

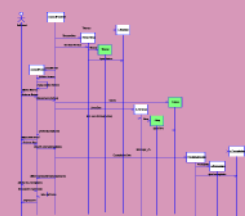
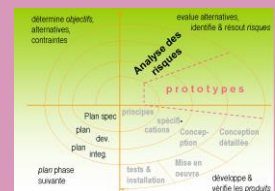
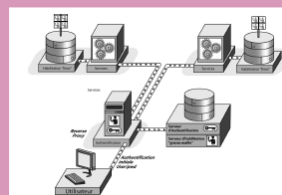
Secteur Tertiaire Informatique
Filière étude - développement

Projet

Dossier de Conception technique IHM Web/ Programmation PHP

Equipe :
(nom et logo)

Participants :
-Maxime
-Antoine
-Francis
-Léa



SOMMAIRE

I	CONTEXTE	4
I.1.1	Reformulation du besoin et User Stories développées	4
I.1.2	Plateforme de développement	4
I.1.3	La base de données utilisée	5
II	LA CONCEPTION IHM.....	8
II.1.1	Graphe du dialogue	8
III	CONCEPTION et développement DES COMPOSANTS	9
III.1.1	Les fonctionnalités	Erreur ! Signet non défini.
III.1.2	Maquettes	Erreur ! Signet non défini.
III.1.3	Les ressources utilisées et particularités de la Story .	Erreur ! Signet non défini.
III.1.4	Code développé.....	Erreur ! Signet non défini.

I CONTEXTE

I.1.1 Reformulation du besoin et User Stories développées

User Story1 : Lister les clients
User Story2 : Lister les clients par page
User Story3 : Rechercher un client
User Story4 : Contacter un client
User Story5 : Mettre à jour les données d'un client
User Story6 : Supprimer un client
User Story7 : Lister les collaborateurs
User Story8 : Rechercher un collaborateur
User Story9 : Créer un utilisateur
User Story12 : Se connecter
User Story13 : Se déconnecter
User Story14 : S'inscrire
User Story 15 : Lister les projets
User Story16 : Lister les collaborateurs par page
User Story17 : Lister les projets par page

I.1.2 Plateforme de développement

Niveau	Produit	Remarques
Système d'exploitation	Windows 10	
SGBD	MySQL	
EDI	Visual Code Studio	
Langage(s)	HTML5/CSS3/JavaScript(Ajax/Jquery) /PHP7.4	
Autres	Xampp	Inclut Apache version... et PHP version...

I.1.3 La base de données utilisée

phpMyAdmin

Récentes Préférées

Nouvelle base de données

- abi
- blog
- blogpool
- information_schema
- mydb
- mysql
- performance_schema
- phpmyadmin
- test
- wordpress

Serveur: 127.0.0.1 Base de données: abi

Structure SQL Rechercher Requête Exporter Importer Opérations Privileges Procédures stockées Plus

Filtres

Contenant le mot :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> contact		100	InnoDB	utf8mb4_general_ci	32,0 kio	-
<input type="checkbox"/> customer		99	InnoDB	utf8mb4_general_ci	64,0 kio	-
<input type="checkbox"/> document		200	InnoDB	utf8mb4_general_ci	80,0 kio	-
<input type="checkbox"/> employee		99	InnoDB	utf8mb4_general_ci	32,0 kio	-
<input type="checkbox"/> project		101	InnoDB	utf8mb4_general_ci	32,0 kio	-
<input type="checkbox"/> project_employee		99	InnoDB	utf8mb4_general_ci	48,0 kio	-
<input type="checkbox"/> user		104	InnoDB	utf8mb4_general_ci	48,0 kio	-
7 tables	Somme	802	InnoDB	utf8mb4_general_ci	336,0 kio	0

☐ Tout cocher Avec la sélection :

Imprimer Dictionnaire de données

Nouvelle table

Nom: Nombre de colonnes: 4

Récentes Préférées

Nouvelle base de données

- abi
- Nouvelle table
- contact
- customer
- document
- employee
- project
- project_employee
- user
- blog
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test

Parcourir Structure SQL Rechercher Insérer Exporter Importer Privileges Opérations Plus

	id	login	password	role	email
<input type="checkbox"/> Éditer Copier Supprimer	105	Maksco	\$2y\$10\$A1p14Kc./ZzGiv4qXw7Jz uwq8.6O9ycVL1O0FRbD9...	Commercial	dedede.ee@dmr.ru
<input type="checkbox"/> Éditer Copier Supprimer	104	Maks	\$2y\$10\$4UE1caWtSj1t1YlcZd7NuwOe1s0jxObtqe1nwd58Z...	Visiteur	dedede.ee@mmm.ru
<input type="checkbox"/> Éditer Copier Supprimer	102	Jenna	\$2y\$10\$DFjh.3oi3r2V8uAVESOGdO1cTzoE.MDQwB4TCQxwMg...	Visiteur	dupont.marie@online.fr
<input type="checkbox"/> Éditer Copier Supprimer	101	Lea	\$2y\$10\$ro/GuTWtEAQkv3TNnD72ju/z1XkRWEV/cwJ76bnFlag...	Visiteur	lea.lele@wanadoo.fr
<input type="checkbox"/> Éditer Copier Supprimer	100	Amélie	&["&/U,w@.o]]	Commercial	salmon.adele@noos.fr
<input type="checkbox"/> Éditer Copier Supprimer	99	Robert	\$2y\$10\$ScHhcJzScdMVKG8Z7c4ZyuuFqzWEVYHwsxHbHrNyLds...	Secrétariat Technique	fbriand@voila.fr
<input type="checkbox"/> Éditer Copier Supprimer	98	Suzanneuh	\$2y\$10\$Jw7DMA5aT8T1umc8mXDIguWEcBBCG6HwWtdpwLxin59...	Technicien Support	valerie.clement@gomes.net
<input type="checkbox"/> Éditer Copier Supprimer	97	Honoré	\$2y\$10\$z1xv11NMNajrosJPu0WEBedH/a1lx1NoV8we74xp6zk...	Technicien Support	daniel.techet@guillet.com
<input type="checkbox"/> Éditer Copier Supprimer	96	Dominique	\$2y\$10\$m639KRntylA98tLnWwyBcedJME1ViueRai.h7EydWC/...	Commercial	talbert@fournier.fr
<input type="checkbox"/> Éditer Copier Supprimer	95	Lucyy	\$2y\$10\$hrOovNg1krEvQ4RRWlieemmMC62PhoBbeoE.HvgPi...	Commercial	alexandre93@gilles.fr
<input type="checkbox"/> Éditer Copier Supprimer	94	Diane	\$2y\$10\$SoLTGkHhpaWuhWa0hQFJvuxsBloFCacbjcAFA1vDPCj...	Chef de Projet	oceane.vincent@guillon.com
<input type="checkbox"/> Éditer Copier Supprimer	93	Suzanno	\$2y\$10\$Uacv1IUC5X.cVbShjBF.hFY95lGpxckJbRmq5zdRa...	Commercial	denise20@gilbert.com

employee

project

project_employee

user

blog

information_schema

mysql

performance_schema

phpmyadmin

test

+ Options

	id	lastname	firstname	role	salary	contract	hired	dismissed	user_id
<input type="checkbox"/> Éditer Copier Supprimer	201	Guibert	Marguerite	Directeur	1994	CDI	2005-08-14 00:00:00	NULL	1
<input type="checkbox"/> Éditer Copier Supprimer	202	Turpin	Valentine	RH	3818	CDI	2018-05-25 00:00:00	NULL	2
<input type="checkbox"/> Éditer Copier Supprimer	203	Lamy	Olivie	RC	2985	CDI	1993-10-07 00:00:00	NULL	3
<input type="checkbox"/> Éditer Copier Supprimer	204	Ramos	Pénélope	Secrétaire	2455	CDD	1992-09-06 00:00:00	NULL	4
<input type="checkbox"/> Éditer Copier Supprimer	205	Pelletier	Stéphane	RDD	1392	CDI	2007-06-30 00:00:00	2018-05-30 00:00:00	5
<input type="checkbox"/> Éditer Copier Supprimer	206	Marion	Clémence	Ingénieur	2443	CDI	2011-05-16 00:00:00	NULL	6
<input type="checkbox"/> Éditer Copier Supprimer	207	Antoine	Charles	Ingénieur	1872	CDI	1990-01-13 00:00:00	NULL	7
<input type="checkbox"/> Éditer Copier Supprimer	208	Bruneau	Christiane	Ingénieur	3951	Interim	1983-09-24 00:00:00	NULL	8
<input type="checkbox"/> Éditer Copier Supprimer	209	Pages	Alfred	Ingénieur	1393	CDD	2020-01-07 00:00:00	NULL	9
<input type="checkbox"/> Éditer Copier Supprimer	210	Bertin	Catherine	Ingénieur	2044	CDI	2009-06-01 00:00:00	NULL	10
<input type="checkbox"/> Éditer Copier Supprimer	211	Learos	Alphonse	Ingénieur	2536	CDI	2014-10-03 00:00:00	NULL	11

