



**remmedia**  
L'Opérateur de la Relation Client

[www.remmedia.fr](http://www.remmedia.fr)

# SOMMAIRE

<b>INTRODUCTION</b>	<b>3</b>
Présentation Personnelle	3
Résumé du projet	3
Compétences Couvertes Par Le Projet	4
<b>ORGANISATION ET CAHIER DES CHARGES</b>	<b>5</b>
Analyse De L'existant	5
Les Utilisateurs Du Projet	6
Les Fonctionnalités Attendues	6
<b>CONCEPTION DU PROJET</b>	<b>7</b>
Choix De Développement	7
Architecture Logicielle	8
Fonctionnement de l'API	8
<b>CONCEPTION DU FRONT-END DE L'APPLICATION</b>	<b>10</b>
Arborescence Du Projet	10
<b>CONCEPTION BACK-END DE L'APPLICATION</b>	<b>11</b>
Arborescence	11
La base de données	12
<b>RÉALISATION</b>	<b>13</b>
Controller & Model	13
Company	13
Ged/FileDocument	14
Composant d'accès aux données	15
Affichage & Rendu	18
<b>ANNEXE</b>	<b>20</b>

# INTRODUCTION

## Présentation Personnelle

Je m'appelle Frédérick SONDER , j'ai trente-six ans. Il y a encore trois ans j'étais magasinier avec plus aucune envie de ce métier. Ma sœur, ayant une entreprise de prothésiste ongulaire, je me suis amusé à lui faire un site Internet (landing page) c'est à ce moment là que j'ai découvert le code informatique. J'ai donc choisi de faire une reconversion professionnelle, J'ai ainsi suivi le cursus de la **Coding School de la plateforme** en 2021. Aujourd' hui je suis en **Bachelor Web CDA**, dans le même établissement afin de préparer mon titre de concepteur / développeur d'application web et en alternance au sein de l'entreprise **STIT Consulting**.

## Résumé du projet

Remmedia is a rental company for telephone numbers, toll-free numbers and premium rate numbers among others.

Remmedia is one of the main clients of Stit Consulting the company of my work-study which makes this project, my main project.

The back office was already in place, it was functional when I arrived at Stit Consulting, it used FileMaker software.

FileMaker is database management software.

It is a backOffice that allows to administer third parties or companies and manages all the information relating to them such as addresses, contacts, consumption data, invoices, payments etc ...

This backoffice also allows to manage the assigned numbers and the available numbers.

Philippe STORA, CEO / CTO of STIT Consulting is my tutor during my work-study program.

Philippe Stora wanted to overhaul the existing back office.

He wanted to stop using FileMaker but to run the back office using a REST API.

So he asked me to act mainly on the Customer part.

## **Compétences Couvertes Par Le Projet**

Ce projet couvre les compétences du titre suivantes :

- Développer des composants d'accès aux données
- Développer la partie front-end d'une interface utilisateur web
- Développer la partie back-end d'une interface utilisateur web
- Mettre en place une base de données
- Développer des composants dans le langage d'une base de données
- Développer des composants métier
- Construire une application organisée en couches

# ORGANISATION ET CAHIER DES CHARGES

## Analyse de l'existant

**Remmedia** est une entreprise de location de numéros, 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 tierces personnes ou entreprises ainsi que toutes les informations relative à celles-ci (adresses, contacts, consommations, factures, règlements), de gérer les numéros attribués et disponibles ainsi qu'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 systeme 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.)

Un système de filtre est aussi disponible afin de pouvoir faire un tri dans les listes affichées.

## Les utilisateurs du projet

Cette application web sera utilisée par un seul type d'utilisateur : les dirigeants ainsi que les salariés de l'entreprise **Remmedia** qui en ont l'autorisation.

## Les fonctionnalités attendues

**Philippe STORA**, CEO / CTO de **STIT Consulting**, et mon tuteur pendant mon alternance, souhaitait faire une refonte de l'existant (ne plus utiliser FileMaker mais à l'aide d'une API REST) et m'a donc demandé d'agir principalement sur la partie Client:

- Sur la fiche client:
  - L'activation du formulaire d'enregistrement des informations bancaires,
  - La création de l'enregistrement des contact ainsi que leurs informations:
    - adresses,
    - communications,
    - relation entre entreprise et contact
  - Pour la liste des document fournis par l'entreprise, une possibilité d'afficher le document (qu'il soit un fichier pdf ou un fichier image),
  - La création de l'enregistrement et la récupération des numéros affectés

# CONCEPTION DU PROJET

## Choix De Développement

### Les langages:

**PHP** : Utilisé pour le développement du back-end de l'application.

**HTML** : Utilisé pour la création des pages web du front-end.

**JavaScript** : Utilisé pour la manipulation du DOM et l'interaction avec les éléments de la page web.

### Les frameworks :

**Symfony** : Un framework PHP utilisé pour le développement des fonctionnalités du back-end.

**Bootstrap** : Un framework front-end qui facilite le développement d'une interface utilisateur responsive et attrayante.

