BRIEF

Création d’un site web dynamique qui interagit avec une base de données relationnelle en PHP

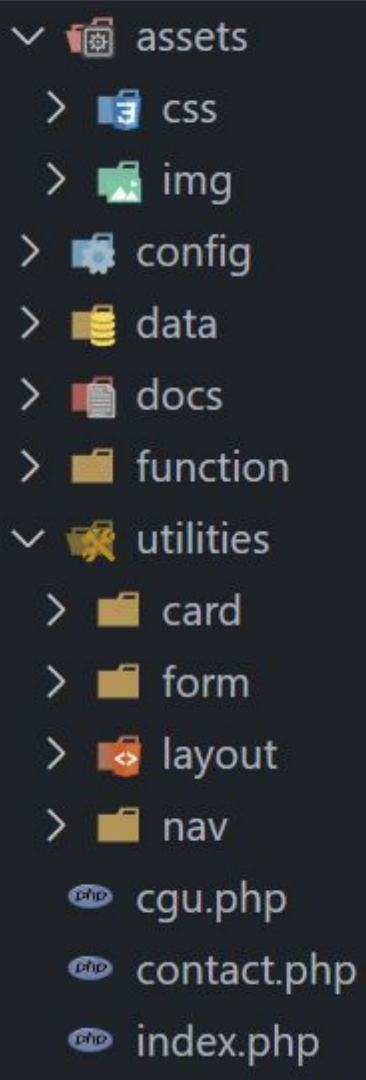
github.com/gpuberos/brief-medbdd

Réalisé par Guy-Philippe Uberos

Réorganisation de la structure du site

La structure en répertoire permet d'organiser les fichiers de manière logique et cohérente.
En séparant les fichiers en fonctionnalités, en stockant dans des sous-répertoires les éléments récurrents.

1. **Organisation** et **clarté**
2. **Maintenance facilitée**
3. **Réutilisation du code**
4. **Évolutivité**



Réflexion sur les données et les fonctionnalités

Il est crucial de réfléchir en profondeur aux données que nous devons stocker et aux fonctionnalités que le système doit prendre en charge.

01

COMPRÉHENSION DES BESOINS

Identifiez les besoins spécifiques du système et les informations stockées.

02

SCÉNARIOS D'UTILISATION

Pensez aux différentes actions que les utilisateurs effectueront.