- + project_employee
 - + user
- + blog
- + information_schema
- + mysql
- + performance_schema
- + phpmyadmin
- + test

+ Options

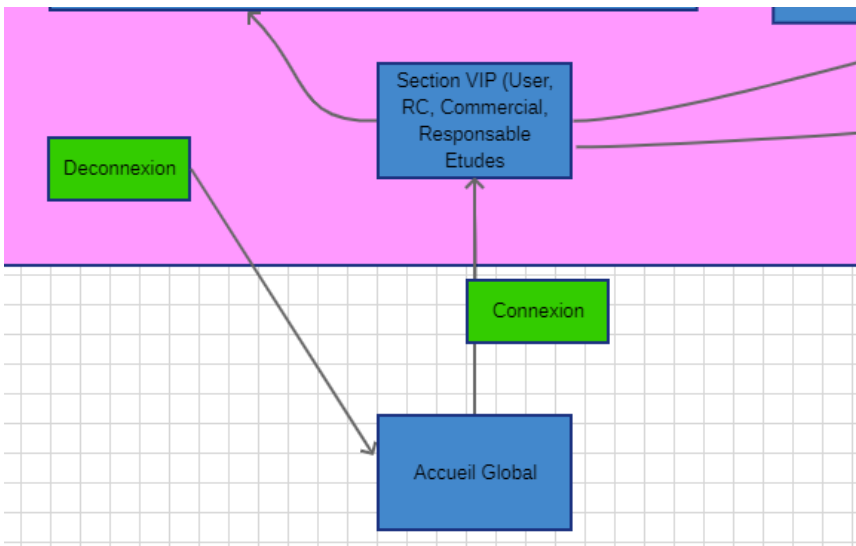
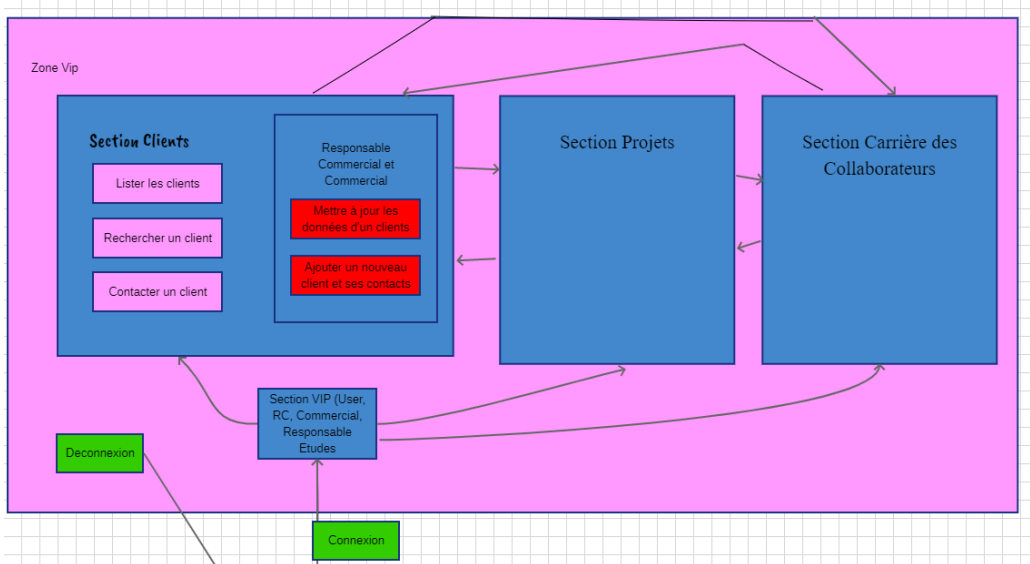
project_id	employee_id
1	217
1	256
1	249
1	271
1	263
2	269
2	262
2	206

- + project
 - + project_employee
 - + user
 - + blog
 - + information_schema
 - + mysql
 - + performance_schema
 - + phpmyadmin
 - + test

	id	shortname	fullname	type	started	customer_id
<input type="checkbox"/> Éditer Copier Supprimer	1	THéODOR	Théodore Ferreira	Sous-traitance	2015-01-16 00:00:00	79
<input type="checkbox"/> Éditer Copier Supprimer	2	ANDRéC	André Colin	Sous-traitance	2013-09-06 00:00:00	26
<input type="checkbox"/> Éditer Copier Supprimer	3	DAVIDDE	David de Schmitt	Conseil	1990-12-31 00:00:00	11
<input type="checkbox"/> Éditer Copier Supprimer	4	ROLANDL	Roland Lopez	Sous-traitance	1973-11-12 00:00:00	56
<input type="checkbox"/> Éditer Copier Supprimer	5	RENéBI	René Bigot	Sous-traitance	2007-10-21 00:00:00	89
<input type="checkbox"/> Éditer Copier Supprimer	6	THOMASD	Thomas de la Lacroix	Sous-traitance	1975-06-25 00:00:00	80
<input type="checkbox"/> Éditer Copier Supprimer	7	MAGGIE-N	Maggie-Nathalie Poirier	Sous-traitance	2002-01-25 00:00:00	93
<input type="checkbox"/> Éditer Copier Supprimer	8	OCéANE	Océane Robert-Arnaud	Développement	1991-02-22 00:00:00	33
<input type="checkbox"/> Éditer Copier Supprimer	9	ISAACBE	Isaac Becker	Développement	1993-10-30 00:00:00	15
<input type="checkbox"/> Éditer Copier Supprimer	10	ISAACGI	Isaac Gilles	Sous-traitance	2013-08-09 00:00:00	35
<input type="checkbox"/> Éditer Copier Supprimer	11	ÉTIENNE	Étienne Leduc	Conseil	1986-07-15 00:00:00	69
<input type="checkbox"/> Éditer Copier Supprimer	12	SUZANNE-	Suzanne-Suzanne Grenier	Développement	1997-01-11 00:00:00	33
<input type="checkbox"/> Éditer Copier Supprimer	13	THéOPHI	Théophile de la Da Costa	Développement	1997-01-28 00:00:00	62
<input type="checkbox"/> Éditer Copier Supprimer	14	AUGUSTIN	Augustin-Michel Allard	Conseil	1994-12-20 00:00:00	93

II LA CONCEPTION IHM

II.1.1 Graphe du dialogue



III CONCEPTION ET DEVELOPPEMENT DES COMPOSANTS

Gestion par Controller

Avant de rentrer dans le détail des User Stories développées dans le cadre de ce projet il est bon de préciser que ce dernier a été réalisé avec une architecture MVC et un router.

Cela signifie que techniquement, l'utilisateur se trouve toujours sur la page index.php dont le contenu est géré dynamiquement par les différentes méthodes de notre classe Controller.

```
<?php
/**
 * Viewer by Controller
 * @author Maxime
 */

include_once '../vendor/autoload.php';
use App\Controller\Controller;

$uri = $_SERVER["REQUEST_URI"];

/*
Nous effectuons plusieurs opérations sur l'uri récupérée.
Nous lui ôtons son premier caractère (le "/")
En cas de variables "GET", nous retirons tous les caractères à partir du "?"
Enfin, nous ajoutons à la chaîne "Controller"
*/
if (strpos($uri, "?")){
    $uri = substr_replace($uri, "", strpos($uri, "?"));
}
$uri = substr($uri, 1) . "Controller";

/*
Une fois ces opérations effectuées, nous faisons appel à notre Controller
qui appellera la méthode adéquate selon le contenu de la barre d'url.
*/
if($uri ==="Controller"){
    Controller::homeController();
} else if (method_exists(Controller::class, $uri)) {
    Controller::$uri();
} else {
    Controller::error404Controller();
}
```

[User Story1 : Lister les clients](#)

[User Story7 : Lister les collaborateurs](#)

[User Story 15 : Lister les projets](#)

L'utilisateur doit être en mesure d'utiliser la plate-forme afin de consulter la liste des Clients, des Collaborateurs et des Projets.

Chacun de ses aspects possède une page dédiée accessible uniquement aux utilisateurs loggés via un header et une navigation adaptée.

L'affichage de ces différentes listes est géré dans un premier temps via des méthodes de notre Controller qui prendra soin de bien vérifier que l'utilisateur essayant d'accéder à la page soit bien loggé en vérifiant sa session.

Si l'utilisateur est accepté, nous faisons appel au Repository correspondant (ici, CustomerRepository, responsable de la recherche des entités « Customer » via la méthode mère « findBy ».

Controller :

```
/**
 * customersController: Controller de la page customer.php,
 * Permet d'authentifier le visiteur,
 * afficher une table avec sa pagination et la possibilité de faire une recherche dans celle-ci
 *
 * @return void
 */
public static function customersController()
{
    session_start();

    if(!isset($_SESSION['login'])) {
        header('Location:/logplz');
    }
    ob_start();

    $customerRepository = new CustomerRepository();

    $arraysult = $customerRepository->paginate('company_name');
    if(isset($_GET['search'])) {
        $customers = $customerRepository->searching($_GET['search'],["id"=>"ASC"],$arraysult[0], $arraysult[1], 'company_name');
    }
    else {
        $customers = $customerRepository->findBy([],["id"=>"ASC"],$arraysult[0], $arraysult[1]);
        $_GET['search'] = null;
    }
}
```

```

        $currentPage = $arraysult[2];
        $pages = $arraysult[3];
        include '../templates/customers.php';

        ob_end_flush();
    }

```

Par défaut, nous allons donc récupérer les 10 premiers éléments de la table correspondante afin d'en préparer l'affichage et les convertissons en Entity selon le cas : les clients deviennent par exemple des instances de l'entité Customer.