### Les logiciels:

**PhpStorm** : Un éditeur de texte populaire utilisé pour le développement PHP.

**Apache** : Un serveur web couramment utilisé pour héberger des applications PHP.

**MySQL** : Un système de gestion de base de données relationnelles utilisé pour la gestion des données.

### Les outils:

#### de développement:

**Git**: Un système de contrôle de version utilisé pour suivre les modifications du code source

**Docker**: Une plateforme de conteneurisation utilisée pour créer et gérer des environnements de développement isolés

#### de communication et d'échange de fichiers:

**Debian** : Un système d'exploitation utilisé pour les transferts automatiques de fichiers

#### de génération de vues dynamiques:

**Twig**: Un moteur de templating utilisé avec Symfony pour générer des vues HTML dynamiques.

## Architecture Logicielle

Dans cette application web nous utilisons une architecture logicielle API REST (Representational State Transfer) qui est un style d'architecture utilisé pour créer des systèmes logiciels distribués, tels que des services web. Elle repose sur des principes et des contraintes qui facilitent la communication entre les applications de manière cohérente et efficace. Voici les points clés :

**Ressources :** Les ressources sont les principales entités exposées par le service, identifiées par des URL.

**Verbes HTTP :** Les interactions avec les ressources se font à l'aide des verbes standard tels que GET, POST, PUT et DELETE.

**Représentation des ressources :** Les données sont structurées dans un format standard comme JSON ou XML pour être transférées entre les applications.

**Stateless :** Chaque requête contient toutes les informations nécessaires, et aucune session n'est conservée côté serveur.

**Hyperliens :** Les liens hypertexte sont utilisés pour définir les relations entre les ressources et permettre la navigation et la découverte dynamique.

**API versionnée :** Les API REST sont souvent versionnées pour assurer la stabilité et la compatibilité lors des évolutions.

En utilisant ces principes, l'architecture logicielle API REST permet de créer des services web flexibles, évolutifs et interopérables, facilitant l'intégration et la communication entre différentes applications via le protocole HTTP.

## Fonctionnement de l'API

Lorsque le client envoie une requête sur l'API, le routeur analyse l'URL, en fonction de la route et de la méthode un controller est appelé. Ce contrôleur va faire appel à un service qui va communiquer avec le modèle afin de récupérer des données. Ensuite, ses données sont analysées par le service puis une réponse est envoyée au format JSON avec un code statut.



Les principaux statuts utilisés dans ce projets sont :

- 200 : OK

Indique que la requête a réussi

- 400 : BAD REQUEST

Indique que le serveur ne peut pas comprendre la requête à cause d'une mauvaise syntaxe

- 401 : UNAUTHORIZED

Indique que la requête n'a pas été effectuée car il manque des informations d'authentification

- 404 : NOT FOUND

Indique que le serveur n'a pas trouvé la ressource demandée

- 405 : METHOD NOT ALLOWED

Indique que la requête est connue du serveur mais n'est pas prise en charge pour la ressource cible

- 500 : INTERNAL SERVER ERROR

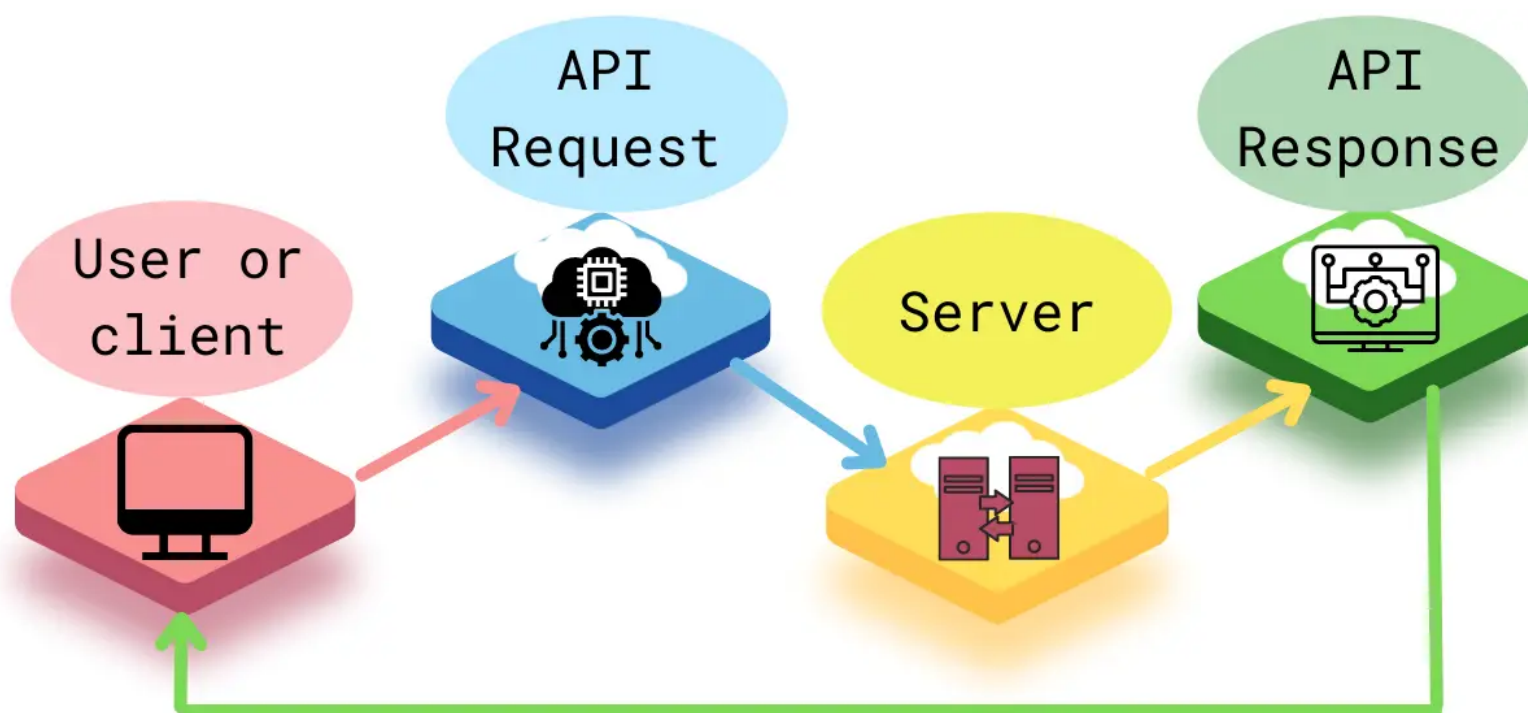
Indique que le serveur a rencontré un problème.

