



# DOSSIER PROFESSIONNEL (DP)

*Nom de naissance* ▶ SONDER  
*Nom d'usage* ▶  
*Prénom* ▶ Frederick  
*Adresse* ▶ 1 rue de Négron  
13118 Entressen

## Titre professionnel visé

*Concepteur Développeur d'Application*

### MODALITÉ D'ACCÈS :

- Parcours de formation
- Validation des Acquis de l'Expérience (VAE)

# DOSSIER PROFESSIONNEL (DP)

## Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.  
**Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE. Il est consulté par le jury au moment de la session d'examen.

### Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel (DP)** dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

*[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]*

### Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

# DOSSIER PROFESSIONNEL (DP)

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

 <http://travail-emploi.gouv.fr/titres-professionnels>

## Sommaire

### Exemples de pratique professionnelle

<b>Développer des composants d'interface utilisateur</b>	p.	<b>5</b>
» <i>Fondation 1ocean</i>	p.	p.
» <i>Remmedia</i>	p.	p.
<b>Développer la persistance des données</b>	p.	<b>20</b>
» Intitulé de l'exemple n° 1	p.	p.
» Intitulé de l'exemple n° 3	p.	p.
<b>Développer une application multicouche</b>	p.	<b>28</b>
» <i>Remmedia</i> pour le moment=>	p.	p.
» Intitulé de l'exemple n° 3	p.	p.
<b>Titres, diplômes, CQP, attestations de formation (facultatif)</b>	p.	<b>34</b>
<b>Déclaration sur l'honneur</b>	p.	<b>35</b>
<b>Documents illustrant la pratique professionnelle (facultatif)</b>	p.	<b>36</b>
<b>Annexes (Si le RC le prévoit)</b>	p.	<b>37</b>

---

# **EXEMPLES DE PRATIQUE**

## **PROFESSIONNELLE**

# DOSSIER PROFESSIONNEL<sup>(DP)</sup>

---

## Activité

- type** Développer des composants d'interface utilisateur en  
**1** intégrant les recommandation de sécurité

*exemple n°2 REMMEDIA Backoffice*

---

---

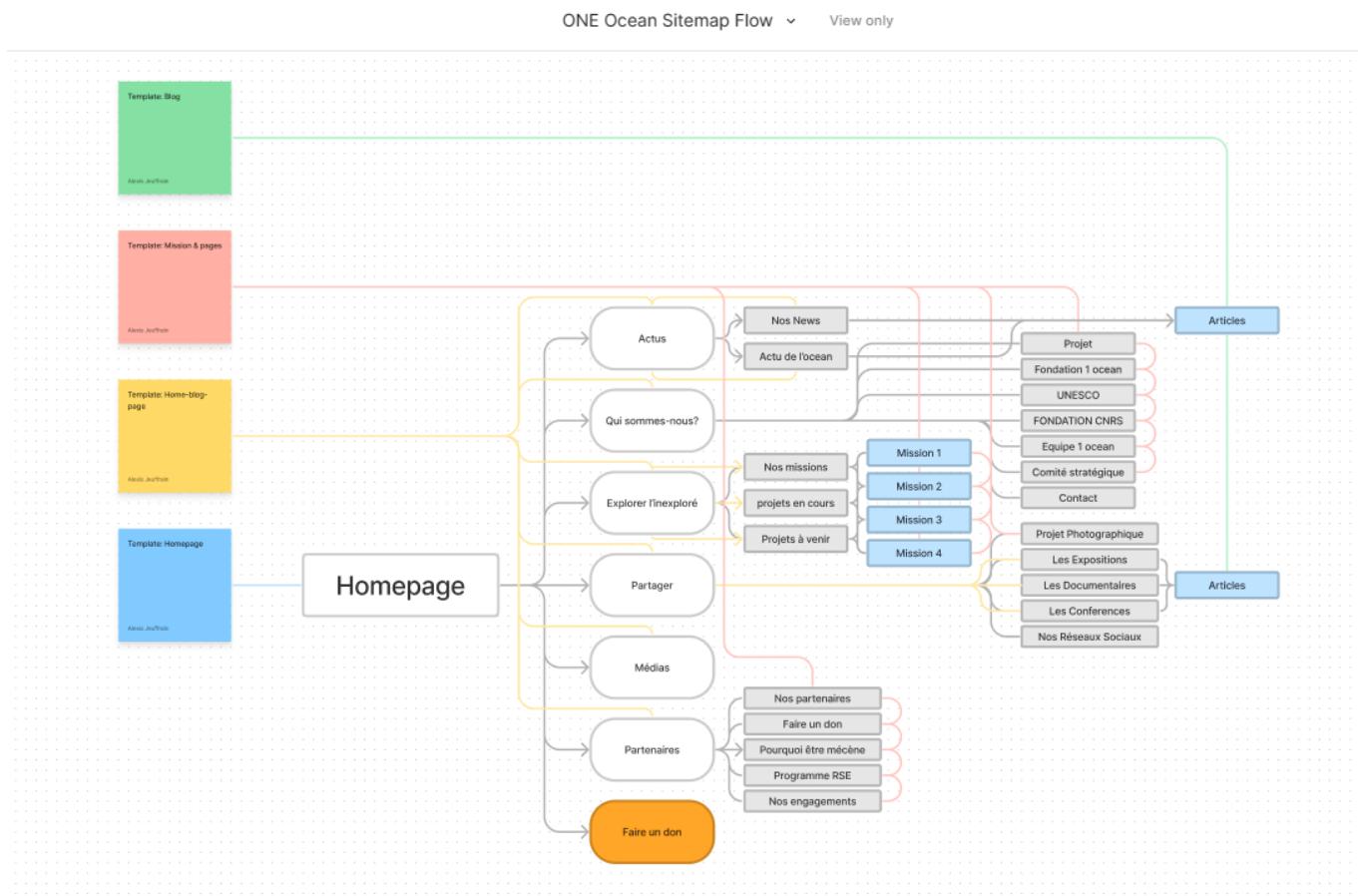
# DOSSIER PROFESSIONNEL (DP)

**La fondation 1ocean** est une fondation sous l'égide de la **fondation CNRS** et travaillant avec l'**UNESCO**.

Elle a pour but de faciliter la communication sur l'exploration, l'évolution et la protection du milieu marin.

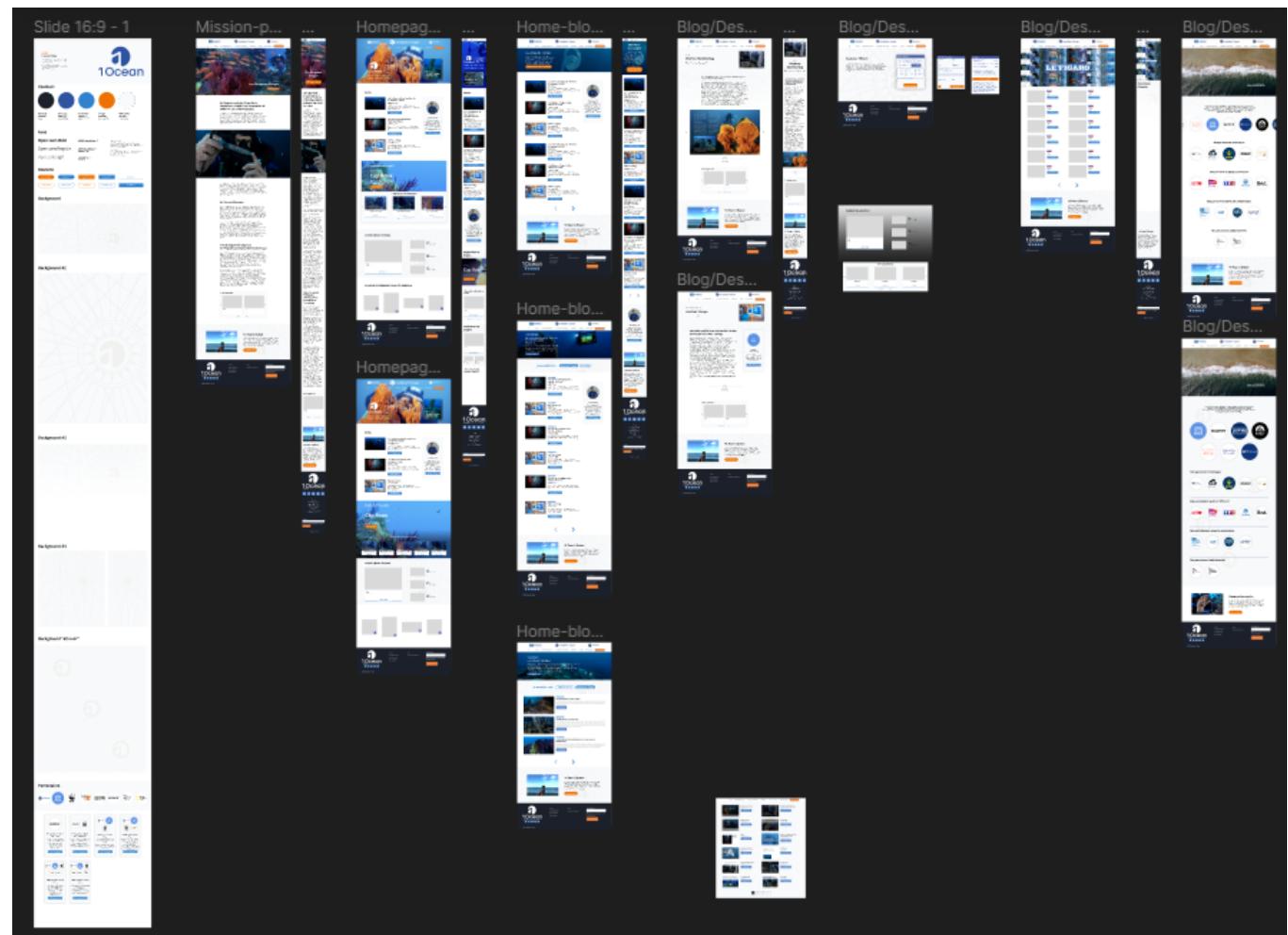
Après le recueil des besoins de la fondation, nous avons défini un ensemble de scénarios utilisateurs en décrivant en détail ce qu' Alexis Rosenfeld (*Photographe, photojournaliste, plongeur professionnel et Project Leader* de la fondation) désirait ainsi que ce que les utilisateurs pourraient faire sur le site et pourquoi ils le font.