Les Entités possèdent des propriétés qui correspondent aux colonnes de leur table respectives ce qui permet d'instancier aisément nos objets directement au bon format grâce à l'attribut FETCH_CLASS de PDO.

Classe mère abstraite « Repository » :

```

abstract class Repository {
    protected PDO $pdo;
    private string $table;
    private string $classname;

    public function findBy(array $criteria, array $orderBy = null, int $limit = null,
        int $offset = null)
    {
        $params = array_values($criteria);
        if (!empty($criteria)) {
            $criteria = " WHERE " . join(" AND ", array_map(fn($key) => "$key = ?
", array_keys($criteria)));
        } else {
            $criteria = null;
        }
        if ($orderBy) $orderBy = " ORDER BY " . join(", ", array_map(fn($key, $va
lue) => "$key $value", array_keys($orderBy), array_values($orderBy)));
        if ($limit) $limit = " LIMIT $limit";
        if ($offset) $offset = " OFFSET $offset";
        $sql = "SELECT * FROM $this->table $criteria $orderBy $limit $offset";
        $query = $this->pdo->prepare($sql);
        $query->execute($params);
        //dd($sql);
        return $query->fetchAll(PDO::FETCH_CLASS, "App\Entity\\"$this-
>classname");
    }
}

```

Une fois les dix premières lignes de la table récupérées, l'affichage est effectué grâce à une boucle php directement dans le code HTML de la page.

```

<table class="table table-sm table-striped table-hover bg-secondary">
    <thead class="thead-light">

```

```

        <tr>
            <th>Id</th>
            <th>Raison Sociale</th>
            <th>#Dpt</th>
        </tr>
    </thead>
    <tbody id="clientsListe" role="button" class="text-white ">

        <?php foreach ($customers as $customer) : ?>

            <tr onclick="select_customer(<?= $customer->getId() ?>)">
                <th><?= $customer->getId() ?></th>
                <th><?= $customer->getCompanyName() ?></th>
                <th><?= substr($customer->getZip(), 3) ?></th>
            </tr>

        <?php endforeach ?>

    </tbody>
</table>

```

Entité « Customer »

```

namespace App\Entity;

use App\Traits\JsonTrait;

class Customer
{
    use JsonTrait;

    private int $id;
    private string $company_name;
    private string $address;
    private string $zip;
    private string $city;
    private int $turnover;
    private string $phone;
    private string $type;
    private string $nature;
    private string $comment;
    private string $activity;
}

```

Index (header pour User loggé)

Abi

Accueil

Nos Clients

Nos Projets



Nos Collaborateurs

Bienvenue Maks
Visiteur

Déconnexion

Qui Sommes-Nous ?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ad, voluptates corporis laudantium cupiditate omnis corrupti! Corrupti doloremque similique veritatis exercitationem eum voluptatum est incidunt temporibus, aperiam iste praesentium non accusamus.



Nous Trouver




60 rue de l'Inexistence
35700 RENNES

Notre Histoire

Lorem ipsum dolor sit amet consectetur adipisicing elit. Odio temporibus mollitia distinctio quam ea ut aspernatur doloribus a aliquam officia ullam repellendus dolore molestias enim, odit aperiam tenetur. Soluta, officiis! Lorem ipsum dolor sit amet consectetur adipisicing elit. Odio temporibus mollitia distinctio quam ea ut aspernatur doloribus a aliquam officia ullam repellendus dolore molestias enim, odit aperiam tenetur. Soluta, officiis!

Lorem ipsum dolor sit amet consectetur adipisicing elit. Odio temporibus mollitia distinctio quam ea ut aspernatur doloribus a aliquam officia ullam repellendus dolore molestias enim, odit aperiam tenetur. Soluta, officiis! Lorem ipsum dolor sit amet consectetur adipisicing elit. Odio temporibus mollitia distinctio quam ea ut aspernatur doloribus a aliquam officia ullam repellendus dolore molestias enim, odit aperiam tenetur. Soluta, officiis!

Suivez-nous !



Informations Générales

- Conditions
- Politique de confidentialité
- Cookies

@2020 Abi, Inc

Nous Contacter

Webmaster :
abifictive@gmail.com

Page « Clients »

Abi

Accueil

Nos Clients

Nos Projets

Nos Collaborateurs

Bienvenue Maks
Visiteur

Déconnexion

Nos clients

Rechercher un Client

Id	Raison Sociale	#Dpt
1	Herve SAS	35
2	Hernandez Bruneau SAS	61
3	Leveque et Fils	01
5	Pons Dupuy SA	06
6	Petit	82
7	Ledoux	11
8	Hardy	85
9	Normand S.A.	76
11	Monnier	81
12	Payet S.A.S.	51

|<

<

1

2

3

>

>|

Informations sur le client

Adresse

CP

Ville

Téléphone

Domaine

Nature

CA

Effectif

ID Client

Type

Nos Contacts avec ce Client

Nom	Prénom	Email
-----	--------	-------

Voir tous les contacts

Commentaires

Abi

Accueil

Nos Clients

Nos Projets

Nos C

Nos Projets

Rechercher un Client

Code	Nom du projet
1	THÉODOR
2	ANDRÉC
3	DAVIDDE
4	ROLANDL
5	RENÉBI
6	THOMASD
7	MAGGIE-N
8	OCÉANE
9	ISAACBE
10	ISAACGI

|<

<

1

2

3

>

>|

Client

Contact

Type de contract

Date de début

Tâches réalisés

User Story

Abi

Accueil

Nos Clients

Nos Projets

Nos Collaborateurs

Bienvenue Maks Visiteur

Déconnexion

Nos collaborateurs

Rechercher un Employée

Matricule	Nom
201	Guibert
202	Turpin
203	Lamy
204	Ramos
205	Pelletier
206	Marion
207	Antoine
208	Bruneau
209	Pages
210	Bertin

|<

<

1

2

3

>

>|

Informations sur le collaborateur

Nom	Matricule	Photo
Contrat	Avenant	
École	Agence d'intérim	
Date d'embauche		
Date de fin de contract		
Qualification principal courante		Fonction
Salaire mensuel		
Modifier des données		
Ajouter un collaborateur		

[User Story16 : Lister les collaborateurs par page](#)

[User Story17 : Lister les projets par page](#)

[User Story2 : Lister les clients par page](#)

Afin de faciliter la lecture des informations, les clients, collaborateurs et projets sont affichés par lot de 10 par page ce qui est géré par la méthode de Repository->paginate() qui se chargera de faire les calculs nécessaires selon les circonstances :

- une recherche a-t-elle effectuée ?
- combien de résultats sont attendus ?
- sur quelle page nous trouvons-nous actuellement.

Elle renverra un tableau contenant le résultat de ces calculs dans des variables ainsi exploitables par le Controller.

```
/**
 * paginate: permet l'affichage avec pagination d'un tableau contenant toutes
 les entrées d'une table
 * l'argument d'entrée est le nom de la colonne qui sera triée par ordre alph
abétique
 * @param mixed $column
 * @return [$nbPerPage, $first, $currentPage, $pages] les variables de ce tab
leau pourront être utilisées dans le Controller afin de permettre
 * un affichage correct de la pagination
 */
public function paginate($column) {
    if(isset($_GET['page']) && !empty($_GET['page'])){
        $currentPage = (int) strip_tags($_GET['page']);
    }else{
        $currentPage = 1;
    }

    if(isset($_GET['search']) && !empty($_GET['search'])){
        $search = "WHERE $column LIKE '{$_GET['search']}'%";
    }else{
        $search = null;
    }

    $sql = "SELECT COUNT(*) AS nb_rows FROM $this->table $search";
    $query = $this->pdo->prepare($sql);
    $query->execute();
    $result = $query->fetch();

    $nbRows = $result['nb_rows'];
    $nbPerPage = 10;
    $pages = ceil($nbRows/$nbPerPage);
    $first = ($currentPage * $nbPerPage) - $nbPerPage;
    if ($first<1) {
        $first = null;
    }
}
```



```
return [$nbPerPage, $first, $currentPage, $pages];
```

Controller :

```
$customerRepository = new CustomerRepository();

$arraysult = $customerRepository->paginate('company_name');
if(isset($_GET['search'])) {
    $customers = $customerRepository-
>searching($_GET['search'],["id"=>"ASC"],$arraysult[0], $arraysult[1], 'company_n
ame');
}
else {
    $customers = $customerRepository-
>findBy([],["id"=>"ASC"],$arraysult[0], $arraysult[1]);
    $_GET['search'] = null;
}

$currentPage = $arraysult[2];
$pages = $arraysult[3];
include '../templates/customers.php';
```