Les différentes méthodes HTTP utilisées dans ce projet :

- GET - Pour la récupération de données

- POST - Pour l'enregistrement de données, la mise à jour partielle ou intégrale des informations d'une donnée

- DELETE - Pour supprimer une donnée



# CONCEPTION DU FRONT-END DE L'APPLICATION

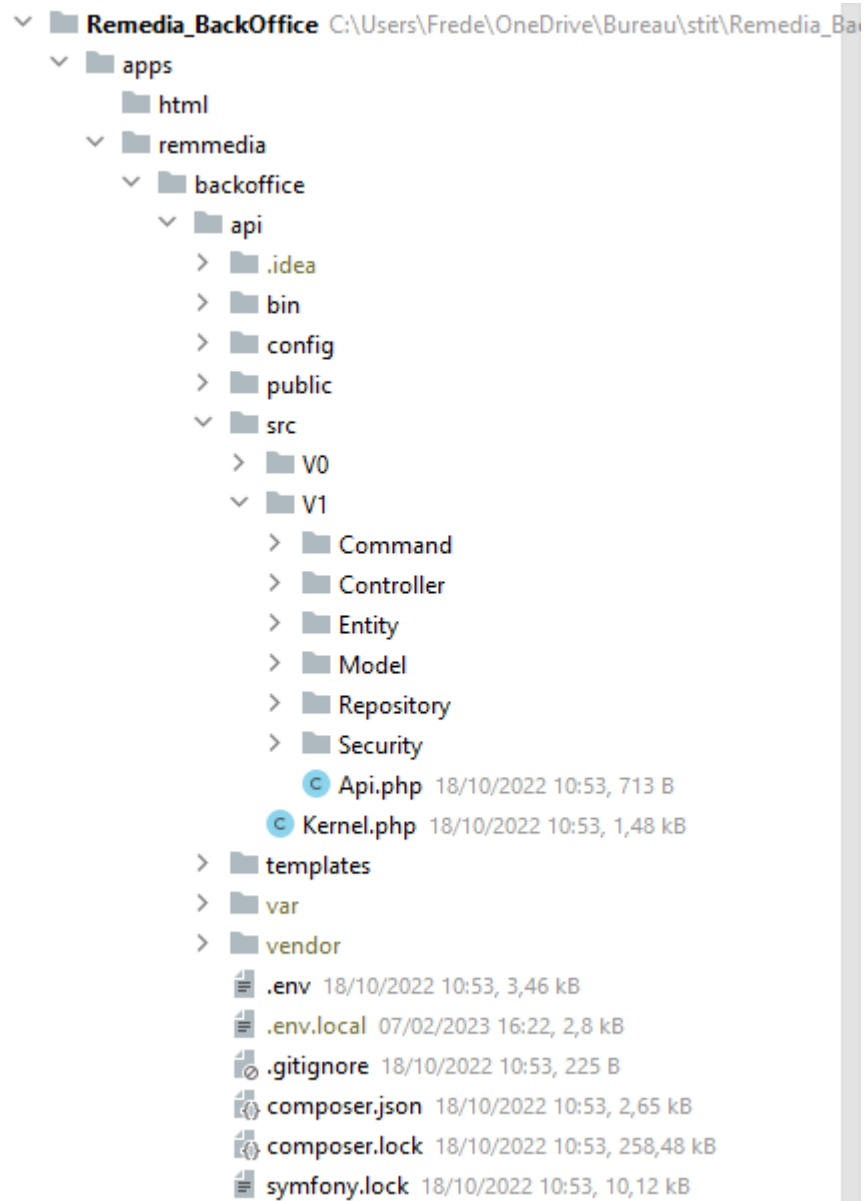
## Arborescence Du Projet

```

└─ app
  └─ bin
  └─ config
  └─ migrations
  └─ public
  └─ src
    └─ V0
    └─ V1
      └─ Controller
      └─ Form
      └─ Model
      └─ Security
      └─ Twig
      └─ Kernel.php 18/10/2022 10:53, 1,49 kB
  └─ templates
  └─ tests
  └─ translations
  └─ var
  └─ vendor
  └─ .env 18/10/2022 10:53, 1,56 kB
  └─ .env.local 18/10/2022 16:10, 238 B
  └─ .env.test 18/10/2022 10:53, 167 B
  └─ .gitignore 18/10/2022 10:53, 322 B
  └─ composer.json 18/10/2022 16:36, 3,3 kB
  └─ composer.lock 18/10/2022 16:35, 317,4 kB
  └─ phpunit.xml.dist 18/10/2022 10:53, 1,1 kB
  └─ symfony.lock 18/10/2022 10:53, 14,22 kB
```

