

Rapport de Stage - BTS SIO SLAM

DEGRAVE MORGAN

Entreprise : MAGASIN POUBEAU

Tuteur de stage : Anton COVU

Période de stage : 6 janvier au 14 février 2024

Formation : BTS Services Informatiques aux Organisations (SIO)

Option Solutions Logicielles et Applications Métier (SLAM)

Introduction

Dans le cadre de ma formation en BTS Services Informatiques aux Organisations, option SLAM, j'ai effectué un stage au sein de l'entreprise POUBEAU. Ce stage a eu pour objectif de mettre en pratique les compétences acquises en cours, notamment en développement web et en gestion de projets informatiques.

Les missions qui m'ont été confiées concernaient principalement la création et la gestion de sites web en utilisant les technologies Javascript, SQL, HTML et CSS, PHP ainsi qu'AJAX et le logiciel Laragon. De plus, j'ai eu l'opportunité d'explorer divers aspects de la gestion de bases de données, de travailler en projet avec une équipe et de participer à l'amélioration de la qualité des projets web ainsi que d'utiliser un nouveau framework, celui de LARAVEL.

Présentation de l'entreprise

Nom de l'entreprise : POUBEAU

L'entreprise POUBEAU est spécialisée dans la vente de produits principalement pour pâtisser depuis 1959, l'entreprise possède un site web pour la vente en ligne. Au sein de cette entreprise, j'ai intégré le service informatique, plus spécifiquement l'équipe chargée du développement des solutions logicielles et des applications métier.

SOMMAIRE

I - Objectifs du stage

II - Deroulement du stage

III - Présentation d'une mission

Page 7 à la 20

IV - Conclusion

V- Annexes

Objectifs du stage

En globalité l'objectif de mon stage a été de créer un moyen de renseigner le stock des produits pour des commandes en ligne entre les 3 magasins physiques, si par exemple dans une commande en ligne il manque tel quantité de tel produit alors l'utilisateur de l'application web pourra faire une demande de stock qui s'enverra sur la discussion teams qui regroupe les 3 magasins différents. Ce qui a pour but d'au lieu de recommander du stock pour le magasin A, si le magasin B ou C possède le produit de tout simplement le transférer dans le magasin A.

Les principaux objectifs de mon stage étaient les suivants :

- Mettre en pratique les compétences acquises en développement web (HTML, CSS, JavaScript, PHP, MySQL) en utilisant **Visual Studio Code et Laragon**.
 - Comprendre et appliquer les méthodes de gestion de projet.
 - Développer des solutions web en réponse à des besoins clients spécifiques.
 - Améliorer mes compétences en autonomie et en travail d'équipe dans un contexte professionnel.
-

Déroulement du stage

1. Laravel et Laragon

Dans un premier temps on a utilisé pour mettre à profit notre travail le framework php Laravel :

Laravel est un framework PHP open-source qui facilite le développement d'applications web. Il suit le modèle MVC (Model-View-Controller) et offre une structure et des outils robustes pour le développement rapide et efficace. Voici quelques caractéristiques clés de Laravel :

1. **Eloquent ORM** : Un système de mapping objet-relationnel (ORM) qui permet de travailler facilement avec les bases de données.
2. **Blade** : Un moteur de templates simple et puissant qui permet de créer des vues dynamiques.
3. **Routing** : Un système de routage intuitif pour définir les chemins d'URL.
4. **Middleware** : Une façon de filtrer les requêtes HTTP et de les gérer avant qu'elles n'atteignent les contrôleurs.
5. **Artisan** : Une interface en ligne de commande avec divers outils pour automatiser les tâches courantes.
6. **Migrations** : Un système de contrôle de version pour gérer les modifications de la base de données.

Laravel est apprécié pour sa simplicité, son élégance et sa communauté active, ce qui en fait un choix populaire pour les développeurs web.

Ainsi que le logiciel Laragon pour héberger un serveur local :

Laragon est un environnement de développement rapide et puissant, conçu pour les développeurs web. Il est portable, isolé et universel, ce qui signifie qu'il peut être utilisé pour différents langages de programmation comme PHP, Node.js, Python, Java, Go et Ruby2.

Voici quelques caractéristiques clés de Laragon :

1. **Installation facile** : Laragon est simple à installer et à utiliser grâce à son interface conviviale.
2. **Portabilité** : Vous pouvez emporter Laragon sur une clé USB et l'utiliser sur n'importe quel ordinateur sans avoir à le réinstaller.
3. **Isolation** : Chaque projet est isolé, ce qui évite les conflits entre les versions de logiciels.
4. **Productivité** : Laragon permet de gagner du temps en automatisant de nombreuses tâches courantes, comme la création de serveurs virtuels et l'ajout de nouvelles versions de logiciels.
5. **Flexibilité** : Vous pouvez facilement ajouter des packages et des extensions en utilisant la fonction d'ajout rapide.

Laragon est particulièrement apprécié pour sa rapidité et sa capacité à simplifier le développement local, ce qui en fait un outil précieux pour les développeurs web.