Enfin, l'affichage de la navigation est assuré en PHP directement dans le code HTML.

```
<ul class="pagination justify-content-center">
  <li class="page-item <?= ($currentPage == 1) ? "disabled" : "" ?>">
    <a href="/customers?page=<?= 1 . "&search" . $_GET['search'] ?>" clas
s="page-link"> |< </a>
  </li>
  <li class="page-item <?= ($currentPage == 1) ? "disabled" : "" ?>">
    <a href="/customers?page=<?= ($currentPage - 1) . "&search=" . $_GET['s
earch'] ?>" class="page-link"> < </a>
  </li>

  <?php for ($page = ($currentPage-2);$page < $currentPage+3; $page++) : ?>
    <?php if ($page>0 && $page <= $pages) : ?>

    <li class="page-item <?= ($currentPage == $page) ? "active" : "" ?>">
      <a href="/customers?page=<?= $page . "&search=" . $_GET['search'] ?>" c
lass="page-link"><?= $page ?></a>
    </li>
    <?php endif ?>
  <?php endfor ?>

  <li class="page-item <?= ($currentPage == $pages) ? "disabled" : "" ?>">
    <a href="/customers?page=<?= ($currentPage + 1) . "&search=" . $_GET['sea
rch'] ?>" class="page-link"> > </a>
```



```

        <li class="page-item <?= ($currentPage == $pages) ? "disabled" : "" ?>">
            <a href="/customers?page=<?= $pages . "&search" . $_GET['search'] ?>" c
lass="page-link"> >| </a>
        </li>
    </ul>

```

Boutons de pagination :

20	Boucher et Fils	81	30	Menard S.A.S.	90
21	Imbert S.A.R.L.	84	31	Pascal Hernandez et Fils	66
22	Lombard Guyot S.A.S.	33	32	Boulangier	39

|<
<
1
2
3
4
>
>|

|<
<
1
2
3
4
5
>
>|

49	Peron SARL	04	100	Herve	43
50	Michel SA	70	103	dede	11
51	Dijoux	09			
52	Raynaud et Fils	08			

|<
<
3
4
5
6
7
>
>|

|<
<
8
9
10
>
>|

Par souci de lisibilité on remarque que la page actuelle est toujours en surbrillance tandis que les options indisponibles sont grisées afin de guider l'utilisateur.

De même, un maximum de 5 pages numérotées est proposé :

- les deux pages précèdent la page actuelle (le cas échéant)
- la page actuelle en surbrillance
- les deux pages suivantes (le cas échéant)

Chacun de ces boutons enverra sur la page « /customers ?page=*numéro de la page désirée* » ce qui permet de récupérer l'information dans une simple variable `$_GET['page']`.

Bien que le « get » ne soit pas une méthode recommandée, elle est ici justifiée par le caractère inoffensif et des informations passées.

User Story3 : Rechercher un client

User Story8 : Rechercher un collaborateur

La recherche est mise à disposition en en-tête des différentes listes.

Sa requête est transmise en get afin d'être récupérée dans la variable \$_GET['search'].

Le Controller ainsi que la méthode Repository->paginate() vérifient tous deux l'existence de la variable \$_GET['search'] afin d'adapter leur exécution.

Ici, le CustomerRepository ne fera donc pas appel à Repository->findBy() mais à la méthode ->searching() en prenant en argument non seulement les variables \$arraysults transmises par paginate mais aussi la recherche de l'utilisateur qui sera utilisée dans la requête SQL.

(WHERE \$column LIKE '{\$search}%')

Les résultats ainsi « fetchés » correspondront strictement à la demande la pagination s'adaptera au nouveau nombre de résultat.

Controller :

```
$customerRepository = new CustomerRepository();

$arrayresult = $customerRepository->paginate('company_name');
if(isset($_GET['search'])) {
    $customers = $customerRepository->
>searching($_GET['search'],["id"=>"ASC"],$arrayresult[0], $arrayresult[1], 'company_n
ame');
}
```

Repository->searching()

```
/**
 * searching: permet d'effectuer des recherches à l'intérieur d'une table affichée
 *
 *
 * @param mixed $search
 * @param mixed $orderBy
 * @param mixed $limit
 * @param mixed $offset
 * @param mixed $column
 * @return void
 */
public function searching(string $search, array $orderBy = null, int $limit = null, int $offset = null, string $column)
{
    if ($orderBy) $orderBy = " ORDER BY " . join(", ", array_map(fn($key, $value) => "$key $value", array_keys($orderBy), array_values($orderBy)));
    if ($limit) $limit = " LIMIT $limit";
    if ($offset) $offset = " OFFSET $offset";
```

```

        $sql = "SELECT * FROM $this->table WHERE $column LIKE '{$search}%' $orderBy $limit $offset";
        $query = $this->pdo->prepare($sql);
        $query->execute();
        return $query->fetchAll(PDO::FETCH_CLASS, "App\Entity\\".$this->classname");
    }

```

Exemple de recherche des clients commençant par la lettre « p » :

localhost/customers?search=p#

Nos clients		
p		
Id	Raison Sociale	#Dpt
5	Pons Dupuy SA	06
6	Petit	82
12	Payet S.A.S.	51
27	Paris	91
31	Pascal Hernandez et Fils	66
39	Paul Hoareau SA	56
49	Peron SARL	04
56	Picard	90

|<
 <
 1
 >
 >|

User Story4 : Contacter un client

L'affichage des données d'un client est assuré en Ajax afin de faciliter et d'accélérer la consultation du contenu.

En effet, chacune des lignes de la liste générées en php comprend un évènement onclick qui fera appelle à « select_customer(id) ».

```
<tr onclick="select_customer(<?= $customer->getId() ?>)">
```

Javascript :

```
/**
 * Assure l'affichage du détail des informations du client sélectionné.
 * @param {int} id
 */
function select_customer(id) {
    let inputs = document.getElementById('clientAdresse');
    if (!inputs.disabled){
        annulation(); // Si le mode d'édition est actif, on le quitte.
    }

    const xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            //La réponse est un objet json que l'on parse pour l'exploiter et assurer
            l'affichage.
            let customer = JSON.parse(this.response);
            document.getElementById('clientName').innerHTML = customer.company_name;
            document.getElementById('clientAdresse').value = customer.address;
            document.getElementById('clientCp').value = customer.zip;
            document.getElementById('clientVille').value = customer.city;
            document.getElementById('clientTelephone').value = customer.phone;
            document.getElementById('clientDomaine').value = customer.activity;
            document.getElementById('clientNature').value = customer.nature;
            document.getElementById('clientCa').value = customer.income;
            document.getElementById('clientEffectif').value = customer.workforce;
            document.getElementById('clientCommentaire').value = customer.comment;
            document.getElementById('clientType').innerHTML = customer.type;
            document.getElementById('clientId').innerHTML = customer.id;

            //On récupère également un tableau d'échantillon de Contacts correspondant
            t au Client (max. 3)
            document.getElementById('contactsListe').innerHTML = "";
            if (customer.contacts.length > 0) {
                customer.contacts.forEach(contact => {
                    let row = '<tr>\n
                        <th scope="col">' + contact.lastname + '</th>\n
                    <td>' + contact.firstname + '</td>\n
                    <td>' + contact.phone + '</td>\n
                    <td>' + contact.email + '</td>\n
                    </tr>';
                    document.getElementById('contactsListe').innerHTML += row;
                });
            }
        }
    };

    xhr.open('GET', 'api/client/' + id, true);
    xhr.setRequestHeader('Content-Type', 'application/json');
    xhr.send();
}
```

```

                <th scope="col">'+ contact.firstname + '</th>\
                <th scope="col"><a href="mailto:'+ contact.email + '">
'+ contact.email + '</a></th>\
                </tr>';

        document.getElementById('contactsListe').innerHTML += row;
    })
}
}
};

xhr.open("POST", "/selectcustomer", true);
xhr.setRequestHeader("Content-type", "application/json");
xhr.send(JSON.stringify({ id: id }));
}

```

Controller :

```

public static function selectcustomerController(){
    $customerRepository = new CustomerRepository();
    $data = file_get_contents("php://input");
    $data = json_decode($data);
    $customer = $customerRepository->find($data->id);

    $contactRepository = new ContactRepository();
    $contacts = $contactRepository->findSampleByCustomer($customer);
    $contacts = array_map(fn($contact)=>$contact->toJSONArray(), $contacts);
    $customer->contacts = $contacts;

    echo $customer->toJson();
}

```

L'Id du client sélectionné est donc pris en paramètre par la fonction JS avant d'être envoyé au Controller ::selectcustomerController afin d'être traité en Php. Dans cet exemple, Vanilla JS a été favorisé ce qui nous contraint d'aller chercher notre id non pas dans le \$POST mais dans php://input pour un résultat sensiblement identique.

Les données sont décodé puis exploité par un objet CustomerRepository dont la méthode ->find() permet d'aller trouver dans notre base de données le Client à l'Id correspondant et retourne un objet « Customer » .

Ce n'est pas tout : nous récupérons également un échantillon de trois Contacts (maximum) rangé dans un tableau \$contacts.

