

CHAVANNE ENZO

# RAPPORT DE STAGE

4 mars – 12 avril 2024

Projet CRM



Laravel



Android Studio



MySQL

# SOMMAIRE

<b>01</b>	<b>Expression des besoins .....</b>	<b>1</b>
<b>1.1</b>	Contexte, domaine, processus métier.....	1
<b>1.2</b>	Demandeur, acteurs, utilisateurs.....	1
<b>1.3</b>	Etude de l'existant, diagnostic.....	1
<b>1.4</b>	Description de la demande, objectifs, bénéfices attendus	1
<b>1.5</b>	Spécifications fonctionnelles.....	2
<b>1.6</b>	Contraintes ou exigences.....	2
<b>02</b>	<b>Conception, spécifications techniques</b>	<b>3</b>
<b>2.1</b>	Description de la solution.....	3
<b>2.2</b>	Outils logiciels de la solution.....	3
<b>2.3</b>	Architecture matérielle et logicielle de la solution.....	5
<b>2.4</b>	Besoins techniques et ressources.....	6
<b>2.5</b>	Analyse des données.....	6
<b>2.6</b>	Interface homme-machine et maquettage.....	4
<b>2.6</b>	Conduite de projet.....	4
<b>03</b>	<b>Développement .....</b>	<b>9</b>
<b>3.1</b>	Réalisation des interfaces et programmes conformes aux spécifications fonctionnelles attendues.....	10
<b>3.2</b>	Dossier de programmation codes sources documentés et commentés.....	10
<b>3.3</b>	Difficultés rencontrées.....	11
<b>04</b>	<b>Exploitation et mise en production.....</b>	<b>14</b>
<b>4.1</b>	Tests.....	14
<b>4.2</b>	Installation et déploiement.....	X
<b>4.3</b>	Rédaction notice utilisateur.....	X
<b>4.4</b>	Formation des utilisateurs.....	X
<b>05</b>	<b>Bilan.....</b>	<b>14</b>

01

---

## **EXPRESSION DES BESOINS**

## 1.1) Contexte, domaine, processus métier

DAHST, société holding, comprend deux sociétés : Restauration Pour Collectivités (RPC) et Sud Est Restauration (SER). La première est une entreprise spécialisée dans la restauration collective (~45 000 repas produits et livrés par jour) pour le scolaire, les centres de loisirs et les personnes âgées (portage) tandis que la seconde intervient dans les secteurs médico-sociaux, handicaps et scolaires (les chefs cuisinent sur place).

## 1.2) Demandeur, acteurs, utilisateurs

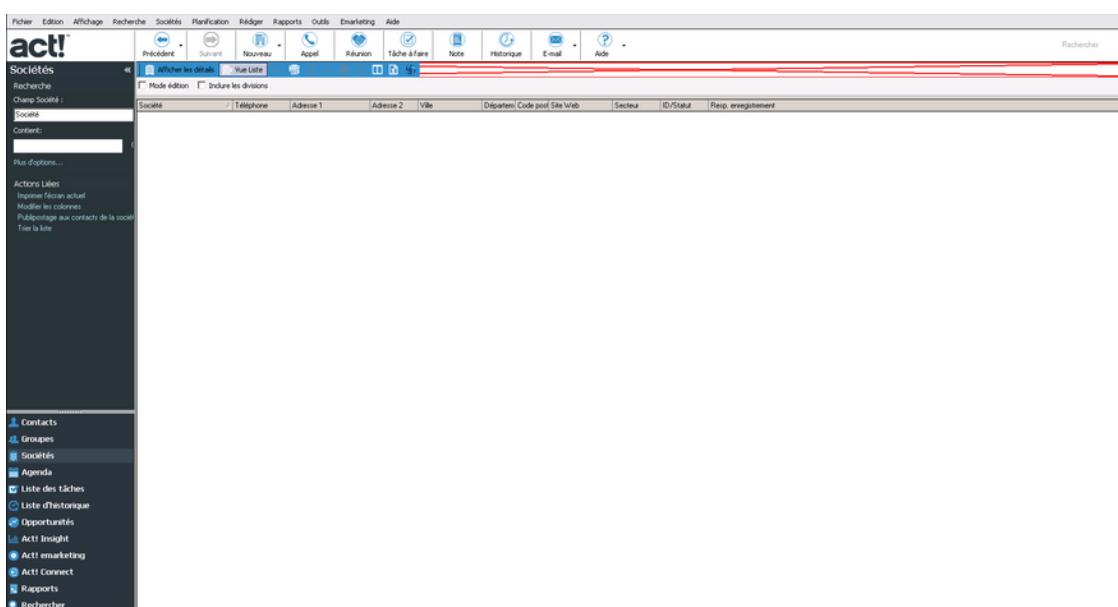
Après avoir reçu plusieurs entreprises proposant des services de CRM mais n'ayant retenu aucune de celle-ci, le directeur commercial a sollicité le service informatique, avec l'appui de la direction générale, pour réaliser le projet en interne afin qu'il corresponde au plus proche des attentes des commerciaux. Le directeur commercial est donc le demandeur et porteur de projet. Il sera aussi utilisateur avec les commerciaux de Sud-Est-Restauration (SER).