Voici un read-me qu'on a écrit :

étape de déploiement:

READ ME. MD

```
## Prérequis Techniques
- **Serveur Web** : Apache
- **Base de données** : MySQL 5.7
- **PHP** : 7.4
- **Composer** : 1.10.1
- **Node.js & npm** : v14

## Étapes de Déploiement
1. Cloner le dépôt :
  'git clone https://github.com/HugoCovu/depannageinternet'
2. **Installer les dépendances** :
  'composer install'
3. **Installer le node package manager** :
  'npm install'
4. **Configurer les variables d'environnement** :
  Modifier le fichier '.env' pour la configuration de la base de données.
  renommer le .env.example en .env et y mettre les variables d'environnement en faisant
  php artisan key:generate
5. **Exécuter les migrations de la base de données** :
  'php artisan make:migration' puis 'php artisan migrate'
6. **Lancer le serveur de développement** :
  'php artisan serve'
7. **Compiler les assets** :
  'npm run build'
8. **Lancer votre navigateur et se rendre sur l'ip du serveur de développement** :
  Exemple : 127.0.0.1:8000 ou alors localhost:8000
9. ** Merci pour votre lecture, à bientôt sur le github
```

Je suis arrivé en cours de projet, c'est-à-dire que j'ai rejoint 2 semaines après son début.

En arrivant j'ai directement programmé en javascript, j'ai bien appris le système de childNodes et de parentNode que je ne connaissais pas.

En JavaScript, `parentNode` et `childNodes` sont des propriétés utilisées pour naviguer et manipuler le DOM (Document Object Model).

1. `parentNode` : Cette propriété renvoie le nœud parent d'un nœud donné. Par exemple, si vous avez un élément enfant (comme un paragraphe `<p>`), vous pouvez obtenir son élément parent (comme une division `<div>`) en utilisant `parentNode`. Voici un exemple :

Exemple :

```
let enfant = document.getElementById('enfant');
```

```
let parent = enfant.parentNode;
```

```
console.log(parent); // Affiche l'élément parent
```

2. `childNodes` : Cette propriété renvoie une collection de tous les nœuds enfants d'un nœud donné, y compris les éléments, les commentaires et les espaces vides (textes vides). Vous pouvez accéder à chaque nœud enfant en utilisant un index, comme un tableau. Voici un exemple :

```
let parent = document.getElementById('parent');
```

```
let enfants = parent.childNodes;
```

```
for (let i = 0; i < enfants.length; i++) {
```

```
    console.log(enfants[i]); // Affiche chaque nœud enfant
```

```
}
```

Dans un premier temps j'ai fini la sidebar

Détails produits :

Référence : 122211
Produits : Mini gouttes de chocolat noir 500g - Les gourmandises de Loulou
Quantité : 2 ☒

Référence : 4654
Produits : Caraïbe 66 % 3 kg - Chocolat noir à pâtisser Valrhona
Quantité : 3 ☒

Référence : 157007
Produits : Barre à mâcher Tête brûlée goût cola - 150 pièces
Quantité : 1 ☐

Référence : 122099
Produits : Brisures de daim 150g - Les gourmandises de Loulou
Quantité : 2 ☐

Référence : 122094
Produits : Eclats de Kit Kat 120g - Les gourmandises de Loulou
Quantité : 2 ☐

Référence : 122097
Produits : Brisures de spéculoos 120g - Les gourmandises de Loulou
Quantité : 2 ☐

Référence : 122095
Produits : Eclats de lion 120g - Les gourmandises de Loulou
Quantité : 2 ☐

Référence : 122096
Produits : Brisures d'oreo 120g - Les gourmandises de Loulou
Quantité : 2 ☐

Référence : 122101
Produits : Billes de crunch 120g - Les gourmandises de Loulou
Quantité : 2 ☐

Référence : 122098

Référence : 157029

+ 2
-

Mini gouttes de chocolat noir 500g - Les gourmandises de Loulou

+ 3
-

Caraïbe 66 % 3 kg - Chocolat noir à pâtisser Valrhona

Envoyer

J'ai affiché 2 svg l'un étant le plus + et l'autre le moins pour pouvoir grâce à du javascript.

The image shows a close-up of a product list element. It features a quantity input field with a plus sign and a minus sign, and a text area for the product name. The text area contains the text "Mini gouttes de chocolat noir 500g - Les gourmandises de Loulou". The input field has a value of 2. The text area is highlighted with a dashed red border.

chaque produit est dans un li qui eux sont dans un ul,

The image shows a button labeled "Envoyer" (Send). The button is rectangular with a light gray background and a thin black border. It is positioned below a list of products.

Réalisation d'un input envoyer qui permet d'afficher un modal similaire à l'interface de microsoft teams pour l'envoi d'un message pour la gestion des stock

Ce bouton possède la capacité d'afficher un modal, de mettre un blur pour l'arrière-plan donc sur le body, récupérer des informations comme la quantité modifiée le nom et la référence du produit dans la sidebar et les implantés dans le textarea.

une boucle pour permettre l'insertion des données de la référence de la quantité et le nom du produit dans un textarea avec des childNodes.

Voici le code du bouton :

```
sendProductsButton.addEventListener("click", function() {  
  teamsModal.classList.add("active");  
  bodyContent.classList.add("blur");  
  let cardQuantityContainer = document.getElementsByClassName("countQuantityContainer");  
  let productNameContainer = document.getElementsByClassName("productListElement");  
  
  for(let q = 0; q < cardQuantityContainer.length; q++)  
  {  
    let productListReference = productNameContainer[q].dataset.reference;  
    console.log(cardQuantityContainer[q].childNodes[0].innerText);  
    textarea.value += cardQuantityContainer[q].childNodes[0].innerText + " x - ";  
    textarea.value += productListReference + " - ";  
    console.log(productNameContainer[q].childNodes[2].innerText);  
    textarea.value += productNameContainer[q].childNodes[2].innerText + "\n";  
  }  
});
```