Nous transformons ces objets en tableaux de variables via la méthode ->toJSONArray().

```

trait JsonTrait {

    /**
     *Encode l'entité en Json passable via une méthode Ajax.

```

```

    * @return string
    */
    public function toJson(){
        return json_encode(get_object_vars($this));
    }
    /**
     *Retourne un tableau associatif contenant les propriétés (y compris privées)
     de l'objet.
     * @return array
     */
    public function toJsonArray(){
        return get_object_vars($this);
    }
}

```

En procédant ainsi, nous obtenons toutes les propriétés de l'objet (y compris privées !). Ce tableau de variables est ensuite « rattaché » à notre Customer et en devient une propriété. Ceci est fait afin de faciliter le transfert en Json qui s'ensuit.

Nous utilisons ensuite la méthode ->toJson afin d'envoyer notre objet complexe sous la forme d'une simple chaîne. Cette réponse est JSON, parsé en javascript et les données peuvent enfin être exploitées et affichées en manipulant le DOM.

L'entreprise peut ensuite être contactée par téléphone ou, encore mieux, en cliquant simplement sur l'e-mail d'un de ses Contacts.

Nos clients			Paul Hoareau SA			
Rechercher un Client			Adresse	4, rue Marin	ID Client	39
Id	Raison Sociale	#Dpt	CP	33456	Type	Publique
5	Pons Dupuy SA	06	Ville	Martin-sur-Carpentier	Nos Contacts avec ce Client	
6	Petit	82	Téléphone	0386754172	Nom	Prénom
12	Payet S.A.S.	51	Domaine	Vente	Jacob	Claire
27	Paris	91	Nature	Secondaire		
31	Pascal Hernandez et Fils	66	CA	901	Robin	Valérie
39	Paul Hoareau SA	56	Effectif	1215		
49	Peron SARL	04			Voir tous les contacts	
56	Picard	90			Commentaires	
<div> <div><</div> <div><</div> <div>1</div> <div>></div> <div>></div> </div>			<div> <div>Aut non quis architecto a eaque unde eos expedita totam aut aut aperiam inventore dolore cum aut quae odio occaecati ex</div> <div></div> </div>			

Nos Contacts avec ce Client		
Nom	Prénom	Email
Boulay	Roland	victoire45@bouygtel.fr
Letellier	Isaac	robin.julien@noos.fr

[Voir tous les contacts](#)

User Story5 : Mettre à jour les données d'un client

Tous les utilisateurs n'ont pas accès aux fonctionnalités d'édition pour des raisons évidentes de sécurité. Aussi, pour ce qui est de la Section « clients », seuls les Commerciaux et Responsables Commerciaux verront apparaître sur leur interface les boutons « Ajouter un Client » et « Modifier un Client » dont la présence ou absence est assurée en PHP via un contrôle de la session utilisateur et plus précisément de son « rôle ».

```
<?php if ($_SESSION['role'] === "Commercial" || $_SESSION['role'] === "RC") : ?>
    <button class="btn btn-danger" type="button" data-
toggle="modal" data-target="#editModal">Ajouter un
    Client</button>
    <?php endif; ?>
```

```
<?php if ($_SESSION['role'] === "Commercial" || $_SESSION['role'] === "RC") : ?>
    <button class="btn btn-
danger" id="modif" onclick="editClient()" type="button">Modifier
    Client</button>
    <button class="btn btn-info mt-
3" id="annuler" style="display:none" onclick="annulation()" type="button">Annuler
</button>
    <button class="btn btn-success mt-
3" id="valider" style="display:none" onclick="validClient()" type="button">Enregi-
strer</button>
    <button class="btn btn-warning mt-
3" id="delete" style="display:none" onclick="deleteClient()" type="button">Suppri-
mer</button>
    <?php endif; ?>
```

Affichage pour utilisateurs loggés

Abi

Accueil

Nos Clients

Nos Projets

Nos Collaborateurs

Bienvenue Maks

Visiteur

Déconnexion

Nos clients

Rechercher un Client

Id	Raison Sociale	#Dpt
1	Herve SAS	35
2	Hernandez Bruneau SAS	61
3	Leveque et Fils	01
5	Pons Dupuy SA	06
6	Petit	82
7	Ledoux	11
8	Hardy	85
9	Normand S.A.	76
11	Monnier	81
12	Payet S.A.S.	51

|<

<

1

2

3

>

>|

Leveque et Fils

Adresse

41, impasse Philippine Breton

ID Client

3

CP

41001

Type

Publique

Ville

Fouquetdan

Nos Contacts avec ce Client

Téléphone

0716916256

Nom

Prénom

Email

Domaine

Conseil

Voir tous les contacts

Nature

Secondaire

Commentaires

CA

5951

Consectetur alias et distinctio officiis quia officia qui dolorum natus quos accusamus laboriosam blanditiis et iste accusantium

Effectif

1184

Affichage pour les Commerciaux et RC

Abi

Accueil

Nos Clients

Nos Projets

Nos Collaborateurs

Bienvenue Maksco

Commercial

Déconnexion

Nos clients

Rechercher un Client

Ajouter un Client

Id	Raison Sociale	#Dpt
1	Herve SAS	35
2	Hernandez Bruneau SAS	61
3	Leveque et Fils	01
5	Pons Dupuy SA	06
6	Petit	82
7	Ledoux	11
8	Hardy	85
9	Normand S.A.	76
11	Monnier	81
12	Payet S.A.S.	51

|<

<

1

2

3

>

>|

Leveque et Fils

Adresse

41, impasse Philippine Breton

ID Client

3

CP

41001

Type

Publique

Ville

Fouquetdan

Nos Contacts avec ce Client

Téléphone

0716916256

Nom

Prénom

Email

Domaine

Conseil

Voir tous les contacts

Nature

Secondaire

Commentaires

CA

5951

Consectetur alias et distinctio officiis quia officia qui dolorum natus quos accusamus laboriosam blanditiis et iste accusantium

Effectif

1184

Modifier Client

Si l'on essaye de rentrer dans le mode d'édition sans avoir au préalable sélectionné un client, une alerte vient nous prévenir.

Informations sur le client

Adresse

CP

Ville

Téléphone

Département

Nature

CA

Effectif

ID Client

Type

Nos Contacts avec ce Client

Nom	Prénom	Email
Voir tous les contacts		

Commentaires

Modifier Client

Aucun client sélectionné.
OK

Si un client a bien été sélectionné, les inputs jusqu'ici « disabled » deviennent actif et permettent la manipulation des données en direct.

Annuler : aucune modification ne sera sauvegardée et l'on quitte le mode édition.

Enregistrer : si les modifications apportées sont acceptées par javascript, php et sql, alors dans ce cas le Client sera effectivement mis à jour dans la base de données.

Leveque et Fils

Adresse

41, impasse Philippine Breton

CP

41001

Ville

Fouquetdan

Téléphone

0716916256

Domaine

Conseil

Nature

Secondaire

CA

5951

Effectif

1184

Annuler

Enregistrer

Supprimer

ID Client

3

Type

Publique

Nos Contacts avec ce Client

Nom

Prénom

Email

[Voir tous les contacts](#)

Commentaires

Consectetur alias et distinctio officiis quia officia qui dolorum natus quos accusamus laboriosam blanditiis et iste accusantium

Pour que les modifications soient acceptées, un certains nombres de critères doivent être remplis, tout d'abord en javascript via un premier contrôle.

Pour cela deux fonctions JS travaillent ensemble afin de non-seulement vérifier la conformité des saisies mais assurent également un affichage dynamique afin de guider l'utilisateur dans son travail :

```

/**
 * User Input Control Function
 * @param {string} input The Id of the input which is to be verified.
 * @param {string} regex The method of control which is to be applied.
 */
function verifNewInput(input, regex) {
    if (regex.test($(input).val())) {
        $(input).removeClass('is-invalid').addClass('is-valid');
    }
    else {
        $(input).removeClass('is-valid').addClass('is-invalid');
    }
}
/**
 * This function filters the information needed by the verifNewInput function.
 */
$('.editableClient, .editableModal').focusout(function () {