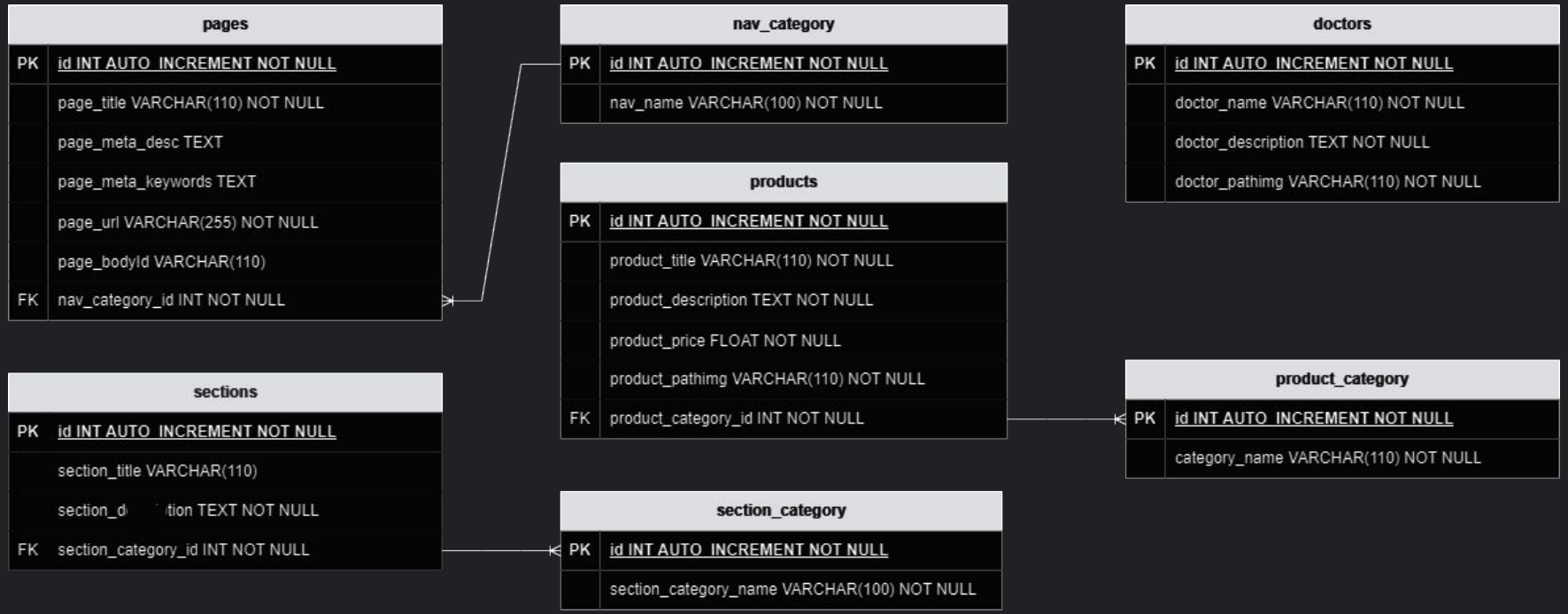
03

ANTICIPATION ET NORMALISATION

Prévoir les évolutions futures et comment organiser les données de manière à minimiser les redondances.

Création Schéma UML

- Pages et Catégories de Navigation : Many-to-One
- Sections et Catégories de section : One-to-Many
- Produits et Catégories de produits : One-to-Many



Création de la base de données

Utilisation du terminal SQL pour créer la structure de la base de données et insérer les données dans les tables.

Il était tout à fait possible d'utiliser PhpMyAdmin afin d'y insérer la requête de création de base de données et de l'exécuter.

```sql
CREATE DATABASE dbbrief_med;

USE dbbrief_med;

CREATE TABLE nav_category (
    id INT AUTO_INCREMENT NOT NULL,
    PRIMARY KEY (id),
    nav_name VARCHAR(100) NOT NULL
);

CREATE TABLE pages (
    id INT AUTO_INCREMENT NOT NULL,
    PRIMARY KEY (id),
    page_title VARCHAR(110) NOT NULL,
    page_meta_desc TEXT,
    page_meta_keywords TEXT,
    page_url VARCHAR(255) NOT NULL,
    page_bodyId VARCHAR(110),
    nav_category_id INT NOT NULL,
    FOREIGN KEY (nav_category_id) REFERENCES nav_category (id)
);

CREATE TABLE section_category (
    id INT AUTO_INCREMENT NOT NULL,
    PRIMARY KEY (id),
    section_category_name VARCHAR(100) NOT NULL
);

CREATE TABLE sections (
    id INT AUTO_INCREMENT NOT NULL,
    PRIMARY KEY (id),
    section_title VARCHAR(110) NOT NULL,
```

require_once () dans le header

On inclut les différents fichiers nécessaires au bon fonctionnement du site. La particularité de require_once() c'est si le fichier a déjà été inclus, il ne le sera pas une deuxième fois.

1. Fichiers de config des chemins par défaut et de la config de la base de données
2. Fichiers de fonctions Database, Header et Frontend.

```php
// Inclusion du fichier de configuration des chemins d'accès (Images)
require_once dirname(dirname(__DIR__)) . '/config/path.cfg.php';

// Inclusion du fichier de configuration de la base de données
require_once dirname(dirname(__DIR__)) . '/config/database.cfg.php';

// Inclusion du fichier contenant les fonctions relatives à la base d
require_once dirname(dirname(__DIR__)) . '/function/database.fn.php';

// Appel de la fonction getPDOlink() pour obtenir un lien vers la bas
$db = getPDOlink($config);

// Inclusion du fichier contenant les fonctions nécessaire à l'en-têt
require_once dirname(dirname(__DIR__)) . '/function/header.fn.php';

// Inclusion du fichier contenant les fonctions nécessaire au fronten
require_once dirname(dirname(__DIR__)) . '/function/frontend.fn.php';

// Appel de la fonction getPageInfo() pour obtenir les informations r
$pageInfo = getPageInfo($db);

?>

<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=
```

Fonction getPDOlink( )

La fonction **getPDOlink**() établit la connexion à la base de données en récupérant les informations du fichier de configuration **database.cfg.php** qui se trouve dans le répertoire **config**.