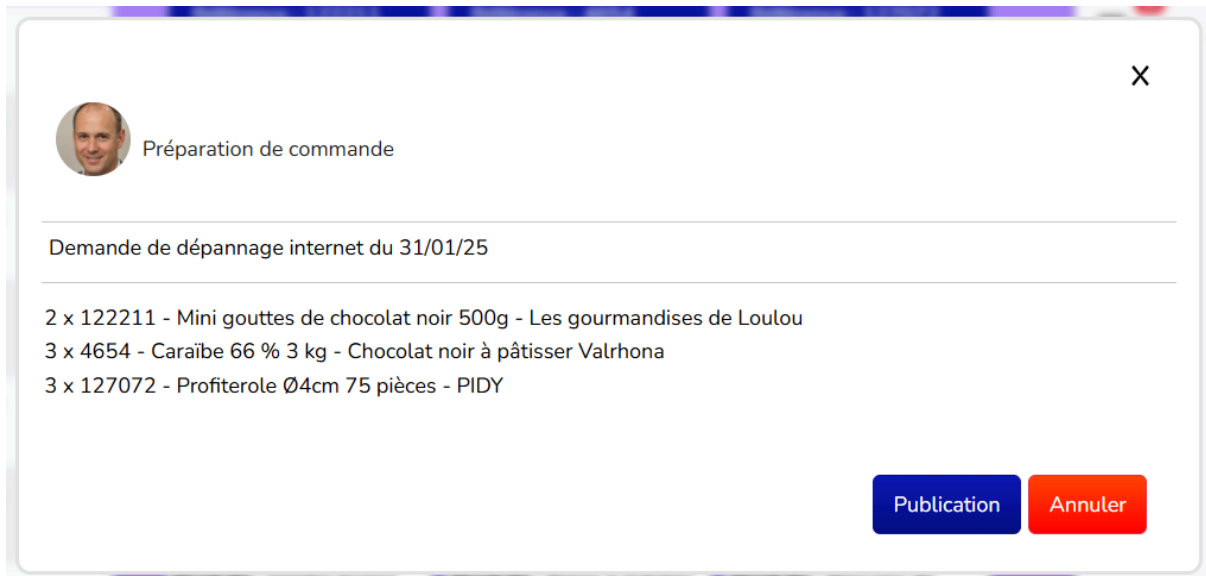
Ce script ajoute un événement au bouton sendProductsButton pour afficher un modal avec le contenu à publier lorsqu'on clique dessus. Si le modal de succès est actif, il le masque. Ensuite, il active le modal principal et applique un effet de flou au contenu du corps de la page. Il boucle ensuite à travers les conteneurs de quantité, ajoute les informations de produits à la zone de texte et ajuste la hauteur de la zone de texte en conséquence.


Création d'une constante pour l'automatisation du changement de la taille du textarea du modal teams content en fonction du nombre de produits sélectionnés et empêcher d'avoir On a aussi fait une fonction adjustHeight, qui permet d'augmenter en fonction du nombre de produits, la taille du textarea pour éviter d'afficher la scrollbar pour une question de pratique.

et d'esthétique pour ajuster la taille du modal en fonction du nombre de produits choisis :

```
const adjustHeight = (el) => {  
  el.style.height = 'auto';  
  el.style.height = el.scrollHeight + 'px';  
};
```

Voici l'apparence du modal reprenant le design de microsoft teams :




Préparation de commande
X

Demande de dépannage internet du 31/01/25

2 x 122211 - Mini gouttes de chocolat noir 500g - Les gourmandises de Loulou
 3 x 4654 - Caraïbe 66 % 3 kg - Chocolat noir à pâtisser Valrhona
 3 x 127072 - Profiterole Ø4cm 75 pièces - PIDY

Publication
Annuler

L'image ici est pour le moment une image qui a pour source le site :

"thispersondoesntexist.com" qui change à chaque fois que la page est rechargée, c'est la photo d'une personne qui n'existe pas, ici on viendra plus tard y placer la vraie photo de profil de l'utilisateur ainsi que son email au lieu de "Préparation de commande" grâce à l'api TEAMS de microsoft.

L'objet du message restera le même, ça sera une demande de dépannage internet sauf que la date change c'est à dire que ça sera la date du jour du système grâce à ce code : `{{date("d/m/y")}}`.

Le textarea est en disabled et dans le scss un "resize none" comme ça l'utilisateur ne peut pas y toucher car il n'en n'aura tout simplement pas besoin.

Un input "Annuler" qui permet d'enlever le blur, de désactiver le modal donc de le faire disparaître et de remettre la valeur du textarea vide comme ça si l'utilisateur revient dans le modal les précédentes données n'apparaîtront pas :

```

cancelationButton.addEventListener("click", function() {
    teamsModal.classList.remove("active");
    successModalContent.classList.remove("active");
    teamsContentToPublish.classList.remove("active");
    bodyContent.classList.remove("blur");
    textArea.value = "";
});

```

Utilisation d'un token csrf pour le bouton de publication.

```

<data type="hidden" name="csrf-token" value="{{ csrf_token() }}" id="token">

```


Dans le bouton de publication :

```
sendMessageButton.addEventListener("click", function() {
  let url = "/send-message";
  textArea.value + "<br>";
  teamsContentToPublish.classList.remove("active");
  loader.style.display = 'block';

  fetch(url, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Accept': 'application/json',
      'X-CSRF-TOKEN': token
    },
    body: JSON.stringify({
      header: header,
      date: formattedDate,
      message: textArea.value,
    })
  })
  .then(response => {
    if (!response.ok) {
      throw new Error('Erreur de réseau');
    } return response.json();
  })
})
```

```

.then(jsonResponse => {
    if (jsonResponse.success) {
        console.log("Le message a été envoyé avec succès.");
        teamsContentToPublish.classList.remove("active");
        successModalContent.classList.add("active");
        textArea.value = "";
        loader.style.display = 'none';
        while (sidebar.childNodes[3].firstChild) {
            sidebar.childNodes[3].removeChild(sidebar.childNodes[3].firstChild);
        }
        toggleSidebar();
        uncheckAllCheckboxes();
    } else {
        console.log("L'envoi du message a échoué.")
    }
})
});

```

Le bouton sendMessageButton donc le bouton de publication, permet, lors d'un click de celui-ci d'exécuter tout ce qui suit :

En parlant de ce qui suit, on a tout d'abord la déclaration d'une variable "url" qui est définie pour l'endpoint [/send-message](#). Le modal teams est retiré de l'affichage . Un loader (indicateur de chargement) est affiché.