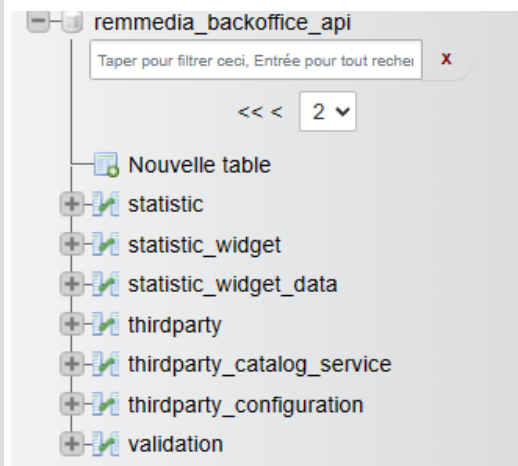
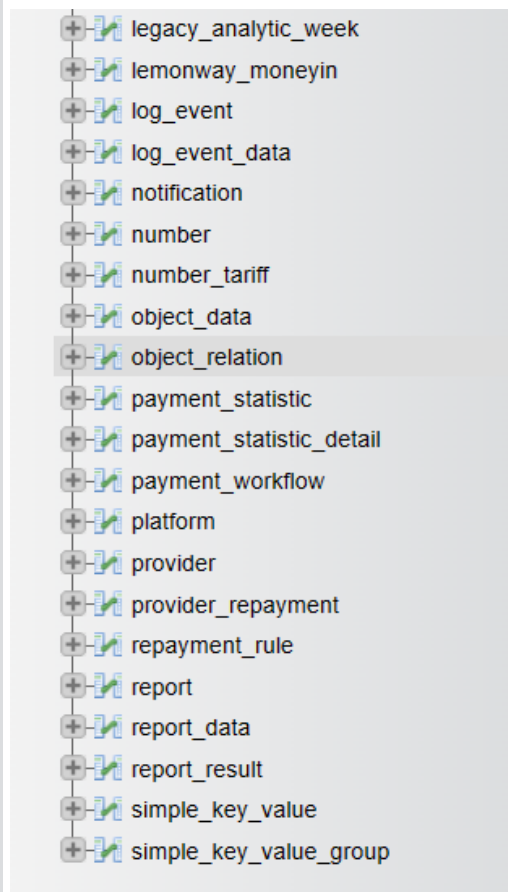
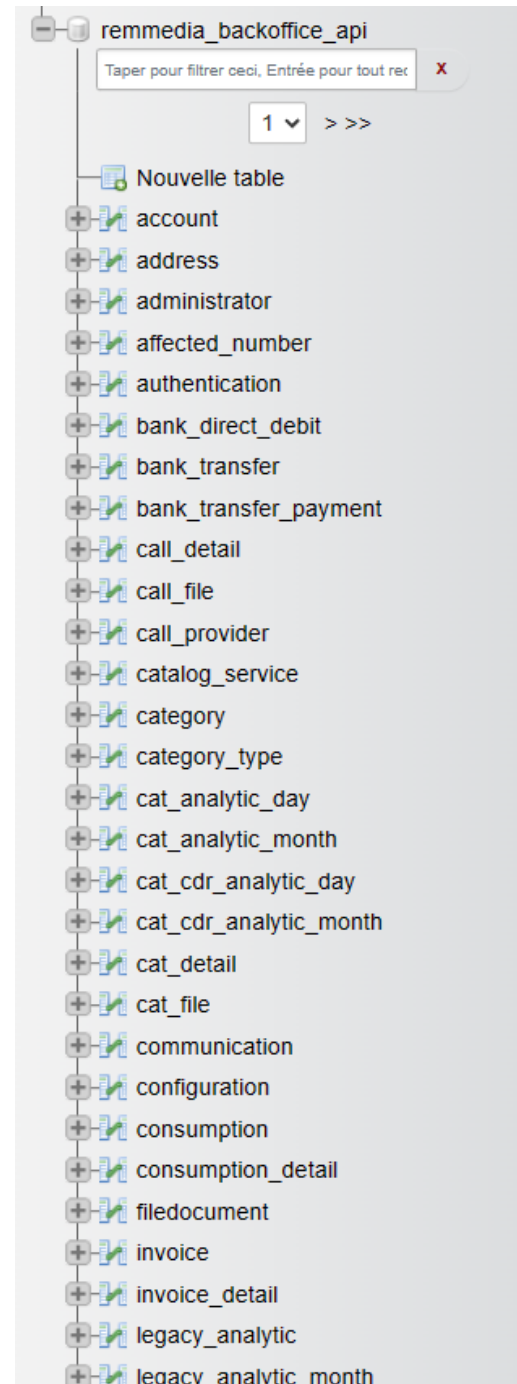
# CONCEPTION BACK-END DE L'APPLICATION

## Arborescence



## La base de données

Le projet étant fonctionnel a mon arrivée, la base de donnée était déjà donc, elle aussi, fonctionnelle et contient 57 tables



## RÉALISATIONS

Pour démontrer mes réalisations je m'appuierai donc sur la fiche client (le "client" sera soit une entreprise soit un particulier) qui aura accès à tout les controllers nécessaires cette page comprend:

### un Formulaire qui contient:

- les informations de l'entreprise ou du particulier (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, etc),
- de contacts,
- des communications (téléphone, mobile et mail des contacts)
- d'adresses des contacts.

