

Entreprise cobra

Responsable de formation : Betsalel Cohen

Tuteur de stage : Jacque Buffeteau

Synthèse de Stage

FELLOUS DAVID

BTS SIO SLAM 1 ÈRE ANNEE

Introduction

Dans le cadre du BTS informatique au seins de l'ORT, nous dévions effectuer un stage chez un professionnel dans le domaine de nos études. Pour ma part j'ai effectué le miens auprès de Betsalel Cohen auto-entrepreneur, développeur spécialisé dans la création et maintenance de site internet.

Ma toute première occupation a été de me familiariser sur le projet en cours, j'ai passé avec mon tuteur la veille du stage plusieurs heures à l'exposition du projet.

Celui-ci, que je dois rejoindre en cours de route a été créé en un an et continue d'être modifié et amélioré continuellement. Quand j'ai aperçu le nombre de page que possédait le projet j'ai tout d'abord eu peur de me perdre, mais au fur et à mesure que mon tuteur exposé le projet je me suis rendu compte que beaucoup de chose ressemblait à ce qu'on avait vu en classe.

L'affichage d'une base de données ou bien certaines requêtes, même s'il s'agissait des versions beaucoup plus évoluées de tous ce que j'ai vue jusqu'à présent en PHP, j'étais rassuré de savoir que je ne serais pas entièrement perdue, meme si j'aller devoir apprendre beaucoup plus d'avantage en me documentant, sur des méthodes PHP, ajax, des bibliothèque JS que je n'avais jamais vue et l'utilisation de Bootstrap.

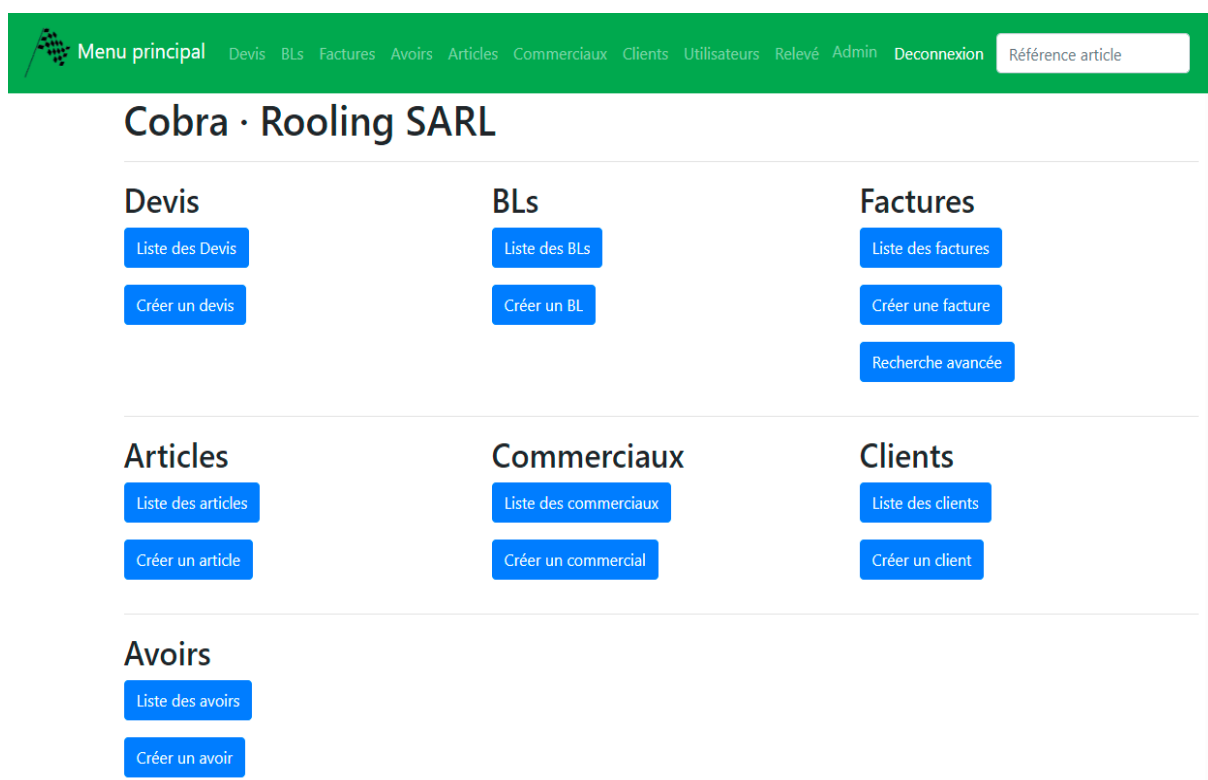
Le projet sur lequel j'ai travaillé est un site de facturation qui permet de gérer des commandes pour une entreprise : Cobra. Cette société vend des pièces d'automobile comme des par-brises, des portes, des vitres, des pots d'échappements etc.