1. Construction du **DSN**
2. Tentative de connexion **Try**
3. Création nouvelle instance de l'objet **PDO** *
4. Choix mode de récupération par défaut
5. On retourne la connexion **$db**
6. Sinon **catch** l'erreur et on l'affiche

** PDO (PHP Data Objects) ou Objets de données PHP*

```php
// Cette fonction établit une connexion à la base de données
// Le paramètre $config est un tableau associatif qui contie
function getPDOlink($config)
{
    // Construction du DSN (Data Source Name) pour la connex
    // Le DSN comprend le type de base de données (mysql), l
    $dsn = 'mysql:dbname=' . $config['dbname'] . ';host=' .

    // Tentative de connexion à la base de données en utilis
    try {
        // Création d'une nouvelle instance de l'objet PDO a
        $db = new PDO($dsn, $config['dbuser'], $config['dbpa

        // Définition du mode de récupération par défaut pou
        // PDO::FETCH_ASSOC signifie que les résultats seron
        // Cela évite d'avoir des doublons dans les résultat
        $db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO:

        // Retour de l'objet PDO qui représente la connexion
        return $db;
    } catch (PDOException $e) {
        // En cas d'erreur de connexion, on utilise l'objet
        exit('Erreur de connexion à la base de données : ' .
    }
}
```

Requête préparées

C’est un modèle compilé pour le SQL qui peut être personnalisable en utilisant des variables comme paramètres.

- La requête n’est **analysée qu’une seule fois**, mais peut être **exécutée plusieurs fois** avec des paramètres différents
- La base de données optimise son plan d’exécution pour la requête, ce qui améliore les performances. (elle met en cache la requête).

01

Qu’est ce que c’est ?

C’est un modèle SQL optimisé, avec des paramètres personnalisé.

02

Pourquoi les utiliser ?

Economie de ressources (mieux optimisés), sécurité contre l’injection SQL.

Fonction getCurrentScriptPath( )

La fonction **getCurrentScriptPath**() retourne le chemin du script en cours d'exécution.

On utilise la variable superglobale **$_SERVER** et on lui ajoute la clé **SCRIPT_NAME** pour avoir le chemin du script PHP.

```php
// Fonction qui retourne le chemin du script en cours d'
function getCurrentScriptPath()
{
    return $_SERVER['SCRIPT_NAME'];
}
```

Fonction getPageInfo( )

La fonction **getPageInfo**() retourne les informations de la page dont l'URL correspond à l'URL courante.

1. Titre de la page (SEO)
2. Meta description (SEO)
3. Meta keywords (SEO)
4. BodyId (id à ajouter au body de la page)

```php
// Fonction qui retourne les informations de la page correspondant
function getPageInfo($db)
{
    // On récupère le chemin du script courant.
    $currentScriptPath = getCurrentScriptPath();

    // Prépare la requête SQL
    $sql = "SELECT page_title, page_meta_desc, page_meta_keywords

    // Prépare la requête SQL pour l'exécution.
    $sth = $db->prepare($sql);

    // Exécute la requête SQL.
    $sth->execute();

    // Récupère tous les résultats de la requête SQL et les stocke
    $pages = $sth->fetchAll();

    // On parcourt chaque page dans la section spécifiée.
    foreach ($pages as $page) {
        // On vérifie si l'URL courante est trouvée dans l'URL de
        // Cela permet de gérer le cas où l'utilisateur saisit di
        if (strpos($page['page_url'], $currentScriptPath) == fals
            // Si une correspondance est trouvée, on retourne les
            return $page;
        }
    }
}
```

Appel de la fonction getPageInfo( ).

Nous appelons la fonction **getPageInfo**(**$db**) en lui passant le paramètre **$db**.
Le résultat est stocké dans la variable **$pageInfo**.

Ensuite on affiche les valeurs, nous utilisons un short echo tag **<?= $pageInfo['page_title'] ?>**.

```php
// Appel de la fonction getPageInfo() pour obtenir les information
$pageInfo = getPageInfo($db);

?>

<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-sca

    <!-- Bootstrap Libraries -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/
    <script defer src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.

    <!-- CSS Custom -->
    <link rel="stylesheet" href="<?= CSS_PATH ?>/styles.css">

    <title><?= $pageInfo['page_title'] ?></title>
    <meta name="description" content="<?= $pageInfo['page_meta_des
    <meta name="keywords" content="<?= $pageInfo['page_meta_keywor
</head>

<body id="<?= $pageInfo['page_bodyId'] ?>" class="d-flex flex-colu
    <header>
        <?php require_once dirname(dirname(__DIR__)) . '/utilities
    </header>
```

Fonction displaySection( )