Ces éléments nous ont servi de base pour mettre en place la navigation entre les interfaces :



Ainsi que pour la création d'une maquette:

# DOSSIER PROFESSIONNEL (DP)



Dans ce projet, il nous a été demandé de réaliser un site internet avec un parcours utilisateur simple et plusieurs demandes particulières:

- un backoffice :
  - gestion et administration des différents projets (passés, en cours, ou à venir),
  - gestion et administration des différentes parutions médiatiques,
  - gestion des dons

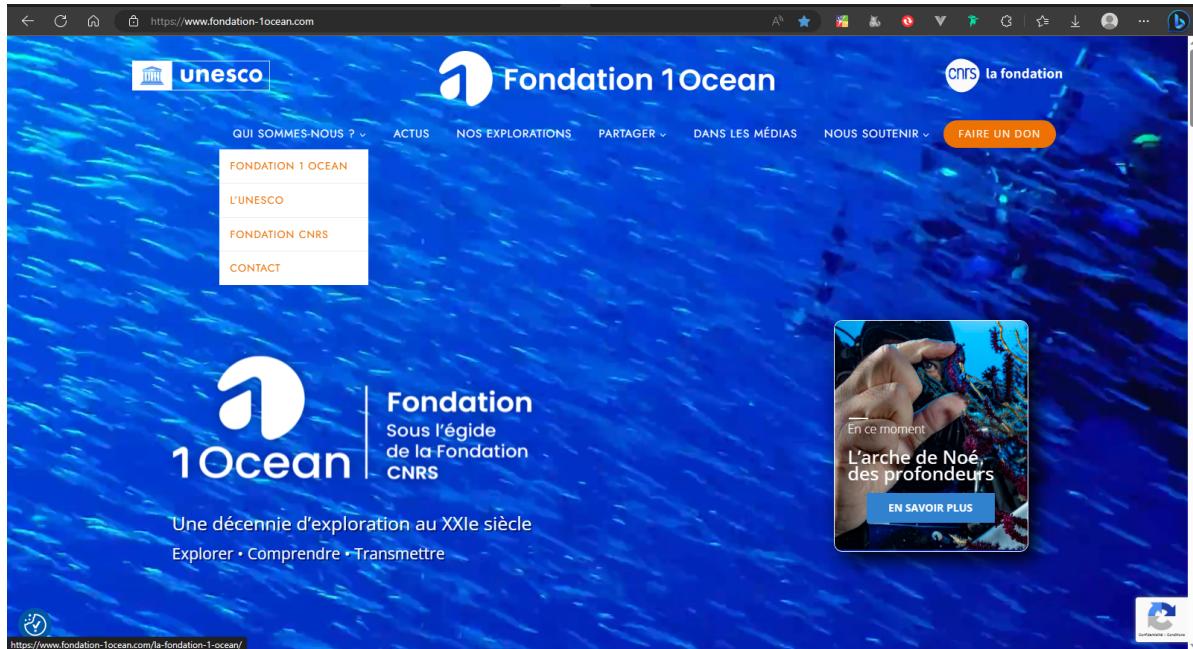
# DOSSIER PROFESSIONNEL (DP)

Actions groupées						Appliquer	Toutes les dates	Toutes	Filtrer	62 éléments	<	<	1 sur 4	>	x
<input type="checkbox"/>	Titre	Auteur/autrice	Catégories	Étiquettes		Date									
<input type="checkbox"/>	Qui est Alexis Rosenfeld — Brouillon	Alexis JEUFFRAIN	Médias	—		Dernière modification 20/12/2023 à 14h00									
<input type="checkbox"/>	tF1 – journal télévisé 20h	Claire MONCHAOUZOU	Médias	L'arche de noe		Publié 09/03/2023 à 14h43									
<input type="checkbox"/>	Le Figaro	Claire MONCHAOUZOU	Médias	L'arche de noe		Publié 09/03/2023 à 11h53									
<input type="checkbox"/>	20 minutes	Claire MONCHAOUZOU	Médias	L'arche de noe		Publié 09/03/2023 à 11h46									
<input type="checkbox"/>	Test colonne — Brouillon	Contact STIT Consulting	Non classé	—		Dernière modification 06/03/2023 à 18h23									
<input type="checkbox"/>	Ouest France – Lily — Elementor	Claire MONCHAOUZOU	Médias	—		Publié 08/02/2023 à 6h12									
<input type="checkbox"/>	La « Vallée aux Mille Roses » récompensée par la revue NATURE — Épinglé. Elementor	Tom ROSENFELD	Actus, Nos news	Vallée 1000 Roses		Publié 14/12/2022 à 22h09									
<input type="checkbox"/>	Antarctique : Une algue rouge capable de piéger le carbone à grande profondeur — Épinglé. Elementor	Tom ROSENFELD	Actus, Les actus de l'océan	—		Publié 04/12/2022 à 14h43									
<input type="checkbox"/>	Rhythms monthly Mag	Tom ROSENFELD	Médias	Volcans		Publié 04/12/2022 à 12h12									
<input type="checkbox"/>	<strong>Uruguay : vers un traité international contre la pollution plastique</strong> — Elementor	Tom ROSENFELD	Actus, Les actus de l'océan	—		Publié 03/12/2022 à 17h40									

- une interface utilisateur:

- Une page d'accueil :
  - un menu de navigation,
  - en premier lieu le projet principal de l'association (en cours),
  - les “news de l'océan”,
  - les principaux projets,
  - les dernières parutions

# DOSSIER PROFESSIONNEL (DP)

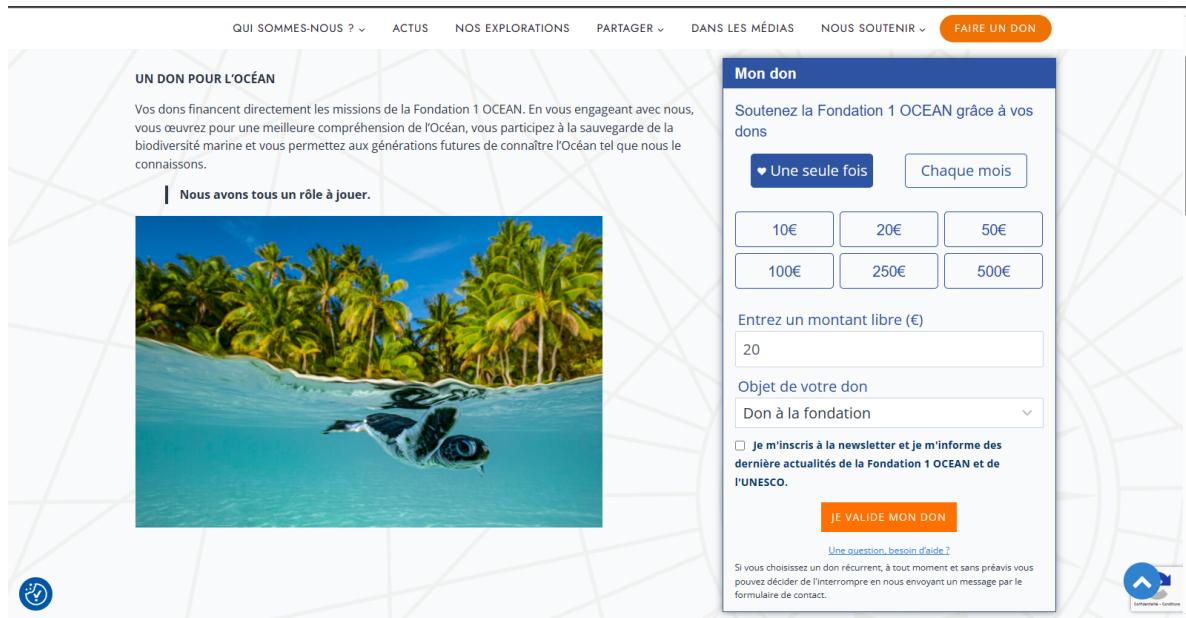


- Une page dédiée à chaque articles, parution ou projet

The screenshot shows a project page titled 'La plus grande migration animale du monde : le sardine run'. It features a large image of a massive school of sardines swimming in the ocean. A blue button labeled 'A VENIR' is at the top right. Below the image, a paragraph describes the Sardine Run migration and its significance. A blue button labeled 'VOIR LA MISSION' is at the bottom. To the right, another section titled 'L'arche de Noé des profondeurs, un avenir pour la biodiversité ?' is partially visible, featuring an image of a hand interacting with marine life.