Ce site est destiné à l'utilisation par les employés, il permet d'avoir accès a la liste des produits proposé par l'entreprise, avoir accès à la liste des clients déjà enregistrés pour créer des factures, des BLS, des

avoirs, des devis en fonction du souhait du client et avoir accès très rapidement à l'ensemble de ces données.

J'ai dû notamment apprendre un minimum le système de facturation pour comprendre le fonctionnement du site et ce que chaque élément propose

Voici à quoi ressemble la page d'accueil du site([index.php](#)) :



The screenshot shows the home page of the 'Cobra · Roolling SARL' website. The header is green and contains a 'Menu principal' with links to 'Devis', 'BLs', 'Factures', 'Avoirs', 'Articles', 'Commerciaux', 'Clients', 'Utilisateurs', 'Relevé', 'Admin', and 'Deconnexion'. There is also a search bar labeled 'Référence article'. The main content area is white and features several sections, each with a title and two buttons:

- Devis**: 'Liste des Devis' and 'Créer un devis'
- BLs**: 'Liste des BLs' and 'Créer un BL'
- Factures**: 'Liste des factures', 'Créer une facture', and 'Recherche avancée'
- Articles**: 'Liste des articles' and 'Créer un article'
- Commerciaux**: 'Liste des commerciaux' and 'Créer un commercial'
- Clients**: 'Liste des clients' and 'Créer un client'
- Avoirs**: 'Liste des avoirs' and 'Créer un avoir'

Première tâche

Ma première mission était simple mais mon tuteur me la donnée pour que je cherche à comprendre comment fonctionne certaine page par moi-même, dans la page quote.php (ci-dessous) qui permet l'affichage des factures. Je devais faire en sorte que l'affichage des dates soit trié par les dates et non plus par le n° d'immatriculation, le système de trie était déjà en place

Factures

Ajouter une facture

Search:

Facture N°	Référence de facture	N° d'immatriculation	Code client	Nom client	Date	Accompte	Actions
54			rezarzazer	dsfqsfsq	01/06/2021 17:03:28	0	<button>Afficher</button> <button>Modifier</button>
53			5555	LeTune	27/05/2021 00:26:17	0	<button>Afficher</button> <button>Modifier</button>
52		fg123gh	cobra	Cobra Store FR	24/05/2021 21:31:13	0	<button>Afficher</button> <button>Modifier</button>
51	test de ref 2	AZ123IB	cobra	Cobra Store FR	13/12/2020 21:39:13	0	<button>Afficher</button> <button>Modifier</button>
36	4f6dq432fds	AZ123ER	cobra	Cobra Store FR	11/12/2020 00:00:00	Payé	<button>Afficher</button> <button>Modifier</button> Payé
49			cobra	Cobra Store FR	01/10/2020 15:36:35	0	<button>Afficher</button> <button>Modifier</button>

Pour accomplir ma tâche il m'a suffi de modifier le code de la manière suivante :

```
$(document).ready(function() {  
    $('#quotes_table').DataTable({  
        "order": [  
            [2, "desc"]  
            [5, "desc"]  
        ]  
    });  
});
```

J'ai dû chercher dans toute la page pour comprendre qu'il fallait passer par là pour le tri, ce qui était le but recherché par le tuteur.

J'ai ensuite reçu pour mission de mettre en place des outils qui permettront des modifié les dates d'échéance des factures selon les moyens de paiement du client (dit "billing" mode) :

- Comptant (billing mode 9) : pas de date d'échéance !
- 30 jours ou par défaut (billing mode 7, ou alors aucun des billing modes 7, 8 et 9 cochés) : ne pas toucher, laisser à 30j
- 30 jours fin du mois (billing mode 8) : mettre la date d'échéance a la fin du mois suivant (ex : 3 mars => échéance le 30 avril ; 15 janvier => échéance le 28 février). Des recherches internet ont été nécessaire :


```

74 + $lastDay = false;
75 + if ($client['clientcode'] != '') {
76 +     $sql_lastDay = "SELECT * From clients_billing_modes WHERE clients_billing_mode_id_client = " . $client['Id'] . " AND clients_billing_mode_id_billing_mode = '8' ";
77 +     $result4 = $bdd->query($sql_lastDay);
78 +     if ($result4->fetch()) $lastDay = true;
79 + }
110 +
111 + if(!$comptant)
112 + {
113 +     if($lastDay)
114 +     {
115 +         $single_invoice ? $invoice->setDue(date('d/m/Y ',(strtotime($invoice_to_print[0]['invoice_date'] . 'last day of next month')))); """;
116 +     }
117 +     else if(!$lastDay)
118 +     {
119 +         $single_invoice ? $invoice->setDue(date('d/m/Y', strtotime($invoice_to_print[0]['invoice_date'] . '+1 month'))): """;
120 +     }
121 + }
122 +
123 + }

```

Les recherches concernent surtout les systèmes de date mis en place.