## Controller & Model

### company

Pour commencer, si c'est une mise à jour, le controller fait appel au model contenant la classe "CardMapper" ainsi que de sa fonction "mapFromApi" (annexe screen 5). Cette fonction récupère toutes les informations relatives à l'entreprise telle que mentionnée dans le formulaire (annexe screen 7) ci-dessus et celui-ci est mappé avec ces informations selon une "Card" (annexe screen 2) .

La fonction "editPost" (annexe screen1) sert à enregistrer ou à mettre à jour les informations du client. Cette fonction prend en paramètre un uuid (qui est un identifiant unique) et un slug (généralement le nom de l'entreprise). ensuite on définit ces deux variables avec les information stockée en BDD avec:

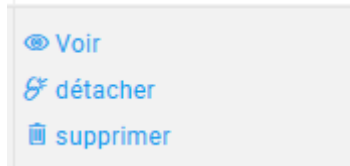
```
$uuid = $request->get('uuid');  
$slug = $request->get('slug');
```

puis on map les informations du formulaire selon la "Card" (annexe screen 2).

Si c'est un ajout on instancie les information récupérées dans cette "Card" et le controller fait appel au model contenant la classe "CardMapper" ainsi que de sa fonction "mapFromCard" (annexe screen 3) et sont enregistrées en BDD grace aux "Setters" (générés par swaggerHub (exemple annexe screen 4).

## Ged/FileDocuments

Pour les documents relatifs à l'entreprise, une liste de ces documents est accessible depuis la fiche client (annexe screen 6). Cette liste est récupérée grâce au controller “listController” et sa fonction “getParameters” qui crée un tableau (annexe screen 8) et insère dans celui-ci les informations récupérées en BDD. Pour chaque document, 3 actions sont définies :



- supprimer: qui permet de supprimer le document,
- détacher: qui permet de dissocier le document de l'entreprise,
- Voir: qui permet d'afficher le document

Pour l'affichage du document je me suis servi de la bibliothèque Javascript “Pdf.js” qui permet le rendu et la visualisation de fichier PDF et intégrer des paramètres pour que d'autres types de fichiers soient pris en compte tels que des fichiers image ('jpg', 'jpeg', 'png', 'gif')(annexe screen 9). Une fois que l'on clique sur le bouton “Voir” une modale s'ouvre avec le contenu désiré

## Composants d'accès aux données

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

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 étend "ApiController" (le controller parent de tous les controllers dont voici la fonctionnalité pour récupérer une liste de numéro:

```
<?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": {}}}
     * )
     */
}
```



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={"uuuid"})
 *     },
 *     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 $uuuid;

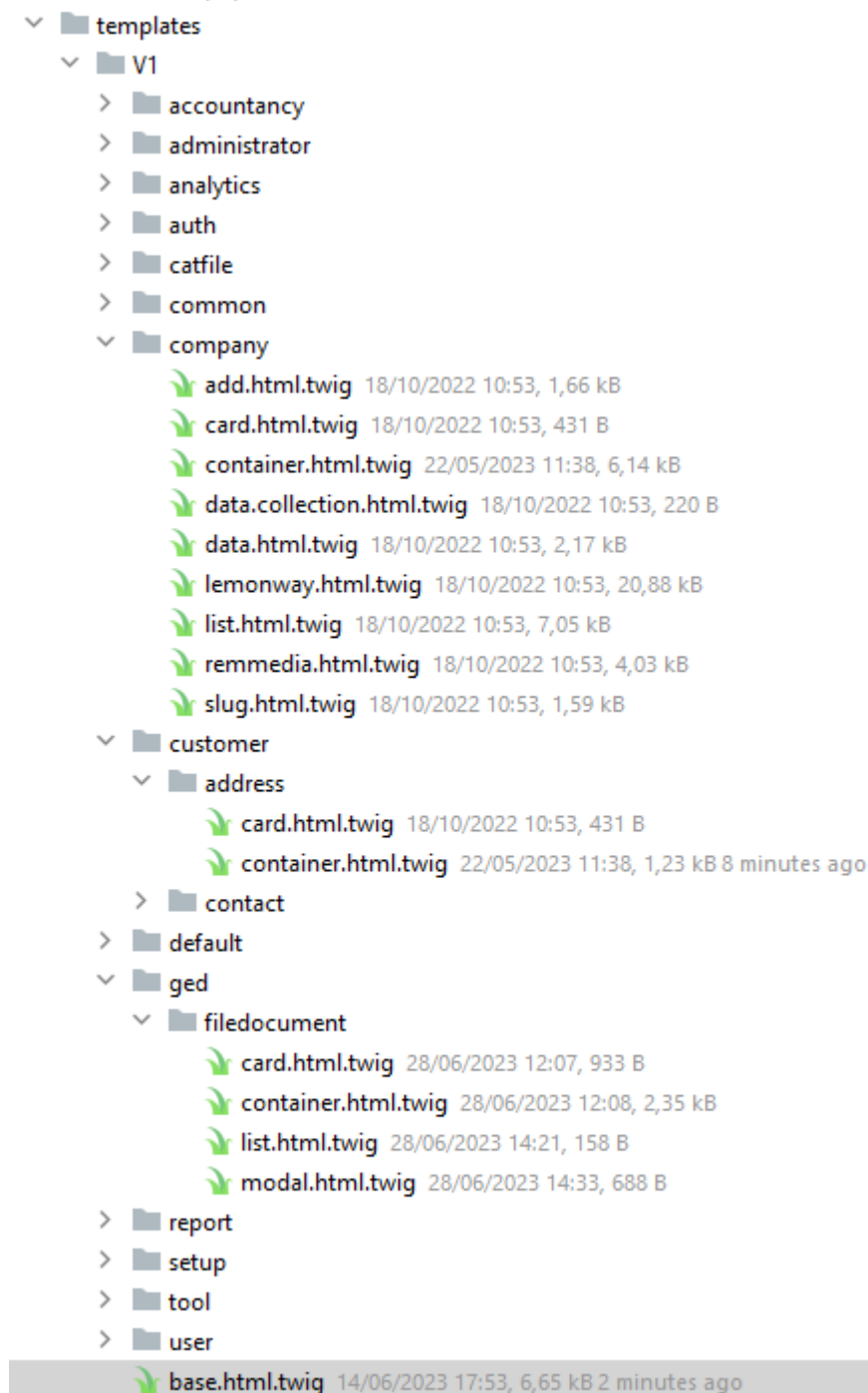
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    protected $id;