Ensuite on a l'envoi des données, une requête fetch est envoyée à l'url définie avec la méthode POST. Les en-têtes spécifient que les données envoyées sont au format JSON et incluent un jeton CSRF pour la sécurité. Les données envoyées incluent l'en-tête, la date formatée, et le contenu du textArea.

Il y a aussi un contrôle de l'envoi, c'est-à-dire que pendant le loader si la réponse de la requête n'est pas correcte, alors une erreur est levée. Sinon, la réponse JSON est traitée.

Succès ou échec de l'envoi : si l'envoi du message est un succès, un message de confirmation est affiché, la zone de texte réinitialisée, le loader est masqué, et certains éléments de l'interface utilisateur sont mis à jour. Si l'envoi échoue, un message d'erreur est affiché dans la console.

Entre autres, ce code gère l'envoi d'un message via une requête POST, met à jour l'interface utilisateur en conséquence et affiche des messages de succès ou d'échec selon le résultat.

Tout dans le textarea est transformé en image car on avait un problème avec l'indentation des lignes. Donc on a créé un controller TextToImage.

TextToImage controller :

```
class TextToImageController extends Controller
{
    public function getImage($message)
    {
        $text = $message;

        $width = 800;
        $font = 'police/Roboto-Regular.ttf';
        $fontSize = 15;

        $textHeight = 25;
        $bottomPadding = 20; // Espace supplémentaire pour la dernière ligne

        $lines = explode("\n", wordwrap($text, ($width / $fontSize) * 2, "\n", true));
        $lineHeight = $textHeight;
        $height = (count($lines) * $lineHeight) + ($lines[count($lines)-1] ? $bottomPadding : 0);

        $image = imagecreatetruecolor($width, $height);

        $white = imagecolorallocate($image, 255, 255, 255);
        $black = imagecolorallocate($image, 0, 0, 0);
        imagefill($image, 0, 0, $white);

        $y = $textHeight;

        foreach ($lines as $line)
        {
            imagettftext($image, $fontSize, 0, 10, $y, $black, $font, $line);
            $y += $lineHeight;
        }

        // Capture du contenu binaire de l'image
        ob_start();
        imagepng($image);
        $imageData = ob_get_contents();
        ob_end_clean();

        // Détruire l'image pour libérer la mémoire
        imagedestroy($image);

        return $imageData;
    }
}
```

Dans un premier temps, on déclare la classe qui hérite de Controller. La méthode getImage prend un message en paramètre et génère une image avec ce texte.

Ensuite le message est stocké dans la variable \$text, les dimensions de l'image sont définies (\$width), ainsi que la police d'écriture (\$font) et la taille de la police (\$fontSize).

Le message est ensuite découpé en lignes pour s'adapter à la largeur de l'image. La hauteur de l'image est calculée en fonction du nombre de lignes et d'un espace supplémentaire pour la dernière ligne si nécessaire.

Une image vide est créée avec les dimensions calculées.

Chaque ligne de texte est ajoutée à l'image en noir, avec un espacement vertical défini par `lineHeight`.

Le contenu binaire de l'image est capturé. L'image est ensuite détruite pour libérer la mémoire.

Les données de l'image sont renvoyées sous forme binaire.

En résumé, ce code génère une image contenant un texte donné, en ajustant la largeur, la police et la taille du texte, puis retourne les données de cette image.

Le TransformTextToImg Controller :

```
class TransformTextToImg extends Controller
{
    public function textToImage(Request $request)
    {
        $text = $request->input('text', 'Hello, world!');

        // Création de l'image
        $width = 800;
        $height = 100;
        $image = imagecreatetruecolor($width, $height);

        // Couleurs
        $white = imagecolorallocate($image, 255, 255, 255);
        $black = imagecolorallocate($image, 0, 0, 0);

        // Remplir le fond en blanc
        imagefill($image, 0, 0, $white);

        // Écrire le texte
        imagestring($image, 5, 10, 40, $text, $black);

        // Retourner directement l'image
        return Response::stream(function() use ($image) {
            imagepng($image);
            imagedestroy($image);
        }, 200, [
            'Content-Type' => 'image/png',
            'Cache-Control' => 'no-cache, no-store, must-revalidate',
            'Pragma' => 'no-cache',
            'Expires' => '0'
        ]);
    }
}
```

La classe TransformTextToImg hérite de Controller. La méthode textToImage prend une requête et génère une image à partir du texte fourni.

Le texte à convertir en image est récupéré à partir de la requête, avec une valeur par défaut "Hello World"

Une image vide est créée avec une largeur de 800 pixels et une hauteur de 100 pixels.

L'image est retournée directement en tant que flux de réponse HTTP, avec des en-têtes pour spécifier qu'il s'agit d'une image PNG et désactiver la mise en cache.

En résumé, ce code crée une image à partir du texte fourni dans la requête et retourne l'image en tant que réponse HTTP. Si tu as besoin de plus de détails.