J'ai dû ensuite mettre en place une recherche dynamique qui permet de rechercher en temps réel dans tous les éléments de la base de données. Ceci afin d'afficher dans le tableau les éléments correspondant à la recherche demandée peu importe le champ utiliser. Et cela en temps réel, dès que l'utilisateur appuis sur une touche d'un des *input*, cela actualise le tableau des résultats en dessous en temps réel, en utilisant la méthode ajax

 Menu principal

DevīsBLsFacturesAvoirsArticlesCommerciauxClientsUtilisateursRelevéAdminDeconnexion

Référence article

Ajouter une facture

Rechercher

Réinitialiser les champs

Pour mettre en place la méthode ajax j'ai dû créer une nouvelle page pour mes fonction JS et une nouvelle page PHP `GetInvoices.php` qui exécutera la requête qu'on récupérera plus tard.

Tous les *input* se trouvent dans un *form* dans la page `advanced search.php`, je leur ai ajouté une classe supplémentaire pour pouvoir lever l'événement *keyUp* dès qu'on entre un caractère dans n'importe quel *input* et appeler ma fonction. Le résultat qu'enverra ensuite la page ajax sera utilisé pour remplir un tableau en dessous de la recherche affichant les factures possédant un de ses éléments.

Pour ce faire, mon responsable m'a transmis divers liens de documentation expliquant plus amplement l'utilisation de l'ajax et également certains raccourcis comme l'utilisation du `$.post` directement au lieu de `$.ajax` et spécifié ensuite la méthode.

J'ai dû apprendre l'utilisation du *serialize*, qui me permettra de récupérer les données de tous les champs dans ma fonction ajax, et

de les envoyer directement dans la page PHP pour que la requête débute

La fonction appelée lors du *keyUp* se nomme *UpdateTable* :

```
// fonction de david

function UpdateTable()

//recupere les 8 champs et 1 envoi eu php et au ajax
{
    $.post(
        'AJAX/GetInvoices.php', // Le fichier cible côté serveur.
        $('#search_form').serialize(),
        nom_fonction_retour, // Nous renseignons uniquement le nom de la fonction de retour.
        'data' // Format des données reçues.
    );

    function nom_fonction_retour(texte_recu) {
        // Du code pour gérer le retour de l'appel AJAX.
        console.log(texte_recu)
    }
}
```

J'ai ensuite créé la page *getInvoice* dans laquelle toute les vérifications (des *assets* qui retournent de paramètre *true* ou *false* utilisé ensuite) seront faites avant de lancer la requête SQL.

Au chargement de la page quand il n'y a encore rien d'entré dans les *input* nous voulons afficher les factures les plus récentes par défaut :

```
$typeOfDoc = $code = $numberplate = $ref = $min = $max = $numberOfDoc = $billingStatus = "";
if ( $_POST['code'] == "" && $_POST['numberplate'] == "" &&
    $_POST['ref'] == "" && $_POST['min'] == "" && $_POST['max'] == "" &&
    $_POST['numberOfDoc'] == "" ) { $whereClause = ""; }
else {
    $whereClauseInitiated = false;
    $whereClause = "WHERE ";

    //Todo
    if (isset($_POST['typeOfDoc'])) {
```

Tous les *input* passe par ce genre de vérification :

```
if (isset($_POST['numberplate']) && $_POST['numberplate'] != "") {
    if ($whereClauseInitiated) $whereClause .= " AND ";
    $whereClause .= "invoice_numberplate LIKE '%" . $_POST['numberplate'] . "%' ";
    if (!$whereClauseInitiated) $whereClauseInitiated = true;
}
```

Une fois les vérifications faites nous passons à la requête SQL

```
$sql_search = "SELECT i.*, c.*,
(
    SELECT ROUND(SUM(`invoices_article_updated_quantity`*(`invoices_article_updated_unit_price`*(100-invoices_article_updated_rate)/100
FROM invoices_articles ia LEFT JOIN bls b ON ia.invoices_article_id_bl = b.bl_id
WHERE ia.invoices_article_id_invoice = i.invoice_id AND (isnull(b.bl_id) or ( b.bl_canceled = 0 AND b.bl_negative = 0 ))
) AS 'invoice_TotalHT'
FROM (SELECT * FROM invoices " . $whereClause . " GROUP BY invoice_id ORDER BY invoice_date DESC LIMIT " . $debut . " , " . $items_in_pag
JOIN clients c ON c.Id = i.invoice_id_client
GROUP BY i.invoice_id ORDER BY i.invoice_date DESC ";
$result = $bdd->query($sql_search);
$result->fetchAll();
```