- une page de don avec possibilité de choisir la cause pour laquelle on souhaite donner

# DOSSIER PROFESSIONNEL (DP)



Pour le backoffice et pour administrer les organes de presse ayant publié les travaux d'Alexis Rosenfeld j'ai donc commencé par me renseigner sur les "*hook wordpress*" pour créer une "*Metabox*" qui contiendrait les organes de presse pour les liés à/aux l'article(s) / parution(s).

```
function add_meta_box_organe_presse(): void
{
    add_meta_box(
        id: 'organe_presse_metabox',
        title: 'Organe de presse',
        callback: 'show_meta_box_organe_presse',
        ['post'],
        context: 'side',
        priority: 'default'
    );
}
add_action( 'add_meta_boxes', 'add_meta_box_organe_presse' );
```

Ensuite pour ajouter la box j'ai créé la fonction *add\_meta\_box\_organe\_presse*, qui appelle la fonction *add\_meta\_box* de wordpress celle-ci comprend:

- l' *Id* qui sera utilisé lors de l'appel pour l'affichage,
- le *Titre* visible par les administrateurs,
- la *fonction callback* nécessaire à l'affichage,
- le *Type de contenu* auxquels la metabox doit être ajouté,
- l'*emplacement* de la box dans le tableau de bord,
- la *priorité* d'affichage,
- et pour finir, la fonction *add\_action* qui est utilisée pour attacher la fonction *add\_meta\_box\_organe\_presse* à l'action *add\_meta\_boxes*.

# DOSSIER PROFESSIONNEL (DP)

Pour l'affichage de la metabox j'ai donc créé la fonction `show_meta_box_organe_presse` qui commence par récupérer une liste des organes de presse, ensuite acquiert les informations liés à l'organe et le envoie dans une liste déroulante disponible à l'administrateur des parutions, j'ajoute enfin un mécanisme de sécurité à l'aide de la fonction `wp_create_nonce` qui permet de prévenir les attaques de falsification de requête inter-site (CSRF : Cross-Site Request Forgery).

```
//////////  
// Afficher la meta box pour sélectionner l'organe de presse/////////  
  
function show_meta_box_organe_presse( $post ): void  
{  
    // Récupérer la liste des organes de presse  
    $organes_de_presse = get_posts( array(  
        'post_type' => 'presse',  
        'numberposts' => -1,  
        'orderby' => 'title',  
        'orden' => 'ASC',  
    ) );  
  
    // Récupérer la valeur sélectionnée (si elle existe)  
    $organe_de_presse_selectionne = get_post_meta( $post->ID, key: '_organe_de_presse', single: true );  
  
    // Afficher la liste déroulante  
    echo '<select name="organe_de_presse">';  
    echo '<option>Aucun</option>';  
    foreach ( $organes_de_presse as $organe_de_presse ) {  
        $selected = '';  
        if ( $organe_de_presse_selectionne == $organe_de_presse->ID ) {  
            $selected = ' selected';  
        }  
        echo '<option value="' . $organe_de_presse->ID . '" ' . $selected . '>' . $organe_de_presse->post_title . '</option>';  
    }  
    echo '</select>';  
  
    // Ajouter un champ caché pour la sauvegarde de la valeur  
    //nonce: hachage composé de chiffres et de lettres utilisé pour protéger les formulaires d'un site contre une utilisation abusive (faille CSRF)  
    echo '<input type="hidden" name="organe_de_presse_meta_box_nonce" value="' . wp_create_nonce( action: 'organe_de_presse_meta_box' ) . '">';  
}  
add_action( 'show_meta_boxes', 'show_meta_box_organe_presse' );
```

Ensuite j'ai créé la fonction de sauvegarde de la valeur et de la liaison à l'article en vérifiant si la requête envoyée est valide, si l'utilisateur a bien les droits d'administration et pour finir met à jour la valeur sélectionnée.

# DOSSIER PROFESSIONNEL (DP)

```
//////////  
// Sauvegarder la valeur sélectionnée pour l'organe de presse/////////  
/////////  
function save_meta_box_organe_presse( $post_id ): void  
{  
    // Vérifier si la requête est valide  
    if ( !isset( $_POST['organe_de_presse_meta_box_nonce'] ) || !wp_verify_nonce( $_POST['organe_de_presse_meta_box_nonce'], action: 'organe_de_presse_meta_box' ) ) {  
        return;  
    }  
  
    // Vérifier si l'utilisateur a la permission de modifier l'article  
    if ( !current_user_can( capability: 'edit_post', $post_id ) ) {  
        return;  
    }  
  
    // Mettre à jour la valeur sélectionnée  
    if ( isset( $_POST['organe_de_presse'] ) ) {  
        update_post_meta( $post_id, meta_key: '_organe_de_presse', absint( $_POST['organe_de_presse'] ) );  
    } else {  
        delete_post_meta( $post_id, meta_key: '_organe_de_presse' );  
    }  
}  
add_action( 'save_post', 'save_meta_box_organe_presse' );
```

Pour terminer j'ai créer la fonctionnalité d'affichage du logo de l'organe de presse relié à l'article s'il y en a un défini

```
//////////  
// add action pour l'organe de presse sélectionné/////////  
/////////  
function boxODP_action()  
{  
    $organe_de_presse = get_post_meta( get_the_ID(), key: '_organe_de_presse', single: true );  
    //add_image_size( 'my_size', 100, 50 );  
  
    // Vérifier si un organe de presse a été sélectionné  
    if ( $organe_de_presse ) {  
        // Récupérer l'image associée à l'organe de presse  
        $image = get_the_post_thumbnail_url( $organe_de_presse, size: 'medium' );  
        // Afficher l'image  
        if ( $image ) {  
            echo '';  
        }  
    }  
}  
add_action('kadence_blocks_post_loop_header', 'boxODP_action',19);
```

## 2. Précisez les moyens utilisés :

# DOSSIER PROFESSIONNEL (DP)

- **WordPress** et certain de ses modules:
  - **Depicter**, pour la gestion des vidéos,
  - **CPT UI**, (CustomPostType UserInterface) pour gérer les types de publications
  - **WP Rocket**, plugin de cache qui permet d'accélérer la vitesse de chargement du site,
- **Figma** Editeur de graphiques vectoriels pour la réalisation de la maquette et la charte graphique),
- **Asana** Gestionnaire de tâche et de coordination d'équipe,
- **WinScp** Logiciel de transfert de fichiers pour la récupération de la partie back de WP,
- **PhpStorm** Éditeur de texte pour le développement des fonctionnalités,
  - JavaScript,
  - Php,
  - Css
- **Docker** pour l'environnement de test

## 3. Avec qui avez-vous travaillé ?

**Philippe Stora** : CEO/CTO Stit consulting,  
**Alexis Rosenfeld** : Président de la Fondation 1ocean,  
**Alexis Jeuffrain** : Graphiste,  
**André Grassi** : Alternant développeur

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ *Stit Consulting*

Chantier, atelier, service ▶ [www.fondation-1coean.com](http://www.fondation-1coean.com)

Période d'exercice ▶ Du : 11/2022 au : 03/2023

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Développer des composants d'interface utilisateur Activité-type 1 en intégrant les recommandation de sécurité

*Exemple n°2 - REMMEDIA Backoffice*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Remmedia est une entreprise de location de numéros surtaxés, c'est l'un des principaux clients de Stit Consulting , l'entreprise de mon alternance ainsi que le projet principal.

Ce projet était déjà en place et fonctionnel à mon arrivée dans l'entreprise et utilisait **FileMaker** qui est un logiciel de gestion de bases de données.

Il s'agit d'un backOffice afin d'administrer les tierce personnes/entreprises ainsi que toutes les informations relative à celles-ci avec dans la fiche client:

Un Menu de navigation latéral (commun à toutes les pages):

→ avec des sous-menus pour accéder aux différentes informations disponibles:

- ◆ Clients:
  - Les Entreprises,
  - Leurs Contacts / les Particuliers,
  - Les Adresses,
  - Les Communications (mail, mobile, téléphone et leur relation).
- ◆ Numéros:
  - La Base de numéro de Remmedia,
  - Les numéros affectés.
- ◆ Comptabilité:
  - Les Consommations,
  - Les Factures,
  - Les Règlements,
  - Les Virements Bancaires,
  - Les Prélèvements Bancaires.
- ◆ GED (Liste des Documents),
- ◆ Outils:
  - LemonWay (qui est un système de paiement sécurisé),
  - FileMaker (l'ancien système de gestion de données (en cours de refonte)).
- ◆ Paramétrages (Disponible uniquement aux Admins):
  - Administrateurs,
  - Catégories,
  - Type ed Catégorie,
  - Tags/Mots clés.

# DOSSIER PROFESSIONNEL (DP)

un Formulaire qui contient:

- les informations de l'entreprise ou du client (raison sociale, nom commercial, SIREN, site Web),
- l'adresse et complément d'adresse,
- le contact principal,
- coordonnées bancaires.

Des widget pour l'affichage des tableaux contenant des listes:

- de numéro en cours d'utilisation
- de consommation (sur les numéros en cours d'utilisation et utilisés),
- de factures,
- de document ( extrait KBIS, photocopie CNI),
- de contacts,
- des communications (téléphone, mobile et mail des contacts)
- d'adresses des contacts.