La fonction **displaySection($db, $sectionCategory)** affiche la section avec son titre et sa description en fonction de la catégorie que nous passons en paramètre.

```php
// Fonction qui affiche une section
function displaySection($db, $sectionCategory)
{
    // Prépare la requête SQL
    $sql = "SELECT sections.* FROM sections
    INNER JOIN section_category ON sections.section_category_id = section_c
    WHERE section_category.section_category_name = :category";

    // Prépare la requête SQL pour l'exécution.
    $sth = $db->prepare($sql);

    // Lie la valeur de $sectionCategory au paramètre nommé dans la requête
    $sth->bindParam(':category', $sectionCategory);

    // Exécute la requête SQL.
    $sth->execute();

    // Récupère tous les résultats de la requête SQL et les stockes dans $g
    $sections = $sth->fetchAll();

    // On vérifie si la section principale de la page courante existe.
    if (!empty($sections)) {
        // Si elle existe, on récupère les informations de cette section.
        foreach ($sections as $section) {
            // On affiche la section en utilisant echo. La section est comp
            // Le titre et le contenu sont récupérés à partir du tableau $s
            echo '
            <section class="mb-5">
                <h2 class="text-center fs-1 mb-4">' . $section['section_tit
                <p class="fs-22">' . $section['section_description'] . '</p
            </section>
            ';
        }
```

Appel de la fonction displaySection( )

Nous appelons la fonction **displaySection**(**$db, $sectionCategory**) affiche la section avec son titre et sa description.

On lui passe deux paramètres, en premier la connexion bdd **$db** et en second paramètre la catégorie de la section **$sectionCategory** que nous souhaitons afficher.

```php
<?php require_once __DIR__ . '/utilities/layout/header.ut.php'; ?>

<div class="container py-4">
    <!-- Section générique titre + paragraphe -->
    <?php displaySection($db, 'home'); ?>

    <?php require_once __DIR__ . '/utilities/card/doctors_card.ut.php'; ?>
</div>

<?php require_once __DIR__ . '/utilities/layout/footer.ut.php'; ?>
```

Fonction generateNavLinks( )

La fonction **generateNavLinks**(**$db, $navName**) génère un tableau de liens de navigation pour une catégorie de navigation spécifiée (navbar, footernav).

Elle retourne le titre de la page, son URL et mets la class active si le lien correspond à la page courante.

```php
// Fonction qui affiche une section
function displaySection($db, $sectionCategory)
{
    // Prépare la requête SQL
    $sql = "SELECT sections.* FROM sections
    INNER JOIN section_category ON sections.section_category_id = section_c
    WHERE section_category.section_category_name = :category";

    // Prépare la requête SQL pour l'exécution.
    $sth = $db->prepare($sql);

    // Lie la valeur de $sectionCategory au paramètre nommé dans la requête
    $sth->bindParam(':category', $sectionCategory);

    // Exécute la requête SQL.
    $sth->execute();

    // Récupère tous les résultats de la requête SQL et les stockes dans $g
    $sections = $sth->fetchAll();

    // On vérifie si la section principale de la page courante existe.
    if (!empty($sections)) {
        // Si elle existe, on récupère les informations de cette section.
        foreach ($sections as $section) {
            // On affiche la section en utilisant echo. La section est comp
            // Le titre et le contenu sont récupérés à partir du tableau $s
            echo '
            <section class="mb-5">
                <h2 class="text-center fs-1 mb-4">' . $section['section_tit
                <p class="fs-22">' . $section['section_description'] . '</p
            </section>
            ';
        }
```

Appel de la fonction generateNavLinks( )

Nous utilisons la fonction **generateNavLinks**(**$db, $navName**) pour récupérer les liens de navigation associés à une catégorie depuis la base de données.

Ensuite, la boucle **foreach** parcourt chaque élément du tableau **$navbarLinks** et génère un élément de liste avec un lien. Si le lien correspond à la page actuelle, nous ajoutons la classe **active**.

```php
<nav class="navbar navbar-expand-lg p-0 bg-light bg-gradient bg-opacity-50
    <div class="container-fluid">
        <a class="navbar-brand" href="/"><img src="<?= IMG_PATH ?>/wonderl
        <button class="navbar-toggler" type="button" data-bs-toggle="colla
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav gap-2 fs-3">
                <?php
                $navbarLinks = generateNavLinks($db, 'navbar');
                foreach ($navbarLinks as $key => $value) :
                ?>
                    <li class="nav-item">
                        <a href="<?= $value['link_url'] ?>" class="nav-lin
                    </li>
                <?php endforeach; ?>

            </ul>
        </div>
    </div>
</nav>
```