La requête étant assez compliqué, elle a été réalisée par mon responsable de formation

Les données seront ensuite parcourues dans un *foreach* pour l'afficher dans un tableau

```
<?php
foreach ($bdd->query($sql_search) as $invoice) {
    $color = "";
    if ($invoice['invoice_paid']) $color = 'class = "table-success"';
    else if ($invoice['invoice_id_credit'] != 0) $color = 'class = "table-warning"';
    ?>

    <tr <?= $color ?>>
        <th scope="row"><?= $invoice['invoice_number'] ?></th>
        <td> <?= $invoice['invoice_ref'] ?> </td>
        <td> <?= $invoice['invoice_numberplate'] ?> </td>
        <td> <?= $invoice['clientcode'] ?> </td>
        <td> <?= $invoice['clientname'] ?> </td>
        <td> <?= dateToCompactDateFR($invoice['invoice_date']) ?> </td>
        <td> <?= htToItc($invoice['invoice_TotalHT']) ?> </td>
        <td> <?= $invoice['invoice_paid'] ? "Payé" : $invoice['invoice_advance'] ?> </td>
        <td>
            <?php if ($invoice['invoice_number'] == 0) { ?> <a href="invoice_to_delete.php?id=<?= $invoice['invoice_id'] ?>" class="btn btn-danger
            <a class="btn btn-success btn-xs" onclick="window.open('invoice_details.php?id=<?= $invoice['invoice_id'] ?>','name','width=1200
            <a class="btn btn-success btn-xs" onclick="window.open('edit_invoice.php?id=<?= $invoice['invoice_id'] ?>','name','width=1200,t
            </td>
        </tr>
    <?php } ?>
```


Ensuite, nous envoyons toutes les données de la page pour les insérer dans la fonction `UpdateTable()`, qui videra préalablement une `<div>` créé `TableInvoices` dans la page *advanced research*, là où sont situés les `input` de recherche, avec l'outil `empty`, puis on affichera le résultat dans cette `<div>`

```
$('.input-to-listen').keyup(UpdateTable);

// $('.form-control').keyup(function() {
//     alert($('#search_form').serialize());
// });

//recupere les 8 champs et 1 envoi eu php et au ajax
function UpdateTable() {
    $.post(
        'AJAX/GetInvoices.php', // Le fichier cible côté serveur.y
        $('#search_form').serialize(),
        nom_fonction_retour, // Nous renseignons uniquement le nom de la fonction de retour.
        'text' // Format des données reçues.
    );

    function nom_fonction_retour(texte_recu) {
        // Du code pour gérer le retour de l'appel AJAX.
        // console.log(texte_recu);
        $('#TableInvoices').empty();
        $('#TableInvoices').append(texte_recu);
    }
}
```

Maintenant grâce à la méthode ajax, le tableau est devenu dynamique en affichant en temps réel les factures qui correspondent à la recherche d'un des inputs. Ainsi, dès qu'on écrit une lettre le tableau se met à jour comportant seulement les éléments recherchés :