The screenshot shows the 'Backoffice' interface for 'remmedia'. On the left, a sidebar lists navigation items: Tableau de bord, Clients, Numéros, Compta, GED, Outils, and Paramétrages. The main content area is titled 'Fiche Entreprise' and contains the following sections:

- Entreprise:** Fields include Raison sociale \* (Remmedia.fr), Nom commercial (Remmedia.fr), Code client (1933), Code parent (vide si aucun), SIREN/SIRET, Site web, and Actif \* (Oui). Buttons: Enregistrer.
- Numéros:** A table with columns 'Valeur' and 'Action'. Status: Aucune donnée.
- Catégories:** A table with columns 'Valeur' and 'Action'. Status: Aucune donnée.
- Synchronisation REMMEDIA Filemaker (ancien backoffice):**
  - Synchroniser la fiche: Dernière synchronisation : Aucune date.
  - Synchroniser les numéros: Dernière synchronisation : Aucune date.
  - Synchroniser les consommations: Dernière synchronisation : Aucune date.
  - Synchroniser les factures: Dernière synchronisation : Aucune date.
- Liste des consommations:**

Type	Période	N° actifs	Appels	Minutes	Référence	Total (€)	Marge	Date	Envoyé	Modifié	Actions
FAC	Nov. 2021	0	0	0,00	Ref: 202111-1933	0,00 0,00 0,00	0,00 0,00 0,00	1,00		06/12/2021 17:23:38	
FAC	Oct. 2021	1	1	0,00	Ref: 202110-1933	0,00 0,00 0,00	0,00 -0,01 -1,00	0,00		05/11/2021 08:55:51	
FAC	Sept. 2021	0	0	0,00	Ref: 202109-1933	0,00 0,00 0,00	0,00 0,00 0,00	0,00		04/10/2021 20:55:38	
FAC	Août 2021	0	0	0,00	Ref: 202108-1933	0,00 0,00 0,00	0,00 0,00 0,00	1,00		07/09/2021 09:19:38	
FAC	Juill. 2021	1	6	0,00	Ref: 202107-1933	0,00 0,00 0,00	0,00 -0,02 -1,00	0,00		05/09/2021 10:25:33	
FAC	Juin 2021	2	2	0,02	Ref: 202106-1933	0,00 0,00 0,00	0,00 -0,03 -1,00	0,00		09/08/2021 15:39:50	
FAC	Mai 2021	1	1	0,00	Ref: 202105-1933	0,00 0,00 0,00	0,00 -0,01 -1,00	0,00		04/06/2021	

Pour développer les composants d'accès aux données j'ai tout d'abord créé le fichier des routes qui auront accès à l' "ApiController" :

Chaque route contient le nom qui lui est attaché, le chemin d'accès, la méthode, et la fonction du controller qui sera appelé lors de l'appel de la route

# DOSSIER PROFESSIONNEL (DP)

le Fichier “number\_affectedNumber.yaml”:

```
### Affected numbers ###
v1_number_affectednumber_get_by_uuid:
    path: /number/affectednumber/{uuid}
    methods: GET
    controller: App\V1\Controller\ApiControllerAffectedNumber::getByUuid

v1_number_affectednumbers_listajax:
    path: /number/affectednumbers/ajax/
    controller: App\V1\Controller\ApiControllerAffectedNumber::ajaxList

v1_number_affectednumbers_list:
    path: /number/affectednumbers/
    methods: GET
    controller: App\V1\Controller\ApiControllerAffectedNumber::getList

v1_number_affectednumber_data_addlist:
    path: /number/affectednumber/{uuid}/data/addlist/
    methods: POST
    controller: App\V1\Controller\ApiControllerAffectedNumber::addListData

v1_number_affectednumber_create:
    path: /number/affectednumber/
    methods: POST
    controller: App\V1\Controller\ApiControllerAffectedNumber::create

v1_number_affectednumber_update:
    path: /number/affectednumber/{uuid}
    methods: PUT
    controller: App\V1\Controller\ApiControllerAffectedNumber::updateByUuid

v1_number_affectednumber_uuid_delete:
    path: /number/affectednumber/{uuid}
    methods: DELETE
    controller: App\V1\Controller\ApiControllerAffectedNumber::deleteByUuid
```

Ensuite j'ai créé le fichier “ApiControllerAffectedNumber” qui extend “ApiController” (le controller parent de tous les controllers dont voici la fonctionnalité pour récupérer une liste de numéro:

# DOSSIER PROFESSIONNEL (DP)

```
<?php

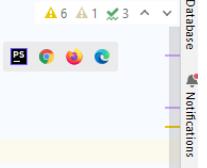
namespace App\V1\Controller;

use ...

class ApiControllerAffectedNumber extends ApiController
{
    protected $entityClassName = AffectedNumber::class;
    protected $managerClassName = AffectedNumberManager::class;
    protected $apicrudObjectName = 'affected_number';
    protected $staticDirectoryEnvVar = null;
    protected $apiReponseListItem = AffectedNumberListResponse::class;
    protected $apiReponseItem = ApiAffectedNumber::class;

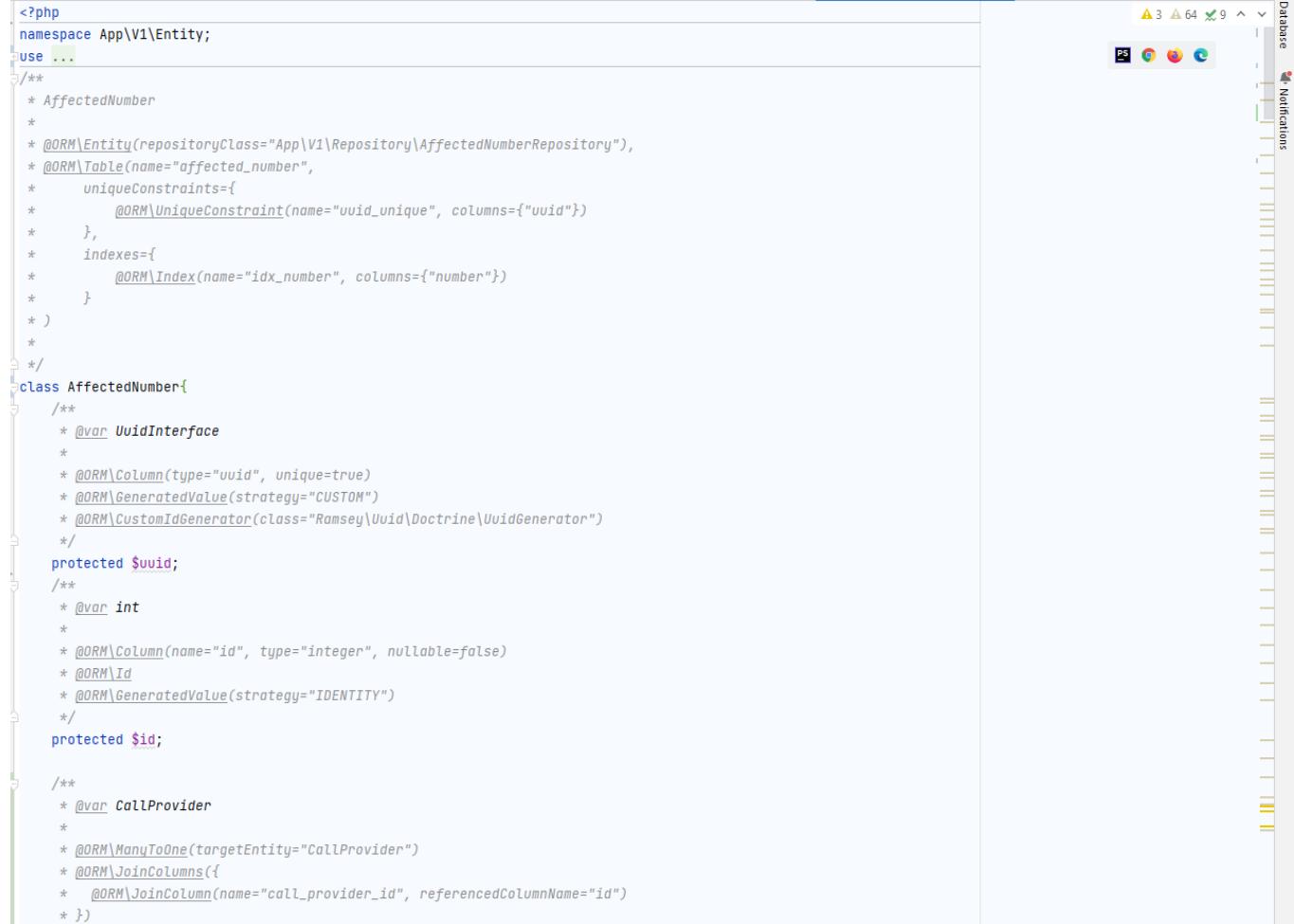
    public function __construct(AffectedNumberManager $manager, RedirectController $redirectController)
    {
        parent::__construct($manager, $redirectController);
    }

    /**
     * @OA\Get(
     *     path="/v1/number/affectednumbers/",
     *     summary="Get list of affected numbers",
     *     description="Returns list of affected numbers full informations",
     *     tags={"Number"},
     *     @OA\Parameter(
     *         name="filter[]", in="query", description="filter", required=false, explode=true,
     *         @OA\Schema( type="array", @OA\Items(type="string") )
     *     ),
     *     @OA\Parameter(
     *         name="basefilter[]", in="query", description="base filter", required=false, explode=true,
     *         @OA\Schema( type="array", @OA\Items(type="string") )
     *     ),
     *     @OA\Parameter(
     *         name="order[]", in="query", description="order", required=false, explode=true,
     *         @OA\Schema( type="array", @OA\Items(type="string") )
     *     ),
     *     @OA\Parameter(
     *         name="size", in="query", description="limit", required=false,
     *         @OA\Schema( type="string" )
     *     ),
     *     @OA\Parameter(
     *         name="mapping", in="query", description="requested fields mapping (json encoded value)", required=false,
     *         @OA\Schema( type="object" )
     *     ),
     *     @OA\Response(response=200, description="successful operation",
     *         @OA\JsonContent(ref="#/components/schemas/AffectedNumberListResponse"),
     *         @OA\XmlContent(ref="#/components/schemas/AffectedNumberListResponse")
     *     ),
     *     @OA\Response(response=400, description="Bad Request"),
     *     @OA\Response(response=401, description="Unauthorized operation"),
     *     @OA\Response(response=404, description="Not found"),
     *     @OA\Response(response="default", description="API descriptor"),
     *     security={"api_key": {}}
     * )
    */
}
```



# DOSSIER PROFESSIONNEL (DP)

et pour finir j'ai créé l'entité ainsi que le model



```
<?php
namespace App\V1\Entity;
use ...

/**
 * AffectedNumber
 *
 * @ORM\Entity(repositoryClass="App\V1\Repository\AffectedNumberRepository"),
 * @ORM\Table(name="affected_number",
 *     uniqueConstraints={
 *         @ORM\UniqueConstraint(name="uuuid_unique", columns={"uuid"})
 *     },
 *     indexes={
 *         @ORM\Index(name="idx_number", columns={"number"})
 *     }
 * )
 */
class AffectedNumber{
    /**
     * @var UuidInterface
     *
     * @ORM\Column(type="uuid", unique=true)
     * @ORM\GeneratedValue(strategy="CUSTOM")
     * @ORM\CustomIdGenerator(class="Ramsey\Uuid\Doctrine\UuidGenerator")
     */
    protected $uuid;

    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    protected $id;

    /**
     * @var CallProvider
     *
     * @ORM\ManyToOne(targetEntity="CallProvider")
     * @ORM\JoinColumns({
     *     @ORM\JoinColumn(name="call_provider_id", referencedColumnName="id")
     * })
     */
}
```

Grâce à la création de ces fichiers, l'utilisation de **Swagger** j'ai récupéré le fichier **openApi.json**. Une fois transmit à Philippe, celui-ci a généré à l'aide de "**SwaggerHub**" les fichiers "**vendor**" contenant toutes les informations nécessaires pour la communication entre l'application et l'api.