Quand le modal success est bien chargé donc quand il apparaît ça hide la sidebar, la vide de ses li, uncheck toutes les checkboxes précédemment check comme ça l'utilisateur peut refaire un traitement de demande :

Ajout d'un modal success content qui permet après click sur le bouton publication du modal teams content d'afficher un loading, Quand le modal success est bien chargé donc quand il apparaît ça hide la sidebar, la vide de ses li, uncheck toutes les checkboxes précédemment check comme ça l'utilisateur peut refaire un traitement de demande et d'ensuite afficher un gif de vérification et un message de confirmation comme quoi la demande a bien été envoyé, contenant aussi un bouton ok et une croix qui tout les deux ferme le modal success, le modal teams content.

Le TeamsWebhookController :

```

class TeamsWebhookController
{
    private $webhookUrl = "https://prod-24.westeurope.logic.azure.com:443/workflows/109fc9711004421b9d65361707904";

    public function sendMessageWithStream(string $message, ?string $title = null, string $imagePath = null): bool
    {
        try {
            // Debug log du contenu du fichier
            Log::info('Contenu du fichier image', [
                'size' => filesize($imagePath),
                'path' => $imagePath
            ]);

            // Créer l'URL publique pour l'image
            $imageUrl = url('storage/' . basename($imagePath));

            // Copier le fichier vers storage/app/public
            $publicPath = storage_path('app/public/' . basename($imagePath));
            copy($imagePath, $publicPath);

            // Création de la charge utile
            $data = [
                'type' => 'message',
                'attachments' => [
                    [
                        'contentType' => 'application/vnd.microsoft.card.adaptive',
                        'content' => [
                            'type' => 'AdaptiveCard',
                            'version' => '1.4',
                            'body' => [
                                [
                                    'type' => 'TextBlock',
                                    'text' => $title,
                                    'weight' => 'Bolder',
                                    'size' => 'Medium'
                                ]
                            ]
                        ]
                    ]
                ]
            ];
        }
    }
}

```

```

        [
            'type' => 'TextBlock',
            'text' => 'Merci à vous d\'avance',
            'wrap' => true
        ],
        [
            'type' => 'Image',
            'url' => $imageUrl,
            'size' => 'Medium', // Vous pouvez a
            'altText' => 'Image du dépannage' //
        ]
    ]
];

// Debug log de la requête
Log::info('Requête Teams', [
    'data' => $data,
    'imageUrl' => $imageUrl
]);

$response = Http::post($this->webhookUrl, $data);

// Debug log de la réponse
Log::info('Réponse Teams', [
    'status' => $response->status(),
    'body' => $response->body()
]);

```



```

        // On vérifie plus précisément la réponse
        if (!$response->successful()) {
            Log::error('Erreur Teams', [
                'status' => $response->status(),
                'body' => $response->body()
            ]);
            return false;
        }

        return true;
    } catch (Exception $e) {
        Log::error('Exception Teams', [
            'message' => $e->getMessage(),
            'trace' => $e->getTraceAsString()
        ]);
        throw $e;
    }
}

```

On déclare la classe qui hérite de Controller

déclaration d'une variable webhook url qui contient l'url Teams Azure

La méthode sendMessageWithStream qui prend en paramètre un message, un titre optionnel, et un chemin vers une image. Elle retourne un booléen indiquant si l'envoi a réussi.

La taille et le chemin du fichier image sont enregistrés pour le débogage.

L'URL publique de l'image est créée, et le fichier image est copié vers un dossier public.

La charge utile du message, contenant le message, le titre et l'image sous forme de carte adaptive, est créée.

Les détails de la requête, y compris l'URL de l'image, sont enregistré pour le débogage.

La requête HTTP POST est envoyée au webhook Teams avec les données de la charge utile.

La réponse de Teams, y compris le statut et le corps de la réponse, est enregistrée.

Si la réponse n'indique pas un succès, une erreur est enregistrée et false est retourné.

En cas d'exception, l'erreur est enregistrée et l'exception est levée à nouveau.

En résumé, ce code gère la création et l'envoi d'un message avec une image à un webhook Teams, en journalisant chaque étape pour le débogage et en vérifiant le succès de l'opération.

On a aussi le ApiSendMessageController :

```
class ApiSendMessageController extends Controller
{
    public function sendMessage(Request $request): JsonResponse
    {
        try {
            // Validation des données
            $request->validate([
                'message' => 'required|string',
                'header' => 'nullable|string',
                'date' => 'required|date' // Validation de la date
            ]);

            // Log de début pour débogage
            Log::info('Début traitement message', [
                'message' => $request->message,
                'header' => $request->header,
                'date' => $request->date
            ]);

            // Appels aux autres contrôleurs (TeamsWebhook, TextToImage)
            $teamsWebhook = new TeamsWebhookController();
            $textToImage = new TextToImageController();

            // Création de l'image
            $imageStream = $textToImage->getImage($request->message);

            if (empty($imageStream)) {
                throw new Exception("Stream d'image vide");
            }

            // Création fichier temporaire pour l'image
            $tempFile = tempnam(sys_get_temp_dir(), 'teams_img_');
            file_put_contents($tempFile, $imageStream);
        }
    }
}
```

```

        $success = $teamsWebhook->sendMessageWithStream(
            $request->message,
            $request->header,
            $tempFile
        );

        unlink($tempFile); // Suppression du fichier temporaire

        // Vérification de l'envoi Teams
        if (!$success) {
            return response()->json([
                'success' => false,
                'message' => 'Échec de l\'envoi vers Teams'
            ], 500);
        }

        $demandeController = new DemandeController();
        $demandeController->store($request);

        return response()->json([
            'success' => true,
            'message' => 'Message envoyé et demande persistée avec succès'
        ]);

        // Réponse JSON de succès
    } catch (Exception $e) {
        // Nettoyage en cas d'erreur
        if (isset($tempFile) && file_exists($tempFile)) {
            unlink($tempFile);
        }
    }

```

```

        // Log de l'erreur
        Log::error('Erreur globale', [
            'message' => $e->getMessage(),
            'trace' => $e->getTraceAsString()
        ]);

        return response()->json([
            'success' => false,
            'message' => 'Erreur : ' . $e->getMessage()
        ], 500);
    }
}

```

Il hérite aussi de Controller.