Type

Client

Plaque

Référence

Facture

a

Date de début

Date de fin

Numéro

Etat du paiement

jj/mm/aaaa

jj/mm/aaaa

Tous

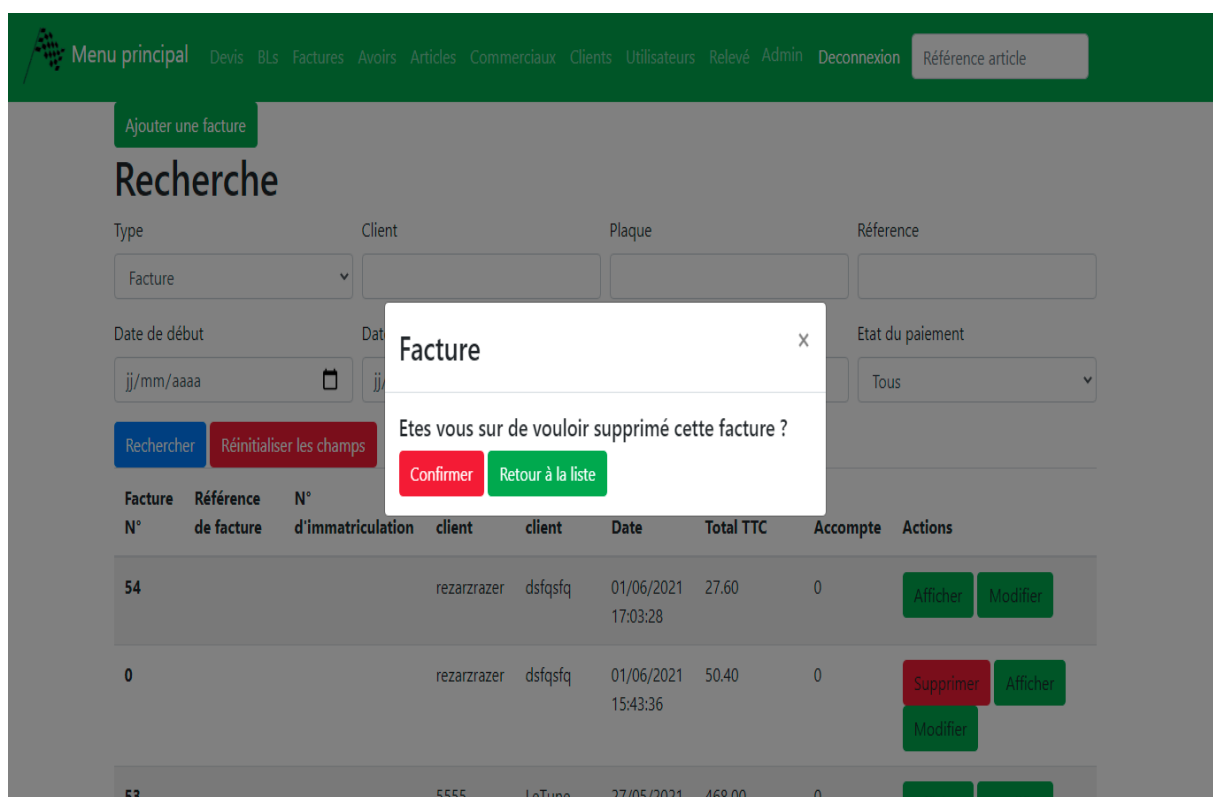
Rechercher

Réinitialiser les champs

Facture N°	Référence de facture	N° d'immatriculation	Code client	Nom client	Date	Total TTC	Accompte	Actions
51	test de ref 2	AZ123IB	cobra	Cobra Store FR	13/12/2020 21:39:13	45.00	0	Afficher Modifier
36	4f6dq432fds	AZ123ER	cobra	Cobra Store FR	11/12/2020 00:00:00	600000126.00	Payé	Afficher Modifier
48		ar456tr	cobra	Cobra Store FR	30/09/2020 22:43:15	118.00	20	Afficher Modifier
34	test de ref 2	ab123cd	amz1234	Amazon FR	06/07/2020 14:00:46	0.00	0	Afficher Modifier

Deuxième tâche

La mission suivante m'a permis d'aborder une autre partie de la bibliothèque : le *modal*. Pour l'ajout d'un système de suppression possible, en plus des boutons afficher (redirige vers la page [invoicesDetail.html](#)) et modifier (qui redirige vers la page [editInvoices.html](#)) de facture uniquement si elle n'a pas de numéro, avec un bouton supprimer et un *modal* qui demande confirmation



Une fois confirmer nous refaisons appel à la fonction `updateTable`, comme à une fois confirmer la facture supprimée ne sera plus présente

```
<?php if ($invoice['invoice_number'] == 0) { ?> <a class="btn btn-danger btn-xs" data-toggle="modal" data-target="#deleteInvoices">Supprime
```

Nous avons ensuite ajouté une croix qui permet de réinitialiser l'input de client et d'actualiser le tableau pour qu'il revienne à l'affichage d'origine sans contrainte

```
$(".close").click(function () {  
    $("#code").val("");  
    $("#nomCli").text("");  
    $("#client").val("");  
    $(".close").hide();  
    UpdateTable();  
});
```

Voici la forme finale du modal :

```
<!-- Modal -->  
<div class="modal fade" id="deleteInvoices" tabindex="-1" role="dialog" aria-labelledby="deleteInvoices" aria-hidden="true">  
  <div class="modal-dialog modal-dialog-centered" role="document">  
    <div class="modal-content">  
      <div class="modal-header">  
        <h3 class="modal-title" id="printModalTitleMail">Facture</h3>  
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">  
          <span aria-hidden="true">&times;</span>  
        </button>  
      </div>  
      <div class="modal-body">  
        <h5>Etes vous sur de vouloir supprimer cette facture ?</h5>  
        <div class="col-sm-offset-11">  
          <input type="hidden" id="invoice_to_delete" name="invoice_to_delete" value="" />  
          <a class="btn btn-danger btn-xs" id="btnDeleteInvoices" data-dismiss="modal">Confirmer</a>  
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">Retour à la liste</button>  
        </div>  
      </div>  
    </div>  
  </div>  
</div>
```