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :

Philippe STORA souhaitait changer de technologie et de ce fait faire une refonte de l'existant.

Nous avons donc utilisés les technos suivantes :

- -**PhpStorm** comme éditeur de texte:
  - pour le **back-end**, sur une architecture orientée "service":
    - **PHP** (Symfony) pour le développement des fonctionnalités,
    - **Apache** comme serveur Web,
    - **Mysql** pour la gestion de bases de données relationnelles,
    - **API Rest** comme type d'architecture API
  - pour le **frontend**:
    - **HTML** pour la création des pages webs,
    - **Jquery** qui est une bibliothèque JavaScript,
    - **Bootstrap** qui est un framework Front-end,
    - **Twig** comme base de moteur de templating,
- -**Debian** pour les transferts automatiques de fichiers,
- -**Git** pour le versioning,
- -**Docker** (pour l'environnement de test),  
+quelques dépendances React dont les modules bibliothèques types plugins
- -**Whatsapp** pour les réunions quotidiennes,
- -**GoogleMeet** pour les réunions hebdomadaires,

## 3. Avec qui avez-vous travaillé ?

**Philippe Stora** : CEO/CTO Stit consulting,

**André Grassi** : Alternant développeur

## 4. Contexte

Nom de l'entreprise, organisme ou association ➤ *Stit Consulting*

Chantier, atelier, service ➤ *Remmedia-Backoffice*

Période d'exercice ➤ Du : *10/2022* au : *2023*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Développer la persistance des données en intégrant Activité-type 2 les recommandation de sécurité

Exemple n° 1 - Application Mobile

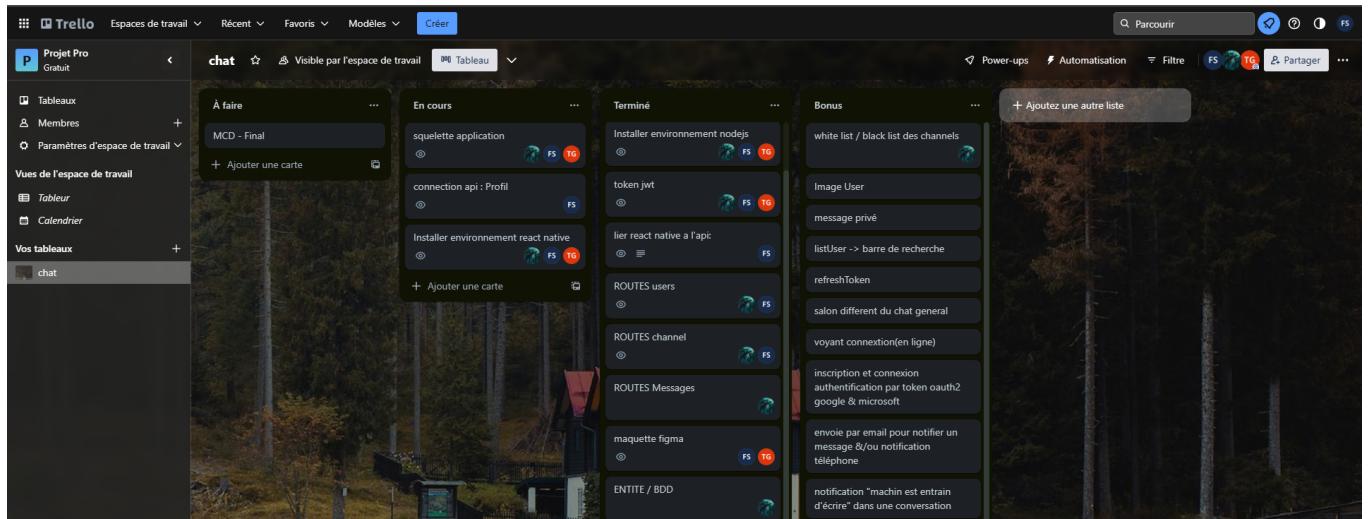
### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour ce projet d'école il nous a été demandé de concevoir, en groupe, une Application Mobile. Plusieurs choix nous étaient proposés, mon groupe, composé de Alex Zicaro, Tony Guillot et moi-même, a choisi de faire une application de discussion.

Nous avons commencés à faire ensemble quelques recherches sur les besoin nécessaire à la création de l'application:

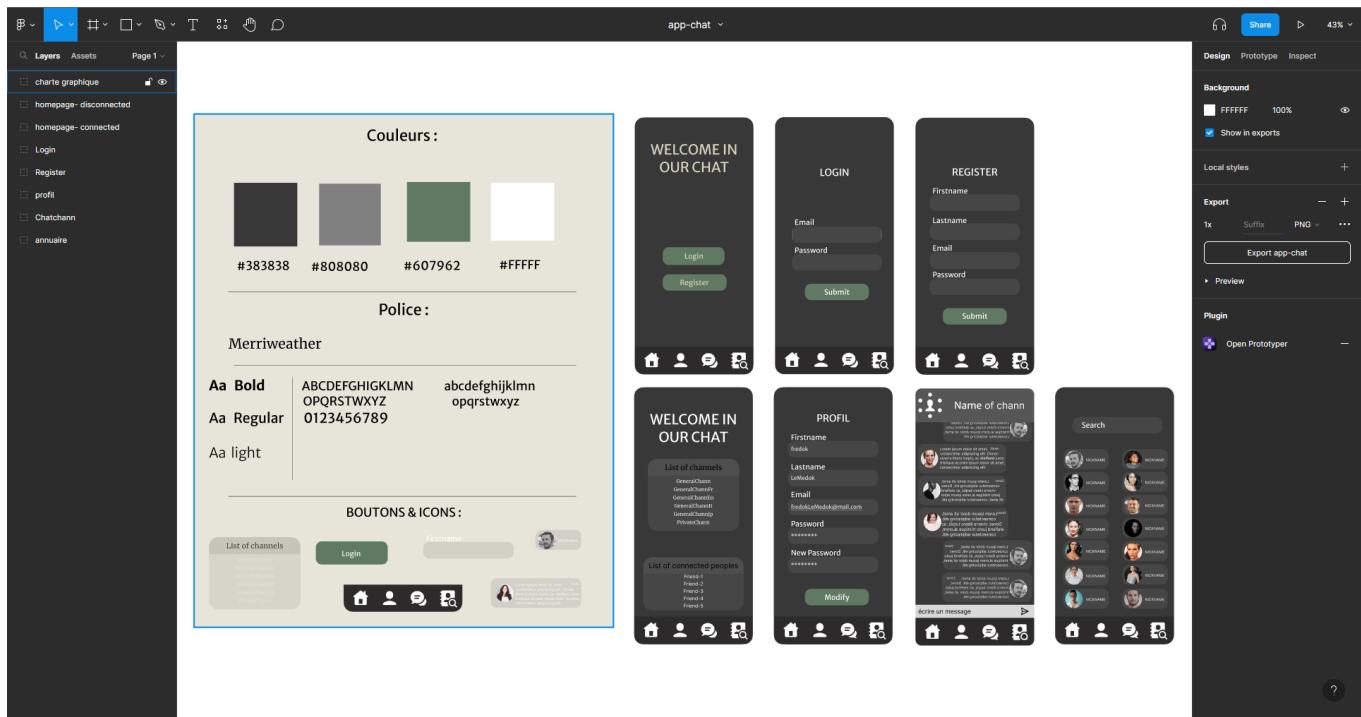
- Une Base de données : pour l'enregistrement des administrateurs, utilisateurs ainsi que les messages et rôles de chacun,
- Une maquette : pour un aperçu d'application finale,
- Quelles technos utiliser? : Notre choix s'est porté vers l'utilisation de **NodeJs** pour la création de notre Api et **React native** qui est un framework open source qui permet de créer des applications mobiles pour IOS et ANDROID.