La méthode sendMessage prend une requête et retourne une réponse JSON.

Les données de la requête sont validées pour s'assurer qu'elles contiennent un message, une date valide et, optionnellement, un en-tête.

Les détails du message sont enregistrés pour le débogage.

Les instances de teamsWebhookController et TextToImageController sont créées.

L'image du message est générée. Si l'image est vide, une exception est levée.

Un fichier temporaire est créé pour stocker l'image. Le contenu de l'image est écrit dans ce fichier.

Le message et l'image sont envoyés à Teams. Si l'envoi échoue, une réponse JSON d'erreur est retournée.

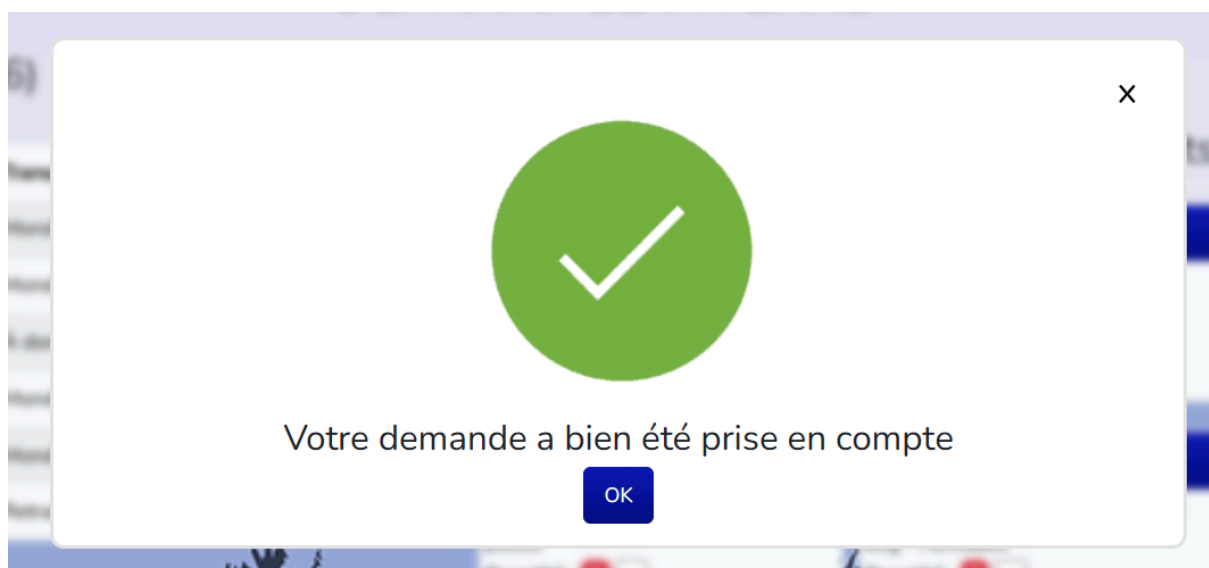
Le fichier temporaire est supprimé après l'envoi.

Le message est sauvegardé via le demandeController.

En cas d'erreur, le fichier temporaire est supprimé, l'erreur est journalisée, et une réponse JSON d'erreur est retournée.

Entre autres, ce code valide les données de la requête, journalise les informations, génère une image du message, envoie le message et l'image à Teams, puis persiste le message.

En cas d'erreur, il gère les exceptions et retourne une réponse appropriée.



```
let closeModalButton = document.getElementById("crossToClose");
closeModalButton.addEventListener("click", function() {
  teamsModal.classList.remove("active");
  successModalContent.classList.remove("active");
  teamsContentToPublish.classList.remove("active");
  bodyContent.classList.remove("blur");
  textArea.value = "";
});
```

Ce script ajoute un événement au bouton `closeModalButton` pour réinitialiser l'interface utilisateur lorsqu'on clique dessus. Il masque les modals, enlève l'effet de flou du contenu de la page, et réinitialise la zone de texte en la vidant.

Ce qui est la même chose pour celui là :

```
let cancelationButton = document.getElementById("btnCancelation");
cancelationButton.addEventListener("click", function() {
    teamsModal.classList.remove("active");
    successModalContent.classList.remove("active");
    teamsContentToPublish.classList.remove("active");
    bodyContent.classList.remove("blur");
    textArea.value = "";
});
```

```
let closeModalSuccessButton = document.getElementById("okToClose");
closeModalSuccessButton.addEventListener("click", function() {
    teamsModal.classList.remove("active");
    successModalContent.classList.remove("active");
    teamsContentToPublish.classList.remove("active");
    bodyContent.classList.remove("blur");
    textArea.value = "";

    let checkboxes = document.querySelectorAll('input[type="checkbox"]');
    checkboxes.forEach(function(checkbox) {
        checkbox.checked = false;
    });

    counter = 0;
    toggleSidebar();
});
```

Ce script ajoute un événement au bouton `closeModalSuccessButton` pour réinitialiser l'interface utilisateur lorsqu'on clique dessus. Il masque les 2 modals et rétablit l'apparence normale de la page, tout en réinitialisant la zone de texte et les cases à cocher. Enfin, il remet le compteur à zéro et active une fonction de la barre latérale.

```
function toggleSidebar() {
  let productListItems = productList.querySelectorAll('.productListElement');

  if (productListItems.length > 0) {
    if (!sidebar.classList.contains("visible")) {
      sidebar.classList.add("visible");
    }
    if (!changingContainer.classList.contains("sidebar-active")) {
      if(window.innerWidth > "1200"){
        changingContainer.classList.add("sidebar-active");
      }
    }
  } else {
    sidebar.classList.remove("visible");
    if(window.innerWidth > "1200"){
      changingContainer.classList.remove("sidebar-active");
    }
  }
}
```

La fonction toggleSidebar() affiche ou masque une barre latérale en fonction de la présence d'éléments dans une liste de produits et de la largeur de la fenêtre. Si des produits sont présents et que la fenêtre est assez large, elle active également un conteneur spécifique. Si aucun produit n'est présent, elle cache la barre latérale et désactive le conteneur.