L'ajax qui est ensuite appelé envoie le numéro de la facture qui a été sélectionné à la page deleteInvoices.html. Il lancera une requête SQL préparée par mon responsable pour supprimer de la BDD la facture correspondante :

```
function deleteInvoice(invoice) {
    $.get(
        "AJAX/DeleteInvoices.php", // Le fichier cible côté serveur.
        {
            invoice_to_delete: invoice,
        },
        invoiceDeleted, // Nous renseignons uniquement le nom de la fonction de retour.
        "text" // Format des données reçues.
    );

    function invoiceDeleted() {
        UpdateTable();
    }
}
```

L'ID de l'Invoice_to_delete est au préalable récupéré :

```
$(document).ajaxComplete(function () {
    $(".delete-btn").click(function () {
        $("#deleteInvoices").modal("show");
        $("#invoice_to_delete").val($(this).attr("id"));
        $("#btnDeleteInvoices").click(function () {
            deleteInvoice($("#invoice_to_delete").val());
        });
    });
});
```

Troisième Tâche

Mise en place de pagination PHP sur la page `advanced_search`, les résultats étaient limités à 100 dans le tableau jusqu'à présent, nous allons utiliser la méthode de pagination pour pouvoir parcourir l'ensemble des résultats

Pour connaître le nombre de page où seront repartis les résultats nous divisons le nombre total de facture (nombre trouvé par requête SQL) par le nombre qu'on veut afficher par page (ici il a été défini à 100)

```
$sql_page_nb = "SELECT COUNT(*) as count
FROM invoices " . $whereClause . "";
$sql_page_nb= ($bdd->query($sql_page_nb));
$sql_page_nb= $sql_page_nb->fetchObject();
var_dump($sql_page_nb);

$pageShow =ceil($sql_page_nb->count / $items_in_page ) ;
```

Nous définissons combien d'élément nous voulons par page et également un moyen de savoir sur quelle page nous sommes

```
$items_in_page = 10;

if (!isset($_POST['page']) || $_POST['page'] == "") $_POST['page'] = 1;
$page = $_POST['page'];
$debut = $page == 1 ? 0 : ($page - 1) * $items_in_page;
```

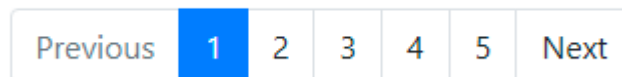
```
$(".nav-button").click(function () {
    $("#page").val($(this).attr("id"));
    UpdateTable();
});
```

```
<input type="hidden" name="page" id="page" value="">
```

Puis nous mettons en place la navigation pour parcourir les pages

```
<nav aria-label="Page navigation">
  <ul class="pagination justify-content-center">
    <li class="page-item <?=$page == 1 ? "disabled" : "" ?>">
      <a class="page-link nav-button" tabindex="-1" id="<?=$page-1?>">Previous</a>
    </li>
    <?php for ($i = 1; $i <= $pageShow; $i++) { ?>
      <li class="page-item <?=$i == $page ? "active" : "" ?>"><a class="page-link nav-button" id="<?=$i?>"><?=$i ?></a>
    </li>
    <?php } ?>
    <li class="page-item <?=$page == $pageShow ? "disabled" : "" ?>">
      <a class="page-link nav-button" id="<?=$page+1?>">Next</a>
    </li>
  </ul>
</nav>
```

Ce qui nous permet d'avoir, en plus des critères de sélection, le moyen de parcourir différente page pour accéder à tous les articles



J'ai ensuite modifié le système de navigation pour qu'il soit plus optimal quand de nombreuses pages doivent être affichées, et en changeant le nom de quelques variables pour qu'elles soient plus cohérentes