Ensuite nous avons mis en place un système de gestion de projet et de collaboration permettant d'assigner une tâche à chacun à l'aide de **TRELLO**



# DOSSIER PROFESSIONNEL (DP)

Pour continuer nous nous sommes concertés pour ce que devait contenir notre application, pour ce faire, nous avons réalisé une charte graphique contenant les couleurs, la police d'écriture ainsi que le visuel des principales icônes, boutons. Et, avec l'aide de celle-ci nous avons créés notre maquette

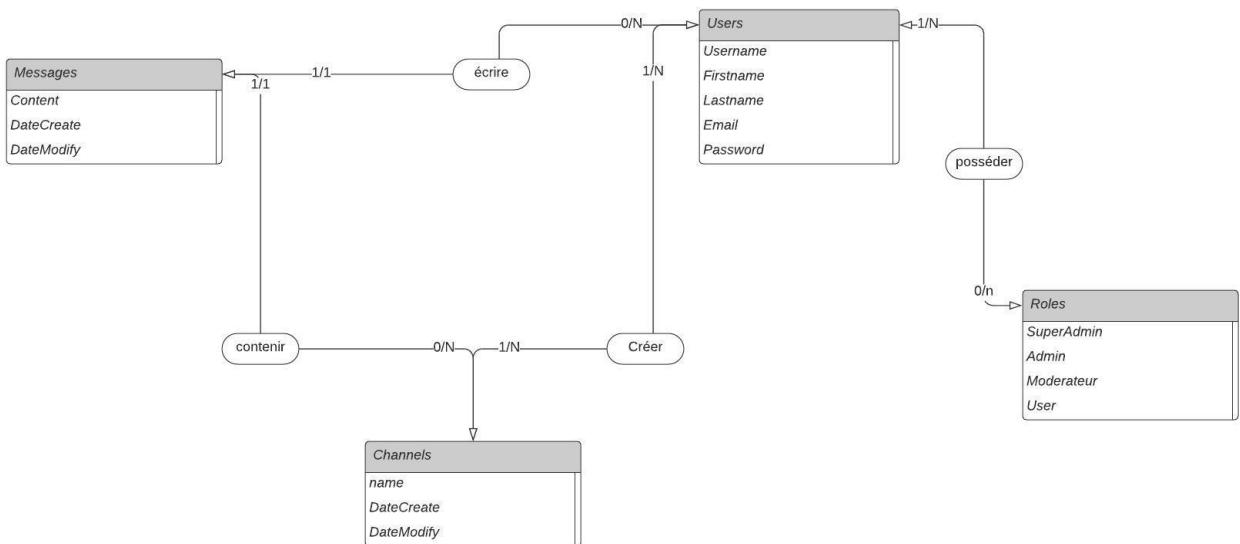


Pour terminer notre travail en commun, nous avons réalisés le Modèle Conceptuel de Données (MCD) et le Modèle Logique de Données (MLD) contenant une table supplémentaire. Elle permet de représenter la relation entre les utilisateurs, les canaux, les rôles et de définir les autorisations et les priviléges accordés à chaque utilisateur dans un canal donné.:

# DOSSIER PROFESSIONNEL (DP)

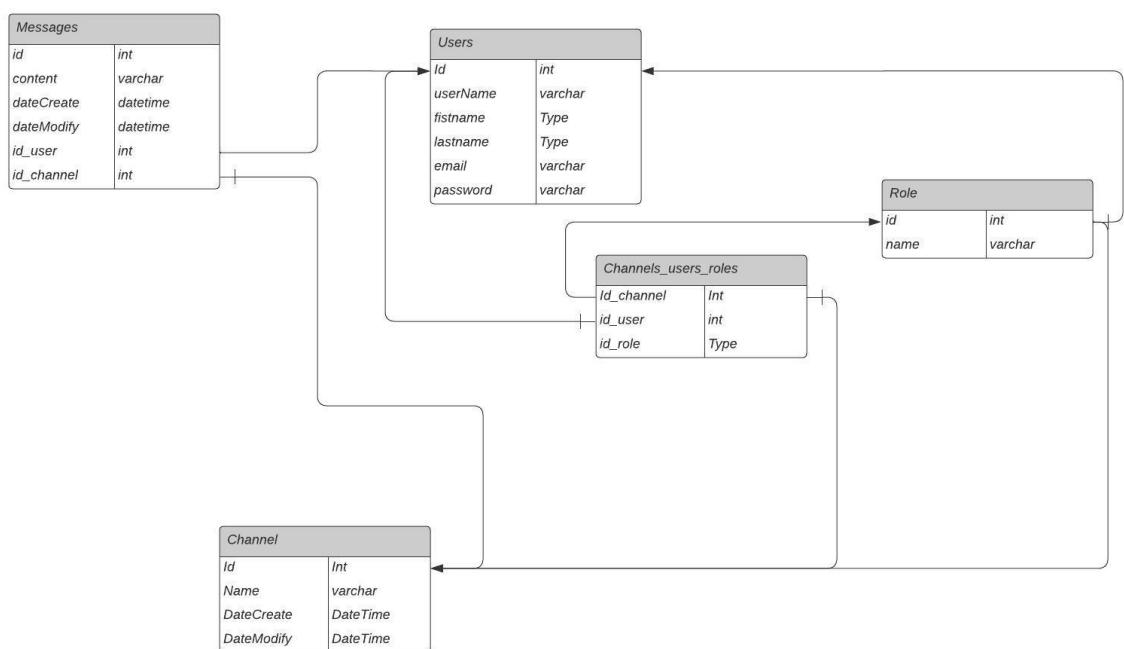
Model Conceptuel de Donnée

Tony Guillot | Alex Zicaro | Frederick Sonder | June 19, 2023



Modèle logique de données

Frederick Sonder | June 19, 2023



# DOSSIER PROFESSIONNEL (DP)

Une fois fait, nous avons commencé nous répartir les tâches : Alex s'est posté sur la création de la BDD, je me suis donc posté sur la création de l'api:

J'ai tout d'abord configuré un serveur Express à l'aide d' Express.js, qui écoute sur le port 3001. Il met à disposition les fichiers statiques à partir du dossier "public" et définit des routes et des middlewares pour gérer les requêtes vers les URL spécifiques liées aux utilisateurs, aux canaux et aux messages.



```
server.js × authentication.js × jwt.js × verifyToken.js ×
1 //nodemon = plugin servant à stopper et relancer le server automatiquement
2 // pour afficher les modification
3 //pour lancer le server avec nodemon: "nodemon index.js" (ou "nom-du-fichier-index.js")
4
5 const express = require('express');
6 const app = express();
7 const server = app.listen( port: 3001); //écoute du port 3001
8 const io = require('socket.io')(server);
9
10 app.use(express.static( root: 'public'));
11 const userRouter = require('./Router/userRouter.js'); //appel du Routeur User
12 const channelRouter = require('./Router/channelRouter.js'); //appel du Routeur Channel
13 const messageRouter = require('./Router/messageRouter.js'); //appel du Routeur message
14
15 app.use(express.json())
16 app.use('/users', userRouter);
17 app.use('/channels', channelRouter);
18 app.use('/messages', messageRouter);
19
```

Ensuite j'ai créé le premier "Router" qui gère différentes routes pour les opérations liées aux utilisateurs. Il inclut des routes pour récupérer tous les utilisateurs, récupérer un utilisateur spécifique, se connecter, s'enregistrer, mettre à jour un utilisateur et supprimer un utilisateur.



```
userController.js × userRouter.js ×
1 let express = require('express'); // appel du module 'express'
2 let router = express.Router(); // appel de la méthode 'router'
3
4 const user = require('../Controllers/userController.js')
5 let users = new user(); // instantiation de l'user
6
7 const verifyToken = require("../middleware/verifyToken");
8
9 router.get( path: "/", verifyToken ,users.listUser) // route pour recuperer tout les utilisateurs
10 router.get( path: "/user/:email", verifyToken ,users.singleUser) // route pour recuperer 1 utilisateur
11 router.post( path: "/login", users.loginUser) // route pour se connecter
12 router.post( path: "/register", users.registerUser) // route pour s'enregistrer
13 router.put( path: "/update/:id", verifyToken , users.updateUser) // route pour mettre à jour 1 utilisateur
14 router.delete( path: "/delete/:id", verifyToken ,users.deleteUser) // route pour supprimer 1 utilisateur
15
16 module.exports = router;
```

# DOSSIER PROFESSIONNEL (DP)

Les routes nécessitent également une vérification de jetons pour des raisons de sécurité à l'aide de **JWT** (Json Web Token).



```
Project: server.js × authentication.js × jwt.js ×
Commit: Pull Requests

1 const jwt = require('jsonwebtoken')
2 const express = require('express')
3 const app = express()
4
5 const dotenv = require('dotenv').config({ path: '../.env' })
6
7 /*extraction du token*/
8 const extract = authorization => {
9   if (!token) {
10     return res.sendStatus(401)
11   }
12 }
13
14 function generateToken(req, res, next) {
15   console.log(req.body);
16   const jwt2 = jwt.sign(req.body, process.env.ACCESS_TOKEN_SECRET, { expiresIn: '1800s' }, (err, res))
17   console.log(res)
18   return jwt2
19   next()
20 }
21 module.exports = generateToken
```



```
verifyToken.js ×
1 const jwt = require('jsonwebtoken');
2 const express = require('express')
3 const app = express()
4 const dotenv = require('dotenv').config({ path: '../.env' })
5
6 function verifyToken(req, res, next) {
7   let token = req.headers['authorization'];
8   token = token && token.split(' ')[1];
9   if (!token) {
10     return res.status(401).send({ error: 'No token provided' });
11   }
12   try {
13     const decoded = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET);
14     if (decoded.exp < Date.now() / 1000) {
15       return res.status(401).send({ error: 'Token expired' });
16     }
17     next();
18   } catch (err) {
19     return res.status(401).send({ error: 'Invalid token' });
20   }
21 }
22 module.exports = verifyToken
```

# DOSSIER PROFESSIONNEL (DP)

Le routeur est configuré pour travailler avec un contrôleur d'utilisateur qui définit une classe User qui contient des méthodes pour effectuer des opérations liées aux utilisateurs. Il utilise un module de connexion à la base de données (DbConnection.js) et un module de cryptage de mot de passe (bcrypt). La classe contient des méthodes pour récupérer la liste des utilisateurs, récupérer les informations d'un utilisateur spécifique, mettre à jour les informations d'un utilisateur, supprimer un utilisateur, se connecter et s'enregistrer. Les méthodes utilisent des requêtes SQL pour interagir avec la base de données.



```
userController.js × userRouter.js ×
1 const db = require('../DbConnection/DbConnection.js');           // require de la connection a l db
2 const bcrypt = require('bcrypt');                                     // require de la méthode de cryptage
3 const e = require("express");
4 const generateToken = require("../middleware/jwt");
5 const jwt = require("jsonwebtoken");
6
7
8 const hashage = (password) => {                                       // définition de quel param on hash
9   return bcrypt.hashSync(password, salt: 10)
10 }
11
12 class User {
13 }
```

Par exemple, la fonction registerUser est une méthode qui permet d'enregistrer un utilisateur dans la base de données:

Elle commence par récupérer les données fournies dans la requête, telles que l'e-mail, le prénom, le nom et le mot de passe, effectue des vérifications pour s'assurer de la validité des données fournies, vérifie que le mot de passe respecte les critères requis (longueur minimale, présence de majuscules, minuscules et chiffres) et que l'adresse e-mail est valide.