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

Grâce à la création de ces fichiers et 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.

## Affichage & Rendu

Pour l’affichage des informations demandées il existe un fichier **Twig** parent a tout les autres fichier Twig qui comprend les feuilles de style en cascade ainsi que les script Javascript (annexe screen 10) et pour chaque fonctionnalité un fichier twig distinct



par exemple le twig de la liste des document (list.html.twig) hérite du twig parent base.html.twig (pour les feuilles CSS et les scripts Javascript ) mais inclu également le twig contenant la modale nécessaire à l’affichage du document (annexe screen 11)

```
{% extends 'V1/common/page/list.html.twig' %}
{% block body %}
    {{ parent() }}
    {% include 'V1/ged/filedocument/modal.html.twig' %}
{% endblock %}
```

## Annexes

### screen 1

```
/**
 * @param $uuid
 * @param $slug
 * @param Request $request
 * @return mixed
 */
public function editPost(Request $request,
                        CompanyClient $companyClient,
                        AddressClient $addressClient,
                        CompanyFormManager $companyFormManager
)
{
    $res = false;
    $formData = $request->get( key: 'form');
    $uuid = $request->get( key: 'uuid');
    $slug = $request->get( key: 'slug');
    $step = $formData['step'] ?? $request->get( key: 'step');
    $action = $request->get( key: 'action');

    switch ($step) {
        case 'form_card':
            $companyCard = new Card();
            $formBuilder = $companyFormManager->getFormBuilder($companyCard, mode: 'save');
            $formBuilder->handleRequest($request);
            if ($formBuilder->isSubmitted() && $formBuilder->isValid()) {
                $mapper = new CardMapper();

                $apiCompanyObject = $mapper->mapFromCard($companyCard);

                if ($action == 'add' && $uuid == null) {
```

### screen 2

```
class Card
{
    // Main informations
    public $uuid = '';
    public $record_type = 1;
    public $firstname = '';
    public $name = '';
    public $brand_name = '';
    public $customer_code = '';
    public $customer_parent_code = '';
    public $website = '';
    public $state = true;
    public $slug = '';
    public $photo = '';
    public $photo_url = '';
    public $date_creation = null;
    public $date_modify = null;
    public $registration_id = '';
    // 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 = '';
    // Main contact
    public $main_contact = null;
```

screen 3

```
/**
 * @param Card $cardObject
 * @return ApiModelThirdparty
 */
public function mapFromCard(Card $cardObject){
    $apiCompanyObject = new ApiModelThirdparty();
    $apiCompanyObject->setUuid($cardObject->uuid);
    $apiCompanyObject->setFirstname($cardObject->firstname);
    $apiCompanyObject->setName($cardObject->name);
    $apiCompanyObject->setBrandName($cardObject->brand_name);
    $apiCompanyObject->setCustomerCode($cardObject->customer_code);
    $apiCompanyObject->setCustomerParentCode( customer_parent_code: $cardObject->customer_parent_code??'');
    $apiCompanyObject->setState($cardObject->state);
    $apiCompanyObject->setDiscountPercent($cardObject->discount_percent);
    return $apiCompanyObject;
}
```

screen 4

```
/**
 * Sets firstname
 *
 * @param string $firstname firstname
 *
 * @return $this
 */
public function setFirstname($firstname)
{
    $this->container['firstname'] = $firstname;

    return $this;
}
```

screen 5

```
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->brand_name = $apiModel->getBrandName();
    $cardObject->customer_code = $apiModel->getCustomerCode();
    $cardObject->customer_parent_code = $apiModel->getCustomerParentCode();
    $cardObject->state = $apiModel->getState();
    $cardObject->slug = $apiModel->getSlug();
    $cardObject->photo = $apiModel->getPhoto();
    $cardObject->photo_url = $apiModel->getPhotoUrl();
    $cardObject->discount_percent = $apiModel->getDiscountPercent();
    $cardObject->date_creation = $apiModel->getDateCreation();
    $cardObject->date_modify = $apiModel->getDateModify();
    // 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();
    }
    // 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
```

screen 6

Fiche Entreprise

Frederick Sonder

FREDERICK SONDER

Adresse

https://frederick-sonder.students-laplateforme.io/cin/

Code postal

13001.1

Ville

Marseille, Provence-Alpes

Pays

France

Contact principal

Email

fredericksonde@gmail.com.1

Téléphone

+33668445645.1

Mobile

afred711phone.1

Divers

Commentaires

raison19/05-2.1

Délai paiement (jours)