Voici la solution pour laquelle j'avais d'abord opté

```
173 +<!-- si nombre de page inferieur ou egal a 6 -->
174 +<nav aria-label="Page navigation">
175 +   <ul class="pagination justify-content-center">
176 +
177 +       <?php if ($pageShow <= 6) { ?>
178 +           <li class="page-item <?= $page == 1 ? "disabled" : "" ?>">
179 +               <a class="page-link nav-button" tabindex="-1" id="<?= $page - 1 ?>">Previous</a>
180 +           </li>
181 +           <?php for ($i = 1; $i <= $pageShow; $i++) { ?>
182 +               <li class="page-item <?= $i == $page ? "active" : "" ?>"><a class="page-link nav-button" id="<?= $i ?>"><?= $i ?></a>
183 +           </li>
184 +           <?php } ?>
185 +           <li class="page-item <?= $page == $pageShow ? "disabled" : "" ?>">
186 +               <a class="page-link nav-button" id="<?= $page + 1 ?>">Next</a>
187 +           </li>
188 +       <?php } else if ($pageShow > 6 && $page < 6) { ?>
189 +           <ul class="pagination justify-content-center">
190 +               <li class="page-item <?= $page == 1 ? "disabled" : "" ?>">
191 +                   <a class="page-link nav-button" tabindex="-1" id="<?= $page - 1 ?>">Previous</a>
192 +               </li>
193 +               <?php for ($i = 1; $i <= $page + 3; $i++) { ?>
194 +                   <li class="page-item <?= $i == $page ? "active" : "" ?>"><a class="page-link nav-button" id="<?= $i ?>"><?= $i ?></a>
195 +               </li>
196 +               <?php } ?>
197 +               <li class="page-item <?= $page == $pageShow ? "disabled" : "" ?>">
198 +                   <a class="page-link nav-button" id="<?= $page + 1 ?>">Next</a>
199 +               </li>
200 +           <?php } else if ($pageShow > 6 && $page > 6) { ?>
201 +               <nav aria-label="Page navigation">
202 +                   <ul class="pagination justify-content-center">
203 +                       <li class="page-item <?= $page == 1 ? "disabled" : "" ?>">
204 +                           <a class="page-link nav-button" tabindex="-1" id="<?= $page - 1 ?>">Previous</a>
205 +                       </li>
206 +                       <?php for ($i = $page - 3; $i <= $page + 3; $i++) { ?>
207 +                           <li class="page-item <?= $i == $page ? "active" : "" ?>"><a class="page-link nav-button" id="<?= $i ?>"><?= $i ?></a>
208 +                       </li>
209 +                       <?php } ?>
210 +                       <li class="page-item <?= $page == $pageShow ? "disabled" : "" ?>">
211 +                           <a class="page-link nav-button" id="<?= $page + 1 ?>">Next</a>
212 +                       </li>
213 +                   <?php } ?>
214 +               </ul>
215 +           </nav>
216 +       </li>
217 +
218 +   </ul>
219 + </nav>
```

en vérifiant la page dans laquelle on se trouve et du nombre total de page mais on répète plusieurs fois le code alors que cela peut être plus simple

Puis nous avons trouvé un moyen de beaucoup mieux optimisé cela, les *if* sont effectués à part du code

```
// if current page is greater than total pages...
if ($page > $total_pages) {
    // set current page to last page
    $page = $total_pages;
} // end if
// if current page is less than first page...
if ($page < 1) {
    // set current page to first page
    $page = 1;
} // end if

//getting offset
$debut = $page == 1 ? 0 : ($page - 1) * $items_in_page;
```

```
161 <nav aria-label="Page navigation">
162 <ul class="pagination justify-content-center">
163 <!-- Boutons début -->
164
165 <li class="page-item <?= $page == 1 ? "disabled" : "" ?>">
166 <a class="page-link nav-button" <?= $page == 1 ? 'tabindex="-1"' : "" ?> id="1">Début</a>
167 </li>
168 <li class="page-item <?= $page == 1 ? "disabled" : "" ?>">
169 <a class="page-link nav-button" <?= $page == 1 ? 'tabindex="-1"' : "" ?> id="<?= $page - 1 ?>">Précédent</a>
170 </li>
171 <?php for ($x = ($page - $range); $x < (($page + $range) + 1); $x++) {
172     if ($x > 0 && $x <= $total_pages) {
173         if ($x == $page) { // Si page active ?>
174             <li class="page-item active">
175                 <a class="page-link" disabled tabindex="-1" ><?= $x ?></a>
176             </li>
177         } else { //sinon ?>
178             <li class="page-item">
179                 <a class="page-link nav-button" id="<?= $x ?>"><?= $x ?></a>
180             </li>
181         }
182     }
183 } ?>
184 <!-- Boutons fin -->
185 <li class="page-item <?= $page == $total_pages ? "disabled" : "" ?>">
186 <a class="page-link nav-button" <?= $page == $total_pages ? 'tabindex="-1"' : "" ?> id="<?= $page + 1 ?>">Suivant</a>
187 </li>
188 <li class="page-item <?= $page == $total_pages ? "disabled" : "" ?>">
189 <a class="page-link nav-button" <?= $page == $total_pages ? 'tabindex="-1"' : "" ?> id="<?= $total_pages ?>">Fin</a>
190 </li>
191 </ul>
192 </nav>
```