Footer

```php
<ul class="nav justify-content-center pb-3 mb-3 text-white">
    <?php
    $footerLinks = generateNavLinks($db, 'footernav');
    foreach ($footerLinks as $key => $value) : ?>
        <li class="nav-item">
            <a href="<?= $value['link_url'] ?>" class="nav-link px-2 text-
        </li>
    <?php endforeach; ?>
</ul>
```

Fonction findAllDatas( )

La fonction **findAllDatas($db, $sql)** est générique et permet d'exécuter n'importe quelle requête SQL pour récupérer des données de la base de données.

Elle prend en premier paramètre la connexion à la bdd **$db** et en second la requête SQL **$sql**.

Cette fonction est utile pour éviter la duplication de code.

```php
// Fonction qui récupère tous les résultats de la base de données.
function findAllDatas($db, $sql)
{
    // Prépare la requête SQL pour l'exécution.
    // La méthode prepare() est utilisée pour préparer la requête SQL pour
    // Elle retourne un objet PDOStatement qui est stocké dans la variable
    $sth = $db->prepare($sql);

    // Exécute la requête SQL.
    // La méthode execute() est utilisée pour exécuter la requête SQL prép
    $sth->execute();

    // Récupère tous les résultats de la requête SQL et les stocke dans $r
    // La méthode fetchAll() est utilisée pour récupérer tous les résultat
    // Elle retourne un tableau associatif de tous les résultats qui sont
    $result = $sth->fetchAll();

    // Retourne les résultats récupérés.
    // La fonction retourne le tableau associatif $result qui contient tou
    return $result;
}
```

Appel de la fonction findAllDatas( )

Nous appelons la fonction **findAllDatas($db, $sql)** qui nous retourne dans **$products** le résultat de la requête $productsQuery.

On utilise ensuite une boucle **foreach** pour générer à chaque itération une carte produit. On utilise la même fonction pour les cartes docteurs.

```
<section>
    <div class="row row-cols-1 row-cols-md-4 row-gap-4">
        <?php
        // La requête SQL est stockée dans la variable $productsQuery puis
        $productsQuery = "SELECT products.*, product_category.category_nam
        $products = findAllDatas($db, $productsQuery);

        foreach ($products as $product) :
        ?>
            <div class="col">
                <div class="card h-100 text-center rounded-0">
                    <img src="<?= PRODUCTS_IMG_PATH . $product['product_pa
                    <div class="card-body">
                        <h5 class="card-title"><?= $product['product_title
                        <p><span class="badge rounded-pill text-bg-light">
                        <p class="card-text"><?= $product['product_descrip
                    </div>
                    <div class="card-footer pb-4">
                        <p class="card-text"><strong>Prix : <?= $product['
                        <a href="#" class="btn bg-blue text-white rounded-
                    </div>
                </div>
            </div>
        <?php endforeach; ?>

    </div>
</section>
```

Documentations

Dans le répertoire **docs** se trouve les différentes documentations (référentiel) au format markdown.

- **STRUCTURE.md** : description des différents fichiers et répertoires.
- **DATABASE.md** : documentation sur la base de données.
- **FUNCTION.md** : documentation détaillée sur les fonctions utilisées sur le site.

```
# Structure des dossiers

## config

Contient les fichiers de configuration du site

- `database.cfg.php` : Contient les informations qui sont nécessaires pour
- `path.cfg.php` : Contient les chemins d'accès pour CSS, pour les images

Ces scripts sont inclus dans les pages du site en utilisant `require_once`

## data

- `dbbrief_med.sql` : Contient le dump SQL pour créer la base de données.

## docs

Contient les différentes documentations :
- `DATABASE.md` : Documentation sur la base de données
- `FUNCTION.md` : Documentation sur les fonctions utilisées
- `STRUCTURE.md` : Documentation descriptif des différents fichiers.

Répertoire `uml` contient le fichier source du schéma UML de la base de do
images exportés du schéma UML.

## function

Répertoire contenant des fichiers de fonctions qui sont utilisés à travers
spécifiques comme la manipulation de données, l'affichage de données...
Ces scripts sont inclus dans les pages en utilisant `require_once`.

## utilities
```

Merci pour votre attention

Des questions ?