Ensuite, elle vérifie si l'adresse e-mail est déjà utilisée par un autre compte utilisateur. Si c'est le cas, elle renvoie une erreur indiquant qu'un compte est déjà lié à cet e-mail.

Si toutes les vérifications sont réussies, elle procède à l'enregistrement de l'utilisateur dans la base de données en utilisant la fonction de hachage pour sécuriser le mot de passe.

Après l'enregistrement, elle effectue une autre requête pour récupérer l'ID de l'utilisateur nouvellement créé.

Enfin, elle ajoute une entrée dans une table de liaison pour associer cet utilisateur au canal général avec un rôle spécifié, puis renvoie une réponse indiquant que l'utilisateur a été enregistré avec succès.

En résumé, la fonction registerUser vérifie la validité des données, évite les duplications d'adresse e-mail, sécurise le mot de passe, enregistre l'utilisateur dans la base de données et associe l'utilisateur à des rôles et canaux spécifiques

# DOSSIER PROFESSIONNEL (DP)

```
userController.js
151 async registerUser(req, res) {
152   let body2 = JSON.parse(req.body.data)
153   let password = hashage(body2.password);
154   let { email, first_name, last_name } = body2;
155
156   if (!email || !password || !first_name || !last_name) {
157     return res.status(400).json({ message: 'Erreur, l\'utilisateur n\'a pu être enregistré' })
158   } else {
159     const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,$)/; //minimum 8char, 1maj, 1minuscule et 1 chiffre
160     const emailRegex = /^(([^<>(){}\\.,;:\\s@"]+(\\\.[^<>(){}\\.,;:\\s@"]+)*|(\".+\"))@(([[0-9]{1,3}\\.][0-9]{1,3}\\.][0-9]{1,3}\\.][0-9]{1,3})|(([a-zA-Z-0-9]+\\.)+[a-zA-Z]{2,}))$/;
161
162     if (!passwordRegex.test(body2.password)) {
163       console.log('UC.js:regex/1')
164
165       return res.status(401).json({ message: 'Erreur, le mot de passe doit contenir au minimum 8 caractères, 1 majuscule et 1 minuscule et 1 chiffre' })
166     }
167
168     if (!emailRegex.test(body2.email)) {
169       return res.status(402).json({ message: 'Erreur, email invalide' })
170     } else {
171       const sql2 = "SELECT * FROM users WHERE email LIKE '%" + body2.email + "%'";
172       let dbemail = db.query(sql2, (ress, data) => {
173
174         if (data.length > 0) { // Vérifie la longueur du résultat et si = de 0 utilisateur : erreur
175
176           return res.status(403).json({ message: 'Erreur, un compte est déjà lié à cet email' })
177         } else {
178           db.query("INSERT INTO users (first_name, last_name, email, password) VALUES ('" + first_name + "','" + last_name + "','" + email + "','" + password + "')");
179           let sql = 'SELECT id FROM users ORDER BY id DESC LIMIT 1'; //récupère le dernier utilisateur enregistré
180           db.query(sql, [], (err, data) => {
181             if (err) {
182               throw err;
183             }
184             let sql2 = 'INSERT INTO channels_users_roles (id_channels, id_users, id_roles) VALUES (1,' + data[0].id + ',2)' // j'ajoute dans la table de liaison le dernier utilisateur
185             db.query(sql2, [], (err, data) => {
186               if (err) {
187                 throw err;
188               }
189             }
190           }
191           return res.status(200).json({ message: 'l\'utilisateur a bien été enregistré' })
192         }
193       }
194     }
195   }
196 }
197 }
198 module.exports = User;
```

```
userRouter.js
199
200 router.post('/users', controller.registerUser);
```

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :

- **PhpStorm** comme éditeur de texte:
  - **NodeJs** pour la création de notre Api,
  - **React native** qui est un framework open source qui permet de créer des applications mobiles pour IOS et ANDROID.
  - **Apache** comme serveur Web,
  - **Mysql** pour la gestion de bases de données relationnelles,
  - **API Rest** comme type d'architecture API
- **Figma** pour la création de la charte graphique et la maquette,
- **LucidChart** pour la réalisation des diagrammes
- **Git** pour le versioning,
- **TRELLO** comme système de gestion de projet et de collaboration

## 3. Avec qui avez-vous travaillé ?

Alex ZICARO Tony Guillot

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ **Projet LaPlateforme**

Chantier, atelier, service ▶

Période d'exercice ▶ Du : **01/2023** au : **02/2023**

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Développer une application multicouche en intégrant Activité-type 3 les recommandation de sécurité

*Exemple n° 1 - Remmedia - Backoffice*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Toujours dans le projet pour l'entreprise **Remmedia**, Pour démontrer l'activité-type 3, je m'appuierai sur l'administration des Contacts:

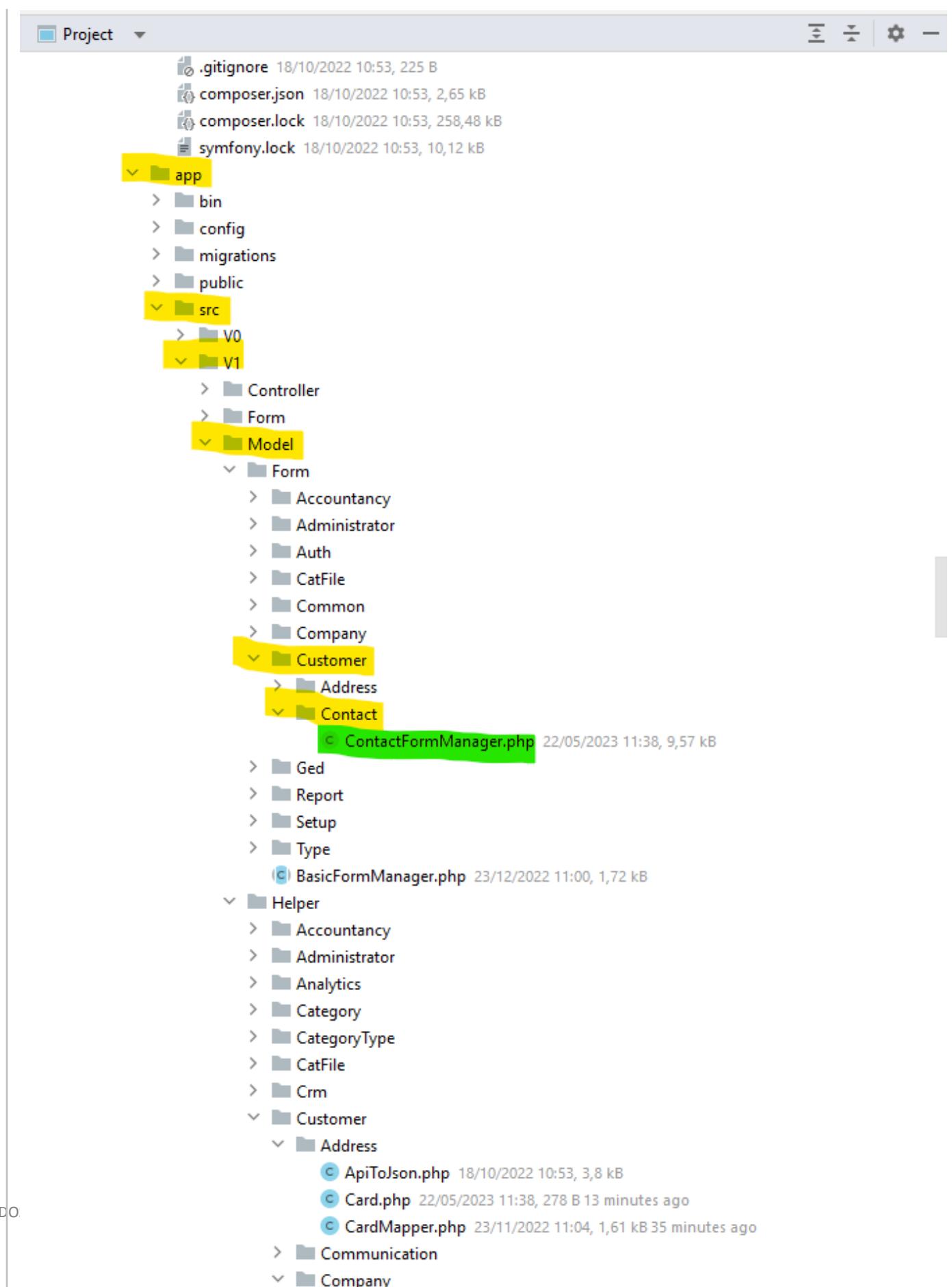
Les clients de Remmedia peuvent être des entreprises et de ce fait disposer de plusieurs contact, et ces contacts ont des informations à enregistrer :