Création d'un bouton accessibilité permettant d'afficher une modal.

Ce code inclut aussi un margin bottom pour permettre de pas surpasser le contenu en dessous de lui:

```
accessibilityButton.addEventListener("click", function() {
  accessibilityModal.classList.add("active");
  headerContainer.style.marginBottom = "100px";
});
```

Accessibilité

Proposant diverses utilités comme mettre le texte de toute la page en gras, d'agrandir ou de rétrécir la page et de changer aussi la police d'écriture de celle-ci.

Accessibilité

Choisir la police d'écriture :

Gras

Agrandir

Rétrécir

Fermer

Code pour cette modal :

Code pour les boutons de zooms, d'abord la déclaration des variables :

```
let zoomLevel = 1;
let zoomInButton = document.getElementById("zoomInButton");
let zoomOutButton = document.getElementById("zoomOutButton");
```

Ensuite le code de chaque button :

```
zoomInButton.addEventListener("click", function() {
    zoomLevel += 0.1; // Augmente le niveau de zoom
    document.body.style.zoom = zoomLevel;
});

zoomOutButton.addEventListener("click", function() {
    if (zoomLevel > 0.2) { // Empêche le niveau de zoom de devenir trop petit
        zoomLevel -= 0.1; // Diminue le niveau de zoom
        document.body.style.zoom = zoomLevel;
    }
});
```

Ce code ajoute des écouteurs d'événements aux boutons de zoom avant et de zoom arrière. Lorsque le bouton de zoom avant est cliqué, le niveau de zoom augmente de 0,1, tandis que lorsque le bouton de zoom arrière est cliqué, le niveau de zoom diminue de 0,1, sauf si le niveau de zoom est déjà inférieur à 0,2 pour éviter que le contenu ne devienne trop petit.

Le code du bouton pour mettre en gras tout le texte de la page :

```
document.getElementById("bold-button").addEventListener("click", function() {
    if (body.style.fontWeight && head.style.fontWeight === "bold") {
        body.style.fontWeight = "normal";
        head.style.fontWeight = "normal";
    } else {
        body.style.fontWeight = "bold";
        head.style.fontWeight = "bold";
    }
});
```

Ce code ajoute un écouteur d'événements au bouton de mise en gras. Lorsque ce bouton est cliqué, il vérifie si le texte du corps et de l'en-tête est déjà en gras. Si c'est le cas, il change le style de police en "normal"; sinon, il le change en gras.

Code pour le changement de police d'écriture :

```
let fontSelect = document.getElementById('fontSelect');
```

```
fontSelect.addEventListener("change", function() {
    console.log(fontSelect.value);
    if (fontSelect.value === "Arial"){
        body.style.fontFamily = "Arial";
    }
    if (fontSelect.value === "Times New Roman"){
        body.style.fontFamily = "Times New Roman";
    }
    if (fontSelect.value === "Courier New"){
        body.style.fontFamily = "Courier New";
    }
    if (fontSelect.value === "Georgia"){
        body.style.fontFamily = "Georgia";
    }
    if (fontSelect.value === "Verdana"){
        body.style.fontFamily = "Verdana";
    }
});
```

Ce code ajoute un écouteur d'événements à un élément de sélection de police (`fontSelect`). Lorsque l'utilisateur sélectionne une police différente, la police du corps de la page (`body`) est mise à jour pour refléter la police choisie, parmi les options suivantes : Arial, Times New Roman, Courier New, Georgia et Verdana.

PARLER DU RESPONSIVE

(en gros ce qu'on a mit dans le responsive, ce qui se passe donc à partir de quelle taille, sur quels éléments etc)

DE LA CSS

MONTRER LE CODE HTML DONC DES MODALS BUTTON ETC,
détailler un peu plus les étapes et les fonction etc.

Création d'une nouvelle page blade permettant d'accéder à l'historique des demandes contenant l'id, la date et le contenu de chaque demande dans un tableau similaire aux commandes, chaque contenu est composé d'un bouton qui quand on clique dessus affiche une modal pour afficher l'entièreté du contenu.

