::$dbHost . ";dbname=" . self::$dbName , self::$dbUsername, self::$dbUserpassword);
            }
            catch(PDOException $e)
            {
                die($e->getMessage());
            }
        }
        return self::$connection;
    }

    public static function disconnect()
    {
        self::$connection = null;
    }

}
?>
```

Vous remarquez que tous les identifiants de connexion ont été mis dans des variables, à l’extérieur de la fonction connect() : c’est bien plus pratique, quand il s’agira de mettre ce site en ligne, il sera facile de remplacer les identifiants de WampServer par ceux de mon provider.

4. Création de l’admin

Il s’agit donc du back office, c’est à dire la page qui permet d’admninistrer le site : on peut ajouter un item le modifier, le supprimer, le voir simplement.

Cette page “ charge ”, à partir de la base de données, tous les items.

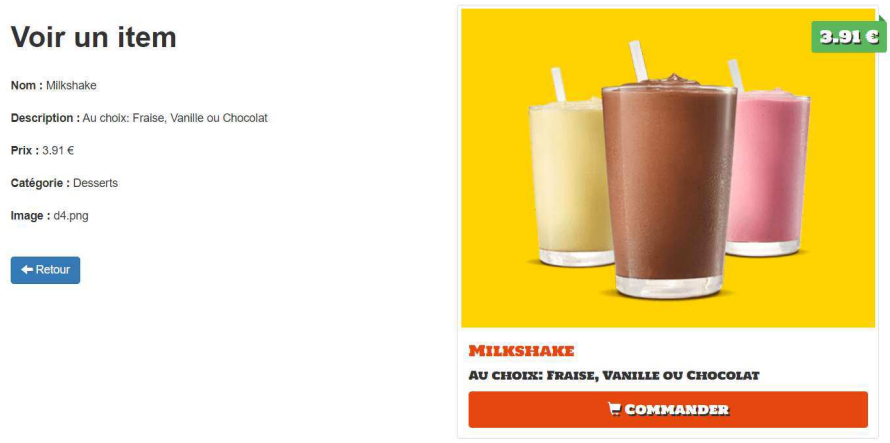


Pour ce faire, dans un fichier index.php du dossier Admin, un tableau a été créé en html, et grâce à php il a été rempli avec les informations récupérées dans la base de données avec une requête de sélection :

```
<tbody>
    <?php
    require 'database.php';
    $db = Database::connect();
    $statement = $db->query('SELECT items.id, items.name, items.description, items.price, categories.name AS category
                            FROM items LEFT JOIN categories ON items.category = categories.id
                            ORDER BY items.id DESC');
    while($item = $statement->fetch()) {
        echo '<tr>';
        echo '<td>' . $item['name'] . '</td>';
        echo '<td>' . $item['description'] .'</td>';
        echo '<td>' . number_format((float)$item['price'],2,'.','') . '</td>';
        echo '<td>' . $item['category'] . '</td>';
        echo '<td width="300">';
        echo '<a class="btn btn-default" href="view.php?id=' . $item['id'] . '"><span class="glyphicon glyphicon-eye-open">
        </span> Voir</a>';
        echo ' ';
        echo '<a class="btn btn-primary" href="update.php?id=' .$item['id'] .'"><span class="glyphicon glyphicon-pencil">
        </span> Modifier</a>';
        echo ' ';
        echo '<a class="btn btn-danger" href="delete.php?id=' . $item['id'] . '"><span class="glyphicon glyphicon-remove">
        </span> Supprimer</a>';
        echo '</td>';
        echo '</tr>';
    }
    Database::disconnect();

    ?>
```

5. Admin : afficher un item

Quand on clique sur le petit bouton “ voir ”, on est re-dirigé vers cette page (view.php) :



En Php, c’est la méthode $_GET qui est utilisée pour récupérer l’identifiant de l’item sur lequel on a cliqué.