Je suis en charge avec mon tuteur de la réalisation du projet. Nous travaillerons en collaboration avec le pôle commercial tout au long du projet.

## 1.3) Etude de l'existant, diagnostic

Actuellement, l'entreprise détient un logiciel qui n'est plus maintenu par l'équipe de développement du prestataire. Le logiciel est obsolète, ne correspond plus aux attentes actuelles des utilisateurs et comporte des risques de sécurité. Il n'est visuellement plus au goût du jour et délivre une mauvaise expérience utilisateur.

capture d'écran du logiciel actuel :



## **1.4) Description de la demande, objectifs, bénéfices attendus**

L'objectif est de mettre en place un nouvel outil dans le but de remplacer le logiciel existant. Cet outil permettra de suivre l'activités des commerciaux tant sur la prospection que sur le suivi de la clientèle ainsi que la prise de rendez-vous. Les bénéfices attendues seront notamment un gain de temps, une meilleure ergonomie et expérience utilisateur ainsi qu'un meilleur échange d'information entre les commerciaux .

## **1.5) Spécifications fonctionnelles**

Les fonctionnalités attendues sont la possibilité de gérer (création, mise à jour, suppression) les prospects et les clients simplement. Le logiciel doit aussi permettre d'accéder à un calendrier dans lequel un commercial pourra consulter ses rendez-vous et ceux de ses collaborateurs. Une fois le rendez-vous terminé, il sera possible de rédiger un compte-rendu de cette réunion, visible par tous. Enfin, il faut qu'un commercial puisse ajouter toutes sortes d'informations supplémentaires liées à un client ou un prospect, tels que des remarques, demandes... en spécifiant l'origine

- Possibilité d'ajouter des informations supplémentaires pour un client ou prospect, en spécifiant l'origine (téléphone, mail ..).

## **1.6) Contraintes ou exigences (matérielles, techniques, délais, budget, ...)**

- Serveur web avec Apache
- Durée : 6 semaines
- Aucun budget n'a été définit
- Maitrise des fondamentaux de PHP 8 et Laravel 10 et notion API

02

---

**CONCEPTION, SPÉCIFICATION,  
TECHNIQUES**

## 2.1) Description de la solution

Mise en place d'un CRM (Customer Relationship Management)

## 2.2) Outils logiciels de la solution

### Looping

Looping est un logiciel open source développé par les étudiants de l'Université Toulouse III. Il permet la modélisation conceptuelle de données.

### Postman

Postman est une application permettant de tester des API, avec la possibilité d'organiser les différents appels dans des projets.

### Figma

Figma est un outil de design collaboratif pour la création d'interfaces utilisateurs.

### Ganttproject

Ganttproject permet de planifier un projet à travers la réalisation de diagrammes de Gantt. C'est un logiciel open source.

### Visual Studio Code

Visual Studio est un environnement de développement complet de Microsoft, permettant la création d'applications logicielles, de sites web et d'applications mobiles

### PhpMyAdmin

phpMyAdmin est une interface web open source pour gérer les bases de données MySQL. Elle permet de créer, modifier et interroger les bases de données



Laravel est un framework de développement web open-source en PHP, offrant une structure élégante et expressive pour construire des applications web rapidement et efficacement.