Donc création de 2 routes GET dans le fichier web.php, une pour la page historique-demandes :

```
Route::get('/historique-demandes', [DemandeController::class, 'index'])->name('demandes.show');
```

Une autre pour avoir get le contenu de chaque demande :


```
Route::get('get-contenu-{id}', [DemandeController::class, 'showContenu']);
```

Code du bouton accessible sur le page principal pour accéder à la page historique demandes :

```
document.getElementById('btnRedirectDemandes').addEventListener('click', function() {  
|   window.location.href = '/historique-demandes'; // Remplace '/ta-route' par l'URL de ta route  
});
```

Pareil pour le btn sur la page historique pour revenir sur celle des commandes :

```
document.getElementById('btnRedirectCommandes').addEventListener('click', function() {  
|   window.location.href = '/';  
});
```

Duplication du bouton accessibilité :

```
<div>  
  <button class="button-left" id= btnAccessibility>Accessibilité</button>  
  <button class="button-commandes" id="btnRedirectCommandes">Commandes</button>  
  <div class="modal-accessibility" id="modal-accessibility">  
    <div id="divPolice">  
      <label for="fontSelect">Choisir la police d'écriture :</label>  
      <select id="fontSelect">  
        <option value="Arial" style="font-family: Arial;">Arial</option>  
        <option value="Times New Roman" style="font-family: Times New Roman;">Times New Roman</option>  
        <option value="Courier New" style="font-family: Courier New;" selected>Courier New</option>  
        <option value="Georgia" style="font-family: Georgia;">Georgia</option>  
        <option value="Verdana" style="font-family: Verdana;">Verdana</option>  
      </select>  
    </div>  
    <div>  
      <button class="bold-button" id="bold-button">Gras</button>  
      <button class="zoom-buttons" id="zoomInButton">Agrandir</button>  
      <button class="zoom-buttons" id="zoomOutButton">Rétrécir</button>  
      <button class="ok-button" id="ok-button">Fermer</button>  
    </div>  
  </div>  
</div>
```

Contenu de la page :

```
<div class="container-fluid">
  <div class="demandes-en-attentes flex-column">
    <h2 class="h-demande">Historique Demandes ({{ count($demandes) }})</div><div class="list-solid"></div></h2>

    <div class="col-5 tableau-demandes">
      <table class="table table-striped mt-4">
        <thead>
          <tr>
            <th scope="col">ID Demande</th>
            <th scope="col">Date Demande</th>
            <th scope="col">Contenu</th>
          </tr>
        </thead>
        <tbody>
          @foreach($demandes as $demande)
            <tr>
              <td>{{ $demande->id }}</td>
              <td>{{ (new \DateTime($demande->dateDemande))->format('d-m-Y') }}</td>
              <td><button data-id="{{ $demande->id }}" class="btn-voir-plus" id="btn-voir-plus">Voir Plus</button></td>
            </tr>
          @endforeach
        </tbody>
      </table>
    </div>
  </div>

</div>

<div id="modal-contenu-voir-plus" class="hidden">
  <div class="modal-voir-plus" id="modal-voir-plus">
    <textarea id="textAreaVoirPlus" disabled></textarea>
    <input class="btn btn-danger btnCancelation" type="reset" value="Annuler" id="btnFermerVoirPlus">
  </div>
</div>
```

Dans un premier temps on a un h2 suivi d'un count qui compte le nombre de ligne du tableaux, donc le nombre de demandes un tableau avec l'id, la date et du bouton pour le contenu des demandes ainsi que la modal s'affichant après avoir cliqué sur le bouton.

Pour ce même bouton on a du faire 2 boucles et un fetch :

```
for(let b = 0; b < voirPlusButton.length; b++) {
  voirPlusButton[b].addEventListener('click', function() {

    for (let d = 0; d < voirPlusButton.length; d++) {
      voirPlusButton[d].disabled = true
    }
    //console.log(voirPlusModal);
    if(voirPlusModal.classList.contains("hidden")){
      let idContenu = voirPlusButton[b].dataset.id;
      console.log(idContenu);
      let url = "/get-contenu-" + idContenu;
      fetch(url,{
        method: 'GET',
      })
      .then(response => {
        if (!response.ok) {
          throw new Error('Erreur de réseau');
        } return response.json();
      })
      .then(jsonResponse => {
        console.log(jsonResponse);
        textAreaVoirPlus.value = jsonResponse.contenu;
        textAreaVoirPlus.style.height = calculateHeightToApply(textAreaVoirPlus.value);
      });
      voirPlusModal.classList.remove("hidden");
      voirPlusModal.classList.add("visible");
      bodyContent.classList.add("blur");

    } else {
      voirPlusModal.classList.remove("visible");
      voirPlusModal.classList.add("hidden");
    }
  });
}
```

La première boucle for sert pour chaque bouton pour que l'évènement suivant ce passe à chaque fois que n'importe quel bouton est cliqué.

Dans la 2ème partie du projet on a pour but d'insérer les valeurs dans un tableau excel.

Pour ce faire nous utiliserons la librairie "phpspreadsheet"

il fallait retirer dans le php.ini le “,” devant le : extension=zip car sinon ça fait un message d'erreur

Ensuite il faut écrire : composer require phpooffice/phpspreadsheet
et l'installation se fait normalement

Ensuite on créer un controller excel : php artisan make:controller ExcelController

Compétences acquises

Au cours de ce stage, j'ai pu acquérir ou améliorer de nombreuses compétences, notamment :

- **Maîtrise des technologies web (HTML, CSS, JAVASCRIPT, PHP, MySQL, AJAX).**
- **Gestion de projets informatiques..**
- **Capacité à travailler en équipe et à respecter des délais de projet.**
- **Utiliser un nouveau framework php ainsi que d'autres logiciels.**

Conclusion

Ce stage a été une expérience très enrichissante qui m'a permis d'appliquer concrètement mes connaissances théoriques en développement web et en gestion de projet, également de découvrir le travail en équipe sur un projet. J'ai pu renforcer mes compétences techniques tout en découvrant les exigences du milieu professionnel. Ce stage m'a conforté dans mon choix de carrière et m'a donné l'envie de poursuivre dans le domaine du développement web.

Je tiens à remercier l'entreprise POUBEAU et mon tuteur Anton COVU pour leur accueil et leur soutien tout au long de cette période de stage.

Annexes

- **Screenshot du projet.**
- **Code source..**
- **Base de données MySQL (structure et tables).**