2

Origine

raison19/05-2.1

Numéro RCS

raison19/05-2.1

Afficher 10 éléments

Type	Période	N° actifs	Appels	Minutes	Référence	Total (€)	Marge	Date								
						Abo/Opt.	HT	TTC	Comm.	€	%	Prog. %	Généré	Envoyé	Modifié	Actions
Aucune donnée disponible dans le tableau																

Affichage de l'élément 0 à 0 sur 0 élément

Précédent Suivant

Liste des factures

Type	Date fac.	Num. facture	Total HT (€)	Total TTC (€)	Marge (%)	Date échéance	Date règlement	Modifié le	Actions
Aucune donnée disponible dans le tableau									

Fichiers (documents)

Fichier/Libellé

libel (Groupe-11586.jpg) #b037

Cat.: Certificat du Registre des Métiers < 3 mois

Type:Jpeg Taille:156.84 Ko Créé le:27/06/2023 09:20:59

Voir

détacher

supprimer

Fichier/Libellé

LIBELL2 (Model Conceptuel de Donnée.pdf) #ca32

Cat.: Pièce d'identité (CNI / Passeport) bénéficiaire effectif > 25%

Type:Pdf Taille:35.65 Ko Créé le:27/06/2023 10:20:05

Voir

détacher

supprimer

Actions +

screen 7

Backoffice

remmedia

Tableau de bord

Clients

Numéros

Compta

GED

Outils

Paramétrages

Fiche Entreprise

Frederick Sonder

Philippe STORA

Entreprise

Raison sociale \*

Remmedia.fr

Nom commercial

Remmedia.fr

Code client

1933

Code parent (vide si aucun)

SIREN/SIRET

Site web

Actif \*

Oui

Adresse principale

Adresse

5 cours Jean Ballard

Code postal

13001

Ville

Marseille

Pays

France

Contact principal

Jean-René ALONSO

serviceclient@remmedia.fr

Communications principales (contact principal)

Email

serviceclient@remmedia.fr

Téléphone

Mobile

Numéros

Valeur	Action
Aucune donnée	
Valeur	Action

Ajouter un numéro

Numéro \*

Enregistrer

Catégories

Valeur	Action
Aucune donnée	
Valeur	Action

Ajouter une catégorie

Catégorie \*

Enregistrer

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

Afficher 10 éléments

Type	Période	N° actifs	Appels	Minutes	Référence	Total (€)	Marge	Date								
						Abo/Opt.	HT	TTC	Comm.	€	%	Prog. %	Généré	Envoyé	Modifié	Actions
FAC	Nov. 2021	0	0	0,00	R09100 Ref : 202111-1933	0,00	0,00	0,00	0,00	0,00	0,00	1,00			06/12/2021 17:29:38	
FAC	Oct. 2021	1	1	0,00	R08986 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	F08669 Ref : 202109-1933	0,00	0,00	0,00	0,00	0,00	0,00	0,00			04/10/2021 20:55:58	
FAC	Août 2021	0	0	0,00	F08490 Ref : 202108-1933	0,00	0,00	0,00	0,00	0,00	0,00	1,00			07/09/2021 09:19:38	
FAC	Juil. 2021	1	6	0,00	F08173 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	F07880 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	F07549	0,00	0,00	0,00	0,00	-0,01	-1,00	0,00			04/06/2021	

screen 8

```
/**
 * getParameters
 * @param Request $request
 * @return array
 */
protected function getParameters(Request $request)
{
    $args = $request->query->all();
    return [
        'title' => 'Liste des documents/fichiers',
        'javascripts' => [
            '/assets-v1/app/ged/document/list.js'
        ],
        'datatable' => [
            'datatableId' => 'datatableFiledocument',
            'labels' => [
                'addItem' => 'Ajouter un document',
                'noItem' => 'Aucun document'
            ],
            'urls' => [
                'ajax' => [
                    'url' => 'v1_ged_filedocuments_ajax',
                    'args' => $args
                ],
                'addItem' => 'v1_ged_filedocument_add',
                'updateItem' => 'v1_ged_filedocument_update',
                'deleteItem' => 'v1_ged_filedocument_delete'
            ],
            'columns' => [
                [
                    'data' => ['id' => 'filename', 'label' => 'Nom de fichier', 'hidden' => false, 'format' => 'text', 'value' => ''],
                    'layout' => ['class' => 'col-filedocument-filename', 'orderable' => true],
                    'filter' => ['name' => 'filterFilename', 'type' => 'text', 'apiField' => 'filename', 'apiDefaultOperator' => 'cs'],
                ],
                [
                    'data' => ['id' => 'mimetype', 'label' => 'Type', 'hidden' => false, 'format' => 'text', 'value' => ''],
                    'layout' => ['class' => 'col-filedocument-mimetype', 'orderable' => true],
                    'filter' => ['name' => 'filterMimetype', 'type' => 'text', 'apiField' => 'mimetype', 'apiDefaultOperator' => 'cs'],
                ],
                [
                    'data' => ['id' => 'filesize', 'label' => 'Taille', 'hidden' => false, 'format' => 'text', 'value' => ''],
                    'layout' => ['class' => 'col-filedocument-filesize', 'orderable' => true],
                    'filter' => ['name' => 'filterFilesize', 'type' => 'none', 'apiField' => 'role', 'apiDefaultOperator' => 'eq'],
                ],
                [
                    'data' => ['id' => 'dateModify', 'label' => 'Modifié le', 'hidden' => false, 'format' => 'date', 'value' => ''],
                    'layout' => ['class' => 'col-filedocument-dateModify col-date', 'orderable' => true],
                    'filter' => ['name' => 'filterDateModify', 'type' => 'daterange', 'apiField' => 'date_modify', 'apiDefaultOperator' => 'bt']
                ],
            ]
        ]
    ];
}
```