- Nom,
- Prénom,
- Date d'anniversaire,
- Lieux de naissance,
- Nationalité
- adresse (qui contient plusieurs informations):
  - Adresse,
  - Code postal,
  - Ville,
  - Pays
- Communication (qui comprend aussi plusieurs informations):
  - E-mail,
  - Téléphone,
  - Téléphone mobile

Comme le projet est déjà avancé, certaines parties ont déjà été développées dont une partie spécifique de l'Api, Philippe m'a alors demandé de développer la partie application :

J'ai donc commencé par chercher dans l'arborescence le fichier correspondant au formulaire,

# DOSSIER PROFESSIONNEL (DP)



# DOSSIER PROFESSIONNEL (DP)

Ce fichier contient en particulier la fonction “getRenderData” qui comprend la variable “\$listFieldData” qui elle-même englobe les champs correspondants au formulaire et leur paramètres.

```
/*
 * @param $companyList
 * @param $partnerList
 * @return array
 */
public function getFieldData($object = null, $mode = self::MODE_VIEW)
{
    $listFieldData = [
        [
            'fieldname' => 'name',
            'type' => TextType::class,
            'attributes' => [
                'label' => 'Nom *',
                'attr' => ['placeholder' => 'Nom *'],
                'row_attr' => ['class' => 'form-label-group'],
                'required' => true
            ]
        ],
        [
            'fieldname' => 'firstname',
            'type' => TextType::class,
            'attributes' => [
                'label' => 'Prénom *',
                'attr' => ['placeholder' => 'Prénom *'],
                'row_attr' => ['class' => 'form-label-group'],
                'required' => true
            ]
        ],
        [
            'fieldname' => 'birthdate',
            'type' => BirthdayType::class,
            'attributes' => [
                'label' => 'Né(e) le',
                'attr' => ['placeholder' => 'Né(e) le'],
                'row_attr' => ['class' => 'input-group'],
                'required' => false
            ]
        ],
        [
            'fieldname' => 'birthplace',
            'type' => TextType::class,
            'attributes' => [
                'label' => 'Ville de naissance',
                'attr' => ['placeholder' => 'Ville de naissance'],
                'row_attr' => ['class' => 'form-label-group'],
                'required' => false
            ]
        ],
        [
            'fieldname' => 'nationality',
            'type' => CustomChoiceType::class,
            'attributes' => [
                'label' => 'Nationalité',
                'attr' => ['class' => 'form-control'],
                'row_attr' => ['class' => 'input-group'],
                'required' => false,
                'choices' => array_flip($this->listCountrySelect)
            ]
        ],
        [
            'fieldname' => 'state',
            'type' => CustomChoiceType::class,
            'attributes' => [
                'label' => 'Actif *',
                'attr' => ['class' => 'form-control'],
                'row_attr' => ['class' => 'input-group'],
                'required' => true,
                'choices' => [
                    '' => '',
                    'Oui' => '1',
                    'Non' => '0'
                ]
            ]
        ]
    ];
}
```

par exemple pour le champ du nom du contact :

```
"fieldname" => "name"
"type" => TextType::class
"attributes" => [
    "label" => "Nom"
    "attr" => ["placeholder"=> nom]
    "row_attr"=> ["class"=>
    "required" => true
]
```

sera le nom du champ,  
corresponds au type, dans ce cas de type texte,  
contiendra tous les attributs  
sera le label visible par l'utilisateur  
attribut spécifiant le texte de substitution  
ajoute une classe CSS à la ligne du formulaire  
indique que le champ est requis pour valider le formulaire

# DOSSIER PROFESSIONNEL (DP)

Grâce à celui-ci j'ai pu faire la "Card" qui va contenir les donnée du formulaire:

```
<?php  
  
namespace App\V1\Model\Helper\Customer>Contact;  
  
class Card  
{  
    // Main informations  
    public $uuid = '';  
    public $record_type = 0;  
    public $firstname = null;  
    public $name = null;  
    public $nationality = null;  
    public $birthplace = null;  
    public $birthdate = null;  
    public $state = true;  
    public $date_creation = null;  
    public $date_modify = null;  
    // Main address  
    public $address_uuid = '';  
    public $address_address = '';  
    public $address_zip = '';  
    public $address_town = '';  
    public $address_country = '';  
    // Main communication  
    public $communication_email = '';  
    public $communication_phone = '';  
    public $communication_mobile = '';  
}
```

Ainsi que le "cardMapper" qui selon la méthode appelée fera appel à l'api pour envoyer ou récupérer les informations fournies:

Par exemple pour récupérer les informations stockées, en m'imprégnant d'autres "mapper", j'ai créé la fonction "mapFromApi" qui prend en paramètre le Modèle de l'api correspondant, dans notre cas "ApiModelThirdparty" (intégré dans les états utilisés:

"use Stit\Remmedia\Backoffice\Api\Base\Model\Thirdparty as ApiModelThirdparty").

dans cette fonction j'ai ensuite créé un objet qui se basera sur la "Card" susmentionnée et pour chaque champ, selon le modèle donné récupère l'information correspondante

- pour les informations principales du contact j'ai donc utilisé le modèle Thirdparty :

```
    public function mapFromApi(ApiModelThirdparty $apiModel){  
        $cardObject = new Card();  
        $cardObject->uuid = $apiModel->getUuid();  
        $cardObject->record_type = $apiModel->getRecordType();  
        $cardObject->firstname = $apiModel->getFirstname();  
        $cardObject->name = $apiModel->getName();  
        $cardObject->state = $apiModel->getState();  
        $cardObject->date_creation = $apiModel->getDateCreation();  
        $cardObject->date_modify = $apiModel->getDateModify();
```

# DOSSIER PROFESSIONNEL (DP)

- Pour les informations relatives à l'adresse j'ai utilisé un autre modèle, celui des adresses. Tout d'abord on définit un objet à "null"
- Ensuite l'appel à l'api est fait et si le résultat n'est pas vide, vérifie pour chaque adresse s'il y en a une principale (siège social, Adresse personnelle, etc).
- Ensuite si ce même résultat obtenu n'est pas nul on intègre à l'objet "cardObject" les valeurs récupérées dans l'objet "mainAddress"

```
// Main address
$mainAddress = null;
$relatedAddresses = $apiModel->getRelatedAddresses();
if (!empty($relatedAddresses)){
    foreach($relatedAddresses as $relatedAddress){
        $targetAddress = $relatedAddress->getTarget();
        if ($targetAddress == null){
            continue;
        }
        $mainAddress = ($mainAddress==null)?$targetAddress:$mainAddress;
        $mainAddress = (($mainAddress==null || $mainAddress->getMainAddress()==false) && $targetAddress->getMainAddress()==true)?$targetAddress:$mainAddress;
    }
}
if ($mainAddress != null){
    $cardObject->address_uuid = $mainAddress->getUuid();
    $cardObject->address_address = $mainAddress->getAddress();
    $cardObject->address_zip = $mainAddress->getZip();
    $cardObject->address_town = $mainAddress->getTown();
    $cardObject->address_country = $mainAddress->getCountry();
}
```

- On reprend la même méthodologie pour les communications,
- Et on vérifie si le contact est un contact principal de l'entreprise
- Enfin on retourne l'objet qui sera utilisé pour l'affichage

```
// Main communication
$cardObject->communication_phone = $this->getMainCommunication($apiModel, recordType: self::RECORDTYPE_PHONE);
$cardObject->communication_mobile = $this->getMainCommunication($apiModel, recordType: self::RECORDTYPE_PHONEMOBILE);
$cardObject->communication_email = $this->getMainCommunication($apiModel, recordType: self::RECORDTYPE_EMAIL);
// Main contact
$mainContact = null;
$relatedThirdpartiesTarget = $apiModel->getRelatedThirdpartiesSource();
if (!empty($relatedThirdpartiesTarget)){
    foreach($relatedThirdpartiesTarget as $relatedThirdparty){
        $targetThirdparty = $relatedThirdparty->getTarget();
        if ($targetThirdparty == null || $targetThirdparty->getRecordType()!=0){
            continue;
        }
        $mainContact = ($mainContact==null)?$relatedThirdparty:$mainContact;
    }
}
return $cardObject;
```

Pour l'affichage

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :

- -**PhpStorm** comme éditeur de texte:
  - pour le **back-end**, sur une architecture orientée "service":
    - **PHP** (Symfony) pour le développement des fonctionnalités,
    - **Apache** comme serveur Web,
    - **Mysql** pour la gestion de bases de données relationnelles,
    - **API Rest** comme type d'architecture API
  - pour le **frontend**:
    - **HTML** pour la création des pages webs,
    - **Jquery** qui est une bibliothèque JavaScript,
    - **Bootstrap** qui est un framework Front-end,
    - **Twig** comme base de moteur de templating,
- -**Debian** pour les transferts automatiques de fichiers,
- -**Git** pour le versioning,
- -**Docker** (pour l'environnement de test),  
+quelques dépendances React dont les modules bibliothèques types plugins

## 3. Avec qui avez-vous travaillé ?

**Philippe Stora** : CEO/CTO Stit consulting,

**André Grassi** : Alternant développeur

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ **Stit Consulting**

Chantier, atelier, service ▶ **Remmedia - Backoffice**

Période Période d'exercice ▶ Du : **10/2022** au : **2023**

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

# **DOSSIER PROFESSIONNEL (DP)**

## **Déclaration sur l'honneur**

Je soussigné(e) SONDER Frederick ,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur(e) des réalisations jointes.

Fait à Entressen le 17/06/2023

pour faire valoir ce que de droit.

Signature :

# DOSSIER PROFESSIONNEL (DP)

## Documents illustrant la pratique professionnelle

(facultatif)

# Intitulé

Cliquez ici pour taper du texte.

# **DOSSIER PROFESSIONNEL** <sup>(DP)</sup>

## **ANNEXES**

*(Si le RC le prévoit)*