```php
require 'database.php';

if(!empty($_GET['id'])) {
    $id = checkInput($_GET['id']);
}

$db = Database::connect();
$statement = $db->prepare('SELECT items.id, items.name, items.description, items.price, items.image, categories.name AS category
                           FROM items LEFT JOIN categories ON items.category = categories.id
                           WHERE items.id = ? ');

$statement->execute(array($id));
$item = $statement->fetch();
Database::disconnect();
```

Ensuite, on se “ contente ” de charger les éléments de la page avec les informations ainsi récupérées.

6. Admin : ajouter un item

Un clic sur le gros bouton vert Ajouter rediriger vers la page insert.php.

Pour cette page, il y a un certain nombre de vérifications à faire avant l'insertion dans la base de données : que tous les champs sont remplis, mais aussi que l'image proposée a le bon format, n'est pas trop lourde, qu'elle n'existe pas déjà, etc.…..

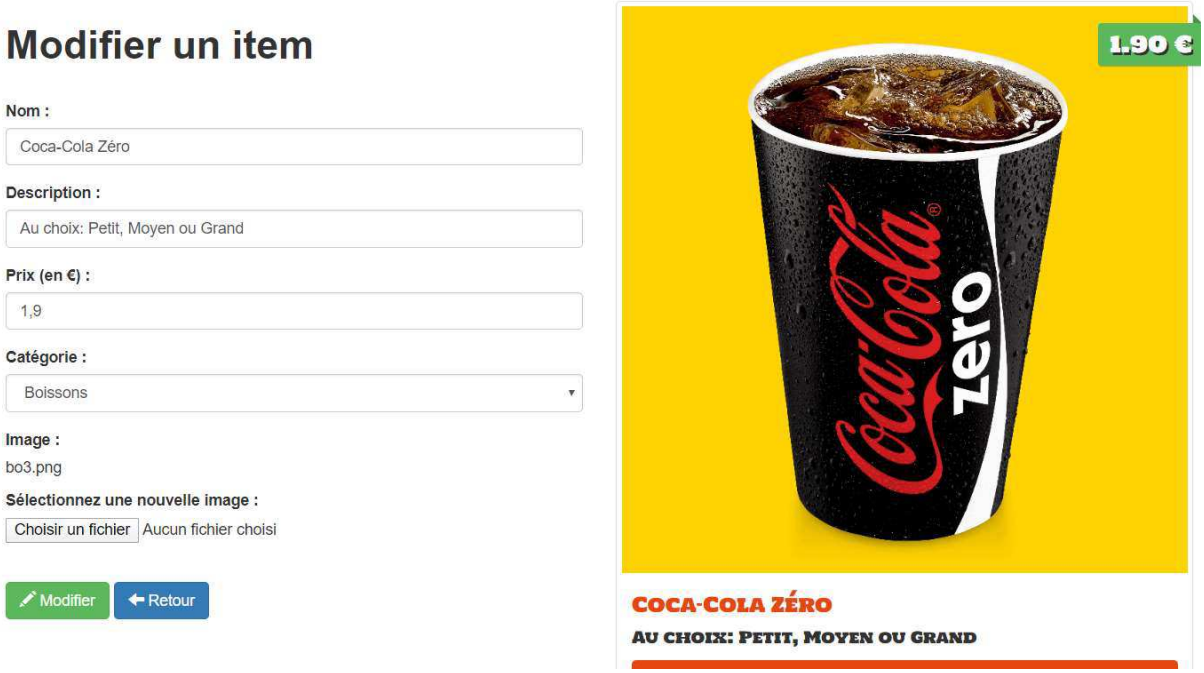
```php
if(!empty($_POST)) {
    $name = checkInput($_POST['name']);
    $description = checkInput($_POST['description']);
    $price = checkInput($_POST['price']);
    $category = checkInput($_POST['price']);
    $image = checkInput($_FILES['image']['name']);
    $imagePath = '../images/' . basename($image);
    $imageExtension = pathinfo($imagePath, PATHINFO_EXTENSION);
    $isSuccess = true;
    $isUploadSuccess = false;

    if(empty($name)) {
        $nameError = 'Ce champ ne peut pas être vide';
        $isSuccess = false;
    }
    if(empty($description)) {
        $descriptionError = 'Ce champ ne peut pas être vide';
        $isSuccess = false;
    }
    if(empty($price)) {
```

Et ensuite seulement, on peut insérer les données dans la base, puis revenir sur la page d'index de l'admin :

```php
if($isSuccess && $isUploadSuccess) {
    $db = Database::connect();
    $statement = $db->prepare('INSERT INTO items (name, description, price, category, image) VALUES (?,?,?,?,?)');
    $statement->execute(array($name, $description, $price, $category, $image));
    Database::disconnect();
    header("Location: index.php");
}
```

7. Admin : modifier un item

Lorsqu'on clique sur modifier un item, on est redirigé sur la page update.php.



Les informations concernant l'item que l'on a sélectionné s'affiche, et on peut donc les modifier et les enregistrer.
Je ne vous remet pas de code cette fois, c'est sensiblement le même que celui de l'ajout d'item, avec le chargement des informations en plus.

8. Admin : supprimer un item

Enfin, lorsque l'on clique sur Supprimer, on est redirigé vers une page delete.php :

C'est probablement ce qu'il y a de plus simple, une fois l'id de l'item récupéré, une simple supression de la base de données :