```

```

/**
 * @type {string} The selected input's id is stored in this variable.
 */
let idInput = $(this).attr('id');
/**
 * @type {Object} Empty variable which is to accept one of seven possible reg
ular expressions.
 */
let regexAuto;
/**
 * @type {Array<string>} Array including keywords which help to identify if t
he current input
 * needs to be verified by the verfString regex.
 */
let tabInputName = ["Adresse", "Domaine", "RaisonSociale", "Ville"];
/**
 * @type {Array<string>} Array including keywords which help to identify if t
he current input
 * needs to be verified by the verfInteger regex.
 */
let tabInputInteger = ["Ca", "Effectif"];

let verfString = new RegExp('^[a-zA-Z0-9À-ÿ.\'\,\_\s-][a-zA-Z0-9À-ÿ. \'\,\_\s-]{0,40}[a-zA-Z0-9À-ÿ.\'\_\s-]$$');
let verfCP = new RegExp('^([0-9]{5})$$');
let verfTelephone = new RegExp('^((([0-9]{2} )){4}[0-9]{2})$|^([0-9]{10})$$');
let verfMail = new RegExp('^([a-zA-Z]{1,})([a-zA-Z._-]{0,1})[a-zA-Z]+@[a-zA-Z-
Z]+\.[a-zA-Z]{2,6})$$');
let verfInteger = new RegExp('^([1-9]{1,}[0-9]{0,})$$');
let verfNature = new RegExp('^principale$|^secondaire$|^ancienne$', 'i');
let verfCommentaire = new RegExp('^([a-zA-Z0-9À-ÿ. \'\_!\?\\,\s-]{0,255})$$');

//Considering the current input (idInput), we determine which regex needs to
be applied.
if (tabInputName.some(el => idInput.includes(el))) {
    regexAuto = verfString;
} else if (tabInputInteger.some(el => idInput.includes(el))) {
    regexAuto = verfInteger;
} else if (idInput.includes('Cp')) {
    regexAuto = verfCP;
} else if (idInput.includes('Email')) {
    regexAuto = verfMail;
} else if (idInput === 'clientCommentaire') {
    regexAuto = verfCommentaire;
} else if (idInput.includes('Telephone')) {
    regexAuto = verfTelephone;
} else if (idInput === 'clientNature') {
    regexAuto = verfNature;
}

```

```
verifNewInput('#' + idInput, regexAuto);
```

```
});
```

The screenshot shows a registration form with the following fields and values:

- Adresse:** 48, place Claude Ruiz (Valid, green checkmark)
- CP:** 868ddede (Invalid, red border and warning icon)
- Ville:** Pires
- Téléphone:** 0511484132
- Domaine:** IT
- Nature:** Secondaire (Valid, green checkmark)
- CA:** 922
- Effectif:** 2661

At the bottom, there are three buttons: "Annuler" (blue), "Enregistrer" (green), and "Supprimer" (yellow).

Afin que des modifications puissent être acceptées et traitées par PHP, il faut qu'au moins un input soit considéré « isvalid » tandis que le moindre champ « isInvalid » empêchera l'enregistrement des modifications.

```
/**  
 * Checks every single .editableClient inputs.
```

```

* If all are valid, enables the user to save the modification to the targeted client.
* Triggers when clicking on the "Enregistrer" button (#valider).
*/
function validClient() {
    if ($.editableClient.is-valid).length > 0 &&
        ($.editableClient.is-invalid).length === 0) {

        let update = {
            id: $('#clientId').html(),
            companyName: $('#clientName').html(),
            type: $('#clientType').html(),
            address: $('#clientAdresse').val(),
            zip: $('#clientCp').val(),
            city: $('#clientVille').val(),
            phone: $('#clientTelephone').val(),
            activity: $('#clientDomaine').val(),
            nature: $('#clientNature').val(),
            turnover: $('#clientCa').val(),
            workforce: $('#clientEffectif').val(),
            comment : $('#clientCommentaire').val(),
        }

        $.post('/updatecustomer', update, function(reponse) {
            if (reponse === "1") {
                alert('Client mis à jour avec succès !');
                annulation();
                ($.editableModal).val('').removeClass('is-valid');
                $('#editModal').modal('hide');
                location.replace('/customers');
            } else {
                alert(reponse);
            }
        })
    }
    else {
        alert('Client non-conforme.');
```

Si les conditions sont remplies les données des différents inputs sont envoyées en Json via Ajax à la méthode Controller ::updatecustomerController.

```

public static function updatecustomerController() {
    if ($_POST['id']) {
        $updateCustomer = new UpdateCustomerForm($_POST);
        echo $updateCustomer->updateCustomer();
    } else {
```

```

        header('Location: /');
    }
}

```

C'est donc un objet UpdateCustomerForm qui prend le relai et se charge de la seconde save de contrôles.

```

class UpdateCustomerForm
{
    private string $company_name;
    private string $address;
    private string $zip;
    private string $city;
    private int $income;
    private int $workforce;
    private string $phone;
    private string $type;
    private string $nature;
    private string $comment;
    private string $activity;
    private int $id;

    /**
     * On crée un constructeur dans laquelle on va passer la méthode POST et on va
     * récupérer les informations que l'on a besoin
     */
    public function __construct($post)
    {
        $this->company_name = htmlspecialchars($post['companyName']);
        $this->address = htmlspecialchars($post['address']);
        $this->zip = (string) htmlspecialchars(trim($post['zip']));
        $this->city = htmlspecialchars($post['city']);
        $this->income = (int) $post['turnover'];
        $this->workforce = (int) $post['workforce'];
        $this->phone = (string) htmlspecialchars(trim($post['phone']));
        $this->type = htmlspecialchars(trim(ucfirst($post['type'])));
        $this->nature = htmlspecialchars(trim(ucfirst($post['nature'])));
        $this->comment = $post['comment'];
        $this->activity = htmlspecialchars($post['activity']);
        $this->id = (int)$post['id'];
    }

    /**
     * Contrôle les données envoyées via formulaire avant de mettre à jour le Customer ciblé.
     * @return bool|string
     */
}

```

```

public function updateCustomer()
{
    if (
        !preg_match('/^[a-zA-Z0-9À-ÿ.\'\,\_s-][a-zA-Z0-9À-ÿ. \'\,\_s-]{0,40}[a-zA-Z0-9À-ÿ.\'\,\_s-]$/', $this->company_name) ||
        $this->address === "" ||
        !preg_match('/^[0-9]{5}$/', $this->zip) ||
        $this->city === "" ||
        $this->income === "" ||
        !preg_match('/^([0-9]{2} )^{4}[0-9]{2})$|^[0-9]{10}$/', $this->phone) ||
        !preg_match('/(Publique|Privée)/', $this->type) ||
        !preg_match('/(^Principale|Secondaire|Ancienne)$/', $this->nature) ||
        $this->activity === ""
    ) {
        return "Client non-conforme.";
    } else {
        $pdo = new \PDO(DbConfig::DSN, DbConfig::USERNAME, DbConfig::PASSWORD);

        $param = get_object_vars($this);
        unset($param['id']);
        $param = join(", ", array_map(fn($key, $value) => "$key = '$value'", array_keys($param), array_values($param)));
        $sql = "UPDATE customer SET $param WHERE id = $this->id";
        $query = $pdo->prepare($sql);
        return $query->execute([$param]);
    }
}
}

```

On applique un certain nombre de fonction sur chacune des données envoyées afin de s'assurer de leur bon format mais aussi de se prévenir des injections SQL. En cas de problème, on renvoie un message d'erreur en guise de réponse à notre script javascript.

Si tout est bon et que « true » est retourné, l'utilisateur sera prévenu du succès de son opération.

Ledoux

Adresse

48, rue Claude Ruiz

✓

CP

86811

Ville

Pires

Téléphone

0511484132

Domaine

IT

Nature

Secondaire

CA

922

Effectif

2661

Annuler

Enregistrer

Supprimer

ID Cl

Type

Nos

Nom

Aube

Voir t

Com

Franc

Ledoux

Adresse

48, rue Claude Ruiz

✓

CP

86811

Ville

Té

Do

Nature

Secondaire

CA

922

Effectif

2661

Annuler

Enregistrer

Supprimer

ID Cl

Type

Nos

Nom

Aube

Voir t

Com

Franc

Client mis à jour avec succès !

OK

User Story6 : Supprimer un client

Si c'est la suppression qui est choisi par l'utilisateur, ce dernier sera prévenu par une alerte afin de s'assurer qu'il ne s'agit pas d'une erreur de sa part ou d'un simple miss-click.

En cas de confirmation, une requête ajax contenant l'id du client sélectionné sera envoyé au Controller qui effectuera la méthode deletecustomerController qui a son tour créera un objet DeleteCustomerForm dont la seule propriété sera l'id passé.

C'est cet objet via sa méthode deletecustomerController() qui se chargera d'effectuer la requête DELETE en SQL. .

JS :

```
function deleteClient() {  
  
    let confirmation = confirm('Ce client sera définitivement supprimé. Continuer ?')  
    if (confirmation){  
        let toDelete = {  
            id: $('#clientId').html()  
        }  
        $.post('/deletecustomer', toDelete, function(reponse) {  
            alert('Client supprimé !');  
        })  
        annulation();  
    }  
};
```

Controller :

```
public static function deletecustomerController() {  
    if ($_POST['id']) {  
        $deleteCustomer = new DeleteCustomerForm($_POST);  
        echo $deleteCustomer->deleteCustomer();  
    } else {  
        header('Location: /');  
    }  
}
```

Classe DeleteCustomerForm :

```
class DeleteCustomerForm  
{  
    private int $id;  
  
    public function __construct($post)  
    {  
        $this->id = (int)$post['id'];  
    }  
}
```

```

/**
 * Supprime le Customer sélectionné.
 * Fonctionnalité d'édition réservée aux Users ayant les rôles "Commercial" et "RC".
 * @return void
 */
public function deleteCustomer()
{
    $pdo = new \PDO (DbConfig::DSN, DbConfig::USERNAME, DbConfig::PASSWORD);

    $sql = "DELETE FROM customer WHERE id = $this->id";
    $query = $pdo->prepare($sql);
    $query->execute();
    dd($query);
}
}

```

Nos clients

Id	Raison Sociale	#Dpt
1	Herve SAS	35
2	Hernandez Bruneau SAS	61
3	Leveque et Fils	01
5	Pons Dupuy SA	06
6	Petit	82
7	Ledoux	11
8	Hardy	85
9	Normand S.A.	76
11	Monnier	81
12	Payet S.A.S.	51

Leveque et Fils

Adresse

41, impasse Philippine Breton

ID Client

CP

41001

Type

Ville

Nos Conta

Nature

Secondaire

Nom

CA

5951

Voir tous les con

Effectif

1184

Commenta

Annuler

Enregistrer

Supprimer

Ce client sera définitivement supprimé. Continuer ?