41

cds123

CDDiscount

20/08/2020

108.00

0

01:01:43

Afficher

Modifier

Début Précédent 5 6 7 8 9 10 11 Suivant Fin

Nous affichons désormais au début quelque page qui suivent la première et quand on déplace ce stade nous affichons les trois pages précédentes et les 3 pages suivantes de celle où on se situe et également un bouton début et un bouton fin qui nous permettent de se déplacer plus rapidement si on se trouve en milieu de recherche

Pour connaître le nombre d'élément que l'utilisateur veut afficher par page nous ajoutons un input select qui permettra d'actualiser dynamiquement les résultats du tableau

```
<div class="col-md-3 mb-3">
  <label for="typeOfDoc">Résultats par page</label>
  <select class="form-control input-to-listen" id="itemsPerPage" name="itemsPerPage">
    <option selected value="10">10</option>
    <option value="20">20</option>
    <option value="50">50</option>
    <option value="100">100</option>
  </select>
</div>
```

```
$('#nbElements').change(function () {
  UpdateTable();
});
```

Dès qu'une option sera choisie nous referons appeler à update table qui mettra à jour le nombre de résultat par pages

J'ai ensuite adapté ce format de recherche aux pages les plus importantes du projet, BLS, *credits*, *invoices*, *quotes*, en renommant les anciennes page 'old', qui permettaient seulement d'afficher les éléments sans pouvoir faire de recherche, et le système de pagination était basé sur une bibliothèque JS, elle était très simple, mais elle charger tous les éléments de la BDD à l'ouverture de la page, ce qui rendait l'affichage au lancement très lent (comportant de milliers d'éléments)

```

<script>
    $(document).ready(function() {
        $('#bls_table').DataTable({
            "order": [
                [5, "desc"]
            ]
        });
    });
</script>

```

Donc j'ai adapté ces pages en ajax pour permettre une recherche complète et dynamique et optimisé l'affichage : grâce aux requêtes progressives on charge de la BDD que ce que l'on veut afficher, et non toute la BDD dès le début. Chaque requête va conserver le même mode opératoire. On passe la page actuelle en 'old' pour garder une trace de l'ancienne version. Ensuite crée une nouvelle comportant la méthode de recherche présenté précédemment avec une adaptation pour chacune d'elle. Car on ne recherche pas toujours la meme chose et donc on adapte également les requêtes. Le passage en ajax se fait grâce à plusieurs pages : la page de base, par exemple [bls.php](#), [GetBls.php](#) pour récupérer le tableau, [DeletBls.php](#) qui permet de supprimer un BLS en utilisant une requête, et enfin une page JS qui lui est accordé [bls.js](#) pour l'utilisation de l'ajax. Les pages concernés par cette méthode sont BLS, credits, invoices et quotes.

▼ AJAX

- DeletBls.php
- DeletCredits.php
- DeletInvoices.php
- DeletQuotes.php
- get_article_from_bl.php
- get_articles.php
- get_clients.php
- GetBls.php
- GetCredits.php
- GetInvoices.php
- GetQuotes.php

▼ Functions

- JS add_article.js
- JS bls.js
- JS credits.js
- JS dateFr.js
- functions.php
- JS invoices.js
- JS quotes.js

Conclusion (et remise du trophée des trois sorciers)

Durant ce stage, j'ai appris à travailler en autonomie, à réfléchir par moi-même pour trouver des solutions ou bien de nouvelles idées pour résoudre un problème, mais également à beaucoup me documenter, car très souvent un problème que l'on rencontre peut déjà avoir été résolu par le passé.

J'ai appris à écouter les reproches que me faisait mon tuteur pour ne plus recommencer les mêmes, qui m'auraient gêné dans le futur. Mon responsable de formations a été très attentionné avec moi, j'ai pu souvent lui poser des questions sur des problèmes rencontrés ou je ne pouvais plus avancer, s'agissant d'un projet en cours de route, beaucoup de méthodes m'ont été inconnues, j'en ai appris davantage sur le PHP, l'ajax, les bibliothèques JS, sur Bootstrap et sur les requêtes SQL.

Je tiens à remercier singulièrement Mr Betsalel Cohen qui a passé énormément de temps à m'expliquer et à me conseiller toujours en cherchant que j'évolue par moi-même pour que je puisse devenir un développeur autonome, en étant patient et en m'expliquant mes erreurs.

Les suites envisagées sont de nouveaux modèles qui remplaceront les pages PHP qui permettent de créer ou modifier un article.