```php
require 'database.php';

if(!empty($_GET['id'])) {
    $id = checkInput($_GET['id']);
}

if(!empty($_POST)) {
    $id = checkInput($_POST['id']);
    $db = Database::connect();
    $statement = $db->prepare('DELETE FROM items WHERE id=? ');
    $statement->execute(array($id));
    Database::disconnect();
    header("Location: index.php");
}
```

9. Rendre le site dynamique

C'est à ce moment-là qu'on remplace plusieurs centaines de lignes de codes html, par … 71 lignes (en comptant le header, qui lui reste statique).
Déjà, on remplace les éléments 'en dur' de la barre de navigation par les catégories :

```php
<div class="container site">
    <h1 class="text-logo"><span class="glyphicon glyphicon-cutlery"></span> Burger Code <span class="glyphicon glyphicon-cutlery"></span>
    </h1>
    <?php
    require 'admin/database.php';
    echo '<nav>
        <ul class="nav nav-pills">';
    $db = Database::connect();
    $statement = $db->query('SELECT * FROM categories');
    $categories = $statement->fetchAll();
    foreach($categories as $category) {
        if($category['id'] == '1') {
            echo '<li role="presentation" class="active"><a href="#' .$category['id']. '" data-toggle="tab">' . $category['name']. '</a>
            </li>';
        } else {
            echo '<li role="presentation"><a href="#' .$category['id']. '" data-toggle="tab">' . $category['name']. '</a></li>';
        }
    }
```

Et ensuite, chaque item est placé : sur la page relative à sa catégorie d'une part, et ensuite 'en liste' :

```php
echo '</ul>
</nav>';
echo '<div class="tab-content">';
foreach($categories as $category) {
    if($category['id'] == '1') {
        echo '<div class="tab-pane active" id="' . $category['id']. '">';
    } else {
        echo '<div class="tab-pane" id="' . $category['id']. '">';
    }
    echo '<div class="row">';
    $statement = $db->prepare('SELECT * FROM items WHERE items.category = ?');
    $statement->execute(array($category['id']));

    while($item = $statement->fetch()) {
        echo '<div class="col-sm-6 col-md-4">
                <div class="thumbnail">
                    <img src="images/' . $item['image'] . '" alt="...">
                    <div class="price">' . number_format($item['price'],2,'.','') . ' €</div>
                    <div class="caption">
                        <h4>' . $item['name']. '</h4>
                        <p>' . $item['description']. '</p>
                        <a href="#" class="btn btn-order" role=button><span class="glyphicon glyphicon-shopping-cart"> </span>
                        Commander</a>
                    </div>
                </div>
            </div>';
    }
    echo '</div>
    </div>';
}
Database::disconnect();
echo '</div>';
```

10. Mise en ligne

Pour la mise en ligne, il a suffit de créer la base de données chez le provider (LWS dans mon cas), puis de modifier mon objet Database() avec les informations communiquées :

```php
class Database {

    private static $dbHost = "[redacted]";
    private static $dbName = "[redacted]";
    private static $dbUser = "[redacted]";
    private static $dbUserPassword = "[redacted]";

    private static $connexion = null;
```

Vous pouvez le constater ici : http://burgercode.lencodage.fr/ cela fonctionne parfaitement.

J'ai bien sûr sécurisé l'accès au back office grâce à des fichiers .htacess et .htpasswd. En ajoutant admin à l'adresse ci-dessus, vous pourrez vérifier la protection de l'accès.

Nom de fichier	Taille de ...	Type de fichier	Dernière modi...
..			
.htaccess	151	Fichier HTACC...	11/05/2020 19:...
.htpasswd	38	Fichier HTPASS...	11/05/2020 19:...
database.php	810	Fichier PHP	11/05/2020 14:...
delete.php	2 338	Fichier PHP	11/05/2020 11:...
index.php	3 554	Fichier PHP	08/05/2020 12:...
insert.php	6 968	Fichier PHP	11/05/2020 13:...
update.php	9 962	Fichier PHP	11/05/2020 12:...
view.php	4 035	Fichier PHP	08/05/2020 12:...