Nos clients

Id	Raison Sociale	#Dpt
1	Herve SAS	35
2	Hernandez Bruneau SAS	61
5	Pons Dupuy SA	06
6	Petit	82
7	Ledoux	11

User Story9 : Créer un Client

Contrairement à l'édition de client, la création d'un nouveau Client fait intervenir une interface « modal » dont la structure permet de guider l'utilisateur et de lui faire comprendre le type de données qui est attendu.

Comme toujours, deux contrôle sont effectués.

Si le formulaire répond aux attentes de notre code Javascript (cf « Mettre à jour un Client ») alors on effectue une requête Ajax en transmettant en POST les informations saisies par l'utilisateur.

```
$("#formNewCustomer").submit(function (event) {
    event.preventDefault();
    if ($('#editableModal.is-valid').length == $('#editableModal').length) {
        let newCustomer = {
            companyName: $('#newRaisonSociale').val(),
            type: $('#input[name=newType]:checked').val(),
            address: $('#newAdresse').val(),
            zip: $('#newCp').val(),
            city: $('#newVille').val(),
            phone: $('#newTelephone').val(),
            activity: $('#newDomaine').val(),
            nature: $('#newNature option:selected').val(),
            turnover: $('#newCa').val(),
            workforce: $('#newEffectif').val(),
            form: "ok"
        }

        $.post('/addcustomer', newCustomer, function (reponse) {
            if (reponse === "1") {
                alert('Nouveau client ajouté avec succès');
                $('#editableModal').val("").removeClass('is-valid');
                $('#editModal').modal('hide');
                location.replace('/customers');
            } else {
                alert(reponse);
            }
        });
    }
    else {
        alert('Client non-conforme.');
```

On instancie un objet AddCustomerForm en lui faisant passer notre \$_POST dans le constructeur afin de récupérer toutes les données et de les traiter.

```
class AddCustomerForm
```

```

{
    private string $company_name;
    private string $address;
    private string $zip;
    private string $city;
    private int $turnover;
    private int $workforce;
    private string $phone;
    private string $type;
    private string $nature;
    private string $comment;
    private string $activity;

    public function __construct($post)
    {
        $this->company_name = htmlspecialchars($post['companyName']);
        $this->address = htmlspecialchars($post['address']);
        $this->zip = (string) htmlspecialchars(trim($post['zip']));
        $this->city = htmlspecialchars($post['city']);
        $this->turnover = (int) $post['turnover'];
        $this->workforce = (int) $post['workforce'];
        $this->phone = (string) htmlspecialchars(trim($post['phone']));
        $this->type = htmlspecialchars($post['type']);
        $this->nature = htmlspecialchars($post['nature']);
        $this->comment = "";
        $this->activity = htmlspecialchars($post['activity']);
    }

    /**
     *Ajoute un nouveau Customer à la base de données si les données saisies répon
ndent à nos attentes.
     *En cas d'erreur, retourne un message approprié à destination de l'utilisateur.
     * @return bool|string
     */
    public function addToDatabase()
    {
        if (
            !preg_match('/^[a-zA-Z0-9À-ÿ.\'\"_\\s-][a-zA-Z0-9À-ÿ. \\'\"_\\s-]{0,40}[a-zA-Z0-9À-ÿ.\'\"_\\s-]$/', $this->company_name) ||
            $this->address === "" ||
            !preg_match('/^[0-9]{5}$/', $this->zip) ||
            $this->city === "" ||
            $this->turnover === "" ||
            !preg_match('/^([0-9]{2} )?{4}[0-9]{2}$|^([0-9]{10})$/', $this->phone) ||
            !preg_match('/(Publique|Privée)/', $this->type) ||
            !preg_match('/(Principale|Secondaire|Ancienne)/', $this->nature) ||

```

```

        $this->activity === ""
    ) {
        return "Client non-conforme.";
    } else {
        $pdo = new \PDO(DbConfig::DSN, DbConfig::USERNAME, DbConfig::PASSWORD
);

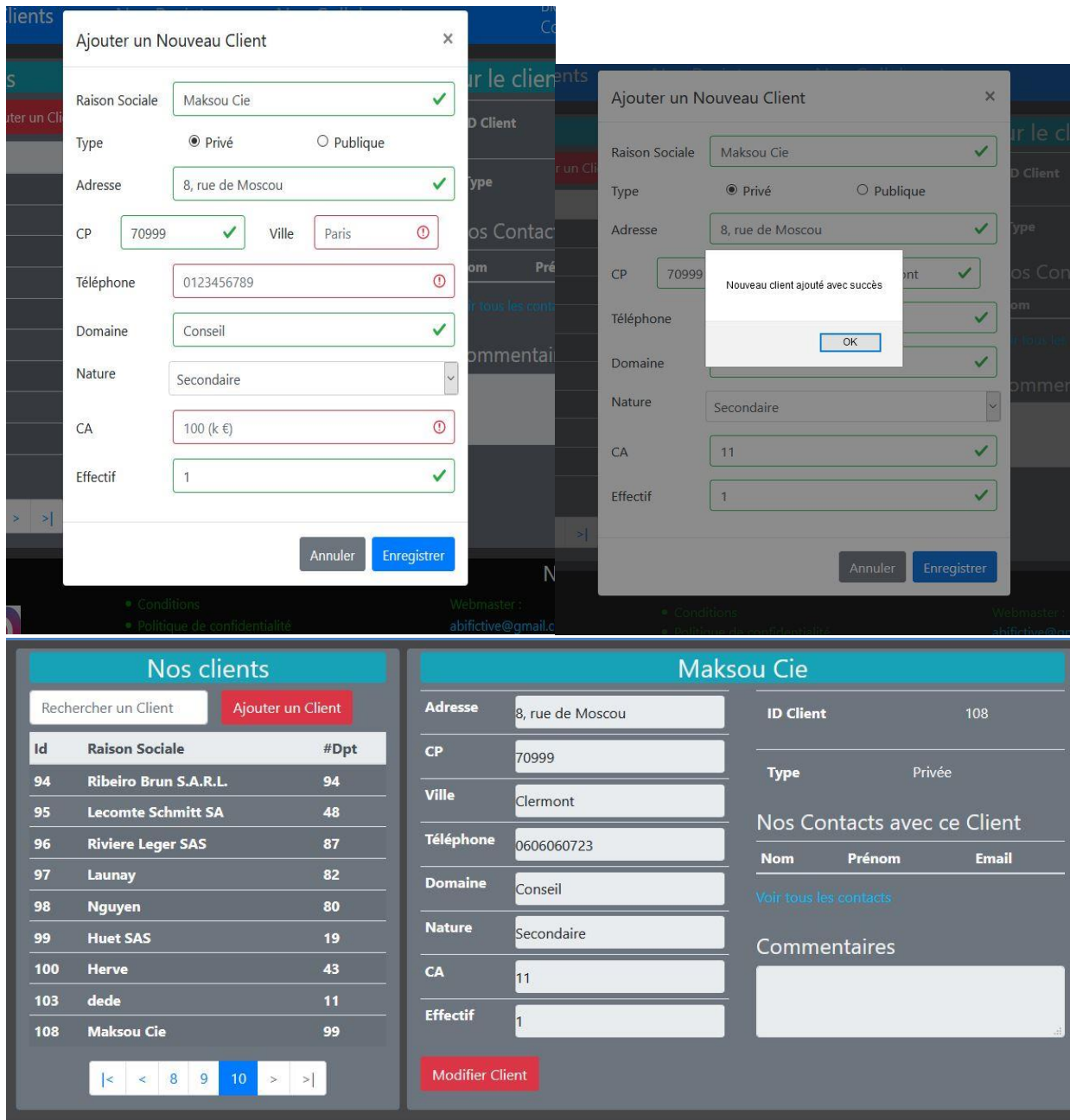
        $sql = "SELECT COUNT(*) AS Nb_results FROM customer WHERE company_nam
e = ?";

        $stmt = $pdo->prepare($sql);
        $stmt->execute([$this->company_name]);
        $result = $stmt->fetch();

        if ($result['Nb_results'] > 0) {
            return "Ce client existe déjà.";
        } else {
            $sql = "INSERT INTO customer
                (company_name, address, zip, city, income, workforce, phone, type
, nature, comment, activity)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
            $stmt = $pdo->prepare($sql);
            return $stmt->execute(array_values(get_object_vars($this)));
        }
    }
}
}
}
}

```

Si le client ainsi créé satisfait l'ensemble de nos contrôles et n'existe pas déjà dans notre base de données, alors il sera ajouté à la table customer et l'utilisateur sera remercié.