screen 9

```
function globalBindDocumentButtons() {
  const openModalButtons = document.getElementsByClassName( className: 'btn-open-pdf');

  for (let openModalButton of openModalButtons) {
    openModalButton.addEventListener( type: 'click', listener: function (e :Event) {
      const clickedButton = e.currentTarget;
      const canvas = document.getElementById( elementId: 'pdfCanvas');
      const pdfPath = clickedButton.getAttribute("data_fileurl");
      const fileUrl = clickedButton.getAttribute("data_fileurl");

      const fileType = getFileType(fileUrl); // Obtenir le type de fichier à partir de l'URL (par exemple, extension du fichier)
      if (fileType === 'pdf') {
        // Traitement des fichiers pdf
        pdfjsLib.getDocument(pdfPath).promise.then(file => {
          return file.getPage(1);
        }).then(page => {
          const scale = 1;
          const viewport = page.getViewport({scale});
          const context = canvas.getContext('2d');

          canvas.width = viewport.width;
          canvas.height = viewport.height;

          const renderContext = {
            canvasContext: context,
            viewport: viewport
          };
          page.render(renderContext);
        }).catch(error => {
          console.error('Une erreur est survenue lors du chargement du fichier: ', error);
        });
      } else if (fileType === 'image') {
        // Traitement des fichiers image
        const image = new Image();
        image.onload = function() {
          // Affichage de l'image sur le canvas
          const context = canvas.getContext('2d');
          canvas.width = image.width;
          canvas.height = image.height;

          context.drawImage(image, 0, 0);
        };
        image.src = fileUrl;
      } // } else if (fileType === 'video') {
      // // Traitement des fichiers vidéo
      // @TODO enable if you want to see video
      } else {
        console.error('Impossible d\'afficher le fichierType de fichier non pris en charge.');
```

screen 10

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="logo-symbol.png" sizes="32x32" />
  <link rel="icon" href="logo-symbol.png" sizes="192x192" />
  <link rel="apple-touch-icon" href="logo-symbol.png" />
  <meta name="msapplication-TileImage" content="logo-symbol.png" />
  <base href="/assets-v1/assets/">
  <title>{% block title %}Backoffice - REMMEDIA{% endblock %}</title>
  <script type="text/javascript" src="{{ asset('assets-v1/assets/jquery/jquery.js') }}"></script>
  <script type="text/javascript" src="{{ asset('assets-v1/assets/vendor/alertify/alertify.min.js') }}"></script>
  {% block stylesheets %}
    <link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/vendor/bootstrap-4.4.1-dist/css/bootstrap.min.css') }}" />
    <link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/vendor/fontawesome-free-5.12.0-web/css/all.css') }}">
    <link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/floating-labels.css') }}" />
    <link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/style.css') }}" />
    <link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/main.css') }}" />
    <link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/vendor/alertify/css/alertify.min.css') }}" />
    <link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/vendor/bootstrap-datepicker/dist/css/bootstrap-datepicker3.min.css') }}" />
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>      {# import de la bibliotheque bootstrap #}
    <script src="{{ asset('assets-v1/assets/pdf.js/pdf.js/build/pdf.js') }}"></script>      {# bibliotheque pdf.js dans le projet #}
    <script src="{{ asset('assets-v1/assets/pdf.js/pdf.js/build/pdf.worker.js') }}"></script>      {# bibliotheque pdf.js dans le projet #}
    <script src="{{ asset('assets-v1/assets/pdf.js/pdf.js/affichPdf.js') }}"></script>      {# récupération du script Js d'affichage du pdf #}

    {#<link rel="stylesheet" type="text/css" href="{{ asset('assets-v1/assets/vendor/alertify/css/themes/bootstrap.min.css') }}" />#}
  {% endblock %}
```

screen 11

```
<div class="modal" id="myModal">
  <div class="modal-dialog modal-xl">
    <div class="modal-content">
      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title"></h4>
        <button type="button" class="btn-close" data-bs-dismiss="modal"><span class="ti-close"></span></button>
      </div>
      <!-- Modal Content -->
      <canvas id="pdfCanvas"></canvas>
      <!-- Modal footer -->
      <div class="modal-footer">
        <button type="button" class="btn btn-danger" data-bs-dismiss="modal">Fermer</button>
      </div>
    </div>
  </div>
</div>
```

Model

Sécurité