User Story12 : Se connecter

Si aucune session n'a été initialisée, l'utilisateur n'aura accès à aucune section « sensible » de la plate-forme et le header l'invitera à s'authentifier.

Abi

Active Bretagne Informatique

S'enregistrer

Envoyer

Accueil



Actualités

F.A.Q

Recrutement

Qui Sommes-Nous ?

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad, voluptates corporis laudantium cupiditate omnis corrupti! Corrupti doloremque similique veritatis exercitationem eum voluptatum est incidunt temporibus, aperiam iste praesentium non accusamus.

S'enregistrer

Envoyer

Identifiant ou mot de passe invalide

L'authentification est gérée en Ajax pour le confort de l'utilisateur et éviter les rafraichissements de page en cas d'erreur.

Cliquer sur « Envoyer » exécutera le code suivant :

JS :

```

/**
 * Fonction connexion qui permet de se connecter a la page accueilvip en tant qu'
utilisateur
 * @param {string} inputEmail
 * @param {string} inputPassword
 * @param {boolean} connect
 */

function connexion() {
    let inputidentifiant = $('#inputIdentifiant').val(); // inputEmail prend la val
eur entrée dans #inputEmail
    let inputPassword = $('#inputPassword').val(); // inputPassword prend la valeur
entrée dans #inputPassword
    let connect = false;
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {

```

```

    if (xhr.readyState == 4 && xhr.status == 200) {
        if (xhr.responseText === "ok") {
            location.href = "/";
        } else {
            $("#message").html("Identifiant ou mot de passe invalide")
        };
    }
};
var data = 'password=' + inputPassword + '&identifiant=' + inputidentifiant;
xhr.open('POST', '/login', true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send(data);
}

```

Controller :

```

public static function loginController()
{
    session_start();

    if (!empty($_POST) && isset($_POST['identifiant'])) {
        $signin = new LoginForm($_POST);
        $signin->authentification();
    }
}

```

LoginForm :

```

class LoginForm {

    private string $identifiant;
    private string $password;

    /**
     * On créer un constructeur dans laquelle on passe la méthode POST
     * Récupérer les informations dont l'on a besoins
     */
    public function __construct($post)
    {
        $this->identifiant = htmlspecialchars($post['identifiant']);
        $this->password = trim($post["password"]);
    }

    /**
     * Assure la connexion d'un utilisateur en vérifiant sa présence dans la base
     de données
     * et la concordance avec le mot de passe saisi.
     */
}

```



```

* @return void
*/
public function authentication(){

    $UserRep = new UserRepository();
    $result = $UserRep->authenticate($this->identifiant);
    if(empty($result)) {
        echo("connection invalide");
    } else {
        if(password_verify($this->password, $result["password"])){
            $_SESSION['login'] = $result['login'];
            $_SESSION['role'] = $result['role'];
            echo "ok";
        } else {
            echo("connection invalide");
        }
    }
}
}
}

```

On vérifie que le nom saisi par l'utilisateur existe bien dans notre base de données des Users. Si c'est le cas, nous procédons à vérifier l'exactitude et la concordance de son mot de passe.

En cas d'erreur, l'utilisateur est averti mais n'obtient pas de détails sur les raisons de l'échec afin de ne pas donner de piste à un éventuel attaquant. En cas de réussite, une session est créée avec le login de l'utilisateur et le rôle qui lui est associé, lui donnant accès à d'avantage de contenu et fonctionnalités.

[User Story13 : Se déconnecter](#)

Si un utilisateur connecté souhaite mettre fin à sa session, il lui suffit de suivre le lien « Déconnexion » qui fera appel à la méthode suivante :

```

public static function disconnectController(){
    session_start();
    session_destroy();
    header("location:/");
}

```

Bienvenue Maksco
Commercial

Déconnexion

Sa session sera détruite et il reviendra sur la page d'accueil en tant que simple visiteur.

User Story14 : S'inscrire

Les utilisateurs non-loggés sont invités à s'authentifier mais également à créer un compte si celui-ci n'existe pas déjà.

Par sécurité, les utilisateurs déjà authentifiés ne peuvent accéder à cette partie du site.

En remplissant ce formulaire, encore protégé par une couche de javascript, il est possible d'envoyer une requête Ajax et de traiter la demande en Php.

Informatique

[Accueil](#)[Actualités](#)[F.A.Q](#)[Recrutement](#)

Nous Rejoindre

Identifiant

Charlottea ✓

Email

dedede.ee@dmm.ru ✓

Mot de passe

..... ✓

Confirmer le mot de passe

... !

S'inscrire

JS :

```
$("#registerForms").submit(function (event) {
    event.preventDefault();
    let identifiant = $(this).find("input[name=identifiant]").val();
    let email = $(this).find("input[name=email]").val();
    let password = $(this).find("input[name=password]").val();
    let confirmPassword = $(this).find("input[name=confirmPassword]").val();

    /**
     * Cette requete AJAX nous permet de relier le PHP et le JS pour faire fonctionner les deux ensembles
     */
    if (identifiantIsValid(identifiant) && emailIsValid(email) && passwordIsValid(password) && confirmPasswordIsValid(confirmPassword)) {
        $.post("/registration", {
            identifiant: identifiant,
            email: email,
            password: password,
            confirmPassword: confirmPassword
        }, function (reponse) {
            console.log(reponse);
            if (reponse === "true") {
                location.href = "/";
            } else if (reponse.includes("Identifiant") !== false) {
                $('#getError').html(reponse);
                $("#identifiant").removeClass('is-valid');
                $("#identifiant").addClass('is-invalid');
            } else if (reponse.includes("Email") !== false){
                $('#getError').html(reponse);
                $("#email").removeClass('is-valid');
                $("#email").addClass('is-invalid');
            } else if (reponse.includes("Le mot de passe") !== false) {
                $('#getError').html(reponse);
                $("#password").removeClass('is-valid');
                $("#password").addClass('is-invalid');

                $("#confirmPassword").removeClass('is-valid');
                $("#confirmPassword").addClass('is-invalid');
            }
        });
    } else {
        $("#registerForms input").each(function () {
            $(this).focusout();
        })
    }
})
```

Si les inputs sont acceptés par le code JS, leur contenu est transmi via Ajax et le \$_POST résultant est utilisé pour instancier un objet RegisterForm :

```
class RegisterForm
{

    private string $identifiant;
    private string $email;
    private string $password;
    private string $confirmPassword;

    /**
     * On crée un constructeur dans laquelle on va passer la méthode POST et on va récupérer les informations que l'on a besoin
     */
    public function __construct($post)
    {
        $this->identifiant = htmlspecialchars(trim($post['identifiant']));
        $this->email = htmlspecialchars(strtolower(trim($post['email'])));
        $this->password = trim($post['password']);
        $this->confirmPassword = trim($post['confirmPassword']);
    }

    /**
     *Contrôle la saisie utilisateur issue du formulaire d'inscription.
     *Retour un message d'erreur approprié en cas de problème.
     *Procède à l'enregistrement du nouvel utilisateur en cas de réussite.
     * @return bool|string
     */
    public function register()
    {
        if ($this->identifiant === "") return "Identifiant invalide.";
        if ($this->email !== filter_var($this->email)) return "Email invalide.";
        if ($this->password !== $this->confirmPassword) {
            return "Le mot de passe et sa confirmation sont différents !";
        } else {
            $this->password = password_hash($this->password, PASSWORD_BCRYPT);
        }

        $pdo = new PDO (DbConfig::DSN, DbConfig::USERNAME, DbConfig::PASSWORD);

        $query = $pdo->prepare("SELECT user.* FROM user WHERE email = ?");
        $query->execute([$this->email]);
        $results = $query->fetchAll(PDO::FETCH_ASSOC);
        if (!empty($results)) return "Email déjà utilisée.";
```

```

$query = $pdo->prepare("SELECT user.* FROM user WHERE login = ?");
$query->execute([$this->identifiant]);
$results = $query->fetchAll(PDO::FETCH_ASSOC);
if (!empty($results)) return "Identifiant déjà pris.";

$query = $pdo-
>prepare("INSERT INTO user (login,password,email, role) VALUES (?,?,,?)");
$query->execute([$this->identifiant, $this->password, $this-
>email, "Visiteur"]);

$_SESSION['login'] = $this->identifiant;
$_SESSION['role'] = "Visiteur";
return "true";

}
}

```

Si les données sont correctes et que l'email et/ou l'identifiant choisis par l'utilisateur sont libres, nous effectuons son enregistrement et créons immédiatement une session afin de lui laisser jouir de ses nouveaux privilèges.

Par défaut, tout utilisateur inscrit possède un rôle « Visiteur » au plus faible niveau d'accès pour des raisons de sécurité.

A ce stade du développement, il est impossible de modifier le rôle d'un user via notre plate-forme : il faut donc faire cela manuellement en passant par PhpMyAdmin.