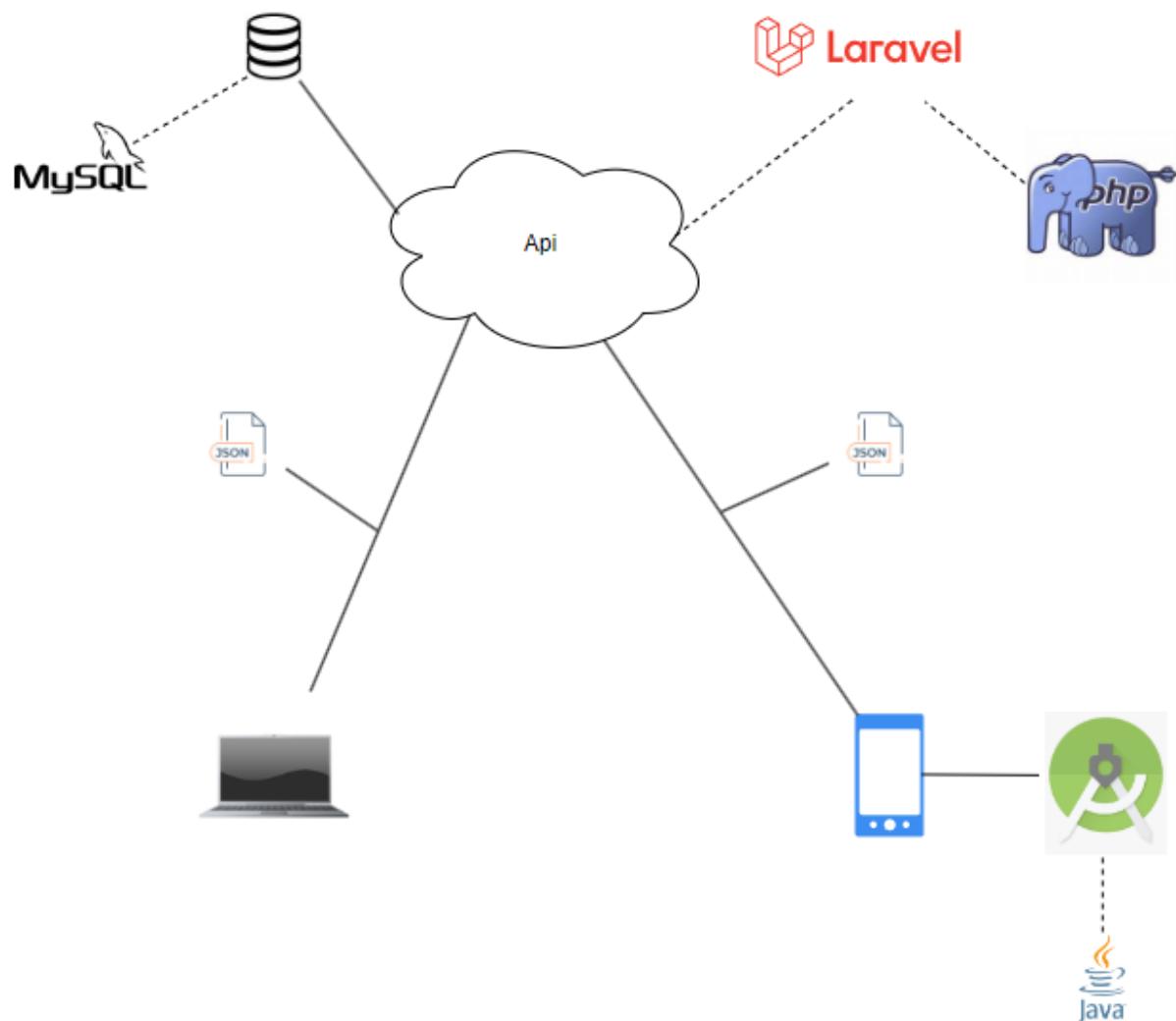
## Eloquent

L'ORM Eloquent est un outil de mapping objet-relationnel intégré au framework PHP Laravel, permettant de simplifier la manipulation des bases de données en utilisant des modèles et des relations définis en PHP plutôt qu'en SQL brut.

## 2.3) Architecture matérielle et logicielle de la solution

L'API, développée à l'aide du framework Laravel, est le point central du projet. C'est par l'API que toutes les données, issues d'une base de données sous le SGBD MySql, vont transiter. Elle sera en charge de valider les données en entrée, de vérifier les droits d'accès et de retourner les données attendues. Les données seront retournées au format JSON aux divers clients, que ce soit web, mobile, postman...

La partie Web est encore au stade de réflexion interne quant aux technologies qui seront utilisées, tandis que la partie mobile sera développée en Java avec l'outil Android Studio. Le choix s'est porté sur cette technologie car l'entreprise n'a pas de besoin crossplatform ; seul des appareils android seront utilisés.



## 2.4) Besoins techniques, ressources (humaines, matérielles, logicielles et budgétaires, coûts)

### Besoins matériels

Pour la réalisation de ce projet, nous allons avoir besoin d'ordinateurs suffisamment puissant pour exécuter simultanément une API, un SGBD et Android Studio pour développer le client.

De plus, lors de la phase de mise en production, nous aurons besoin d'un serveur. L'entreprise RPC ayant déjà plusieurs contrats chez le fournisseur OVH, le serveur de production sera également hébergé chez OVH.

### Besoins logiciels

L'environnement de développement (IDE) utilisé est Visual Studio Code. Il sera agrémentés d'extensions pour rendre le développement plus ergonomique.

Pour tester les endpoints API, nous utilisons le logiciel Postman dans sa version gratuite qui est suffisante. Enfin, wamp...

## 2.5) Analyse des données (modélisation, diagramme de classes, schéma relationnel)

La base de données est composée d'utilisateurs (users) qui pourront créer/modifier une ou plusieurs notes (des informations sur un clients). Le modèle "Customers" représente les entreprises qui sont rattaché à plusieurs contacts. Chaque utilisateur pourra organiser un rendez-vous avec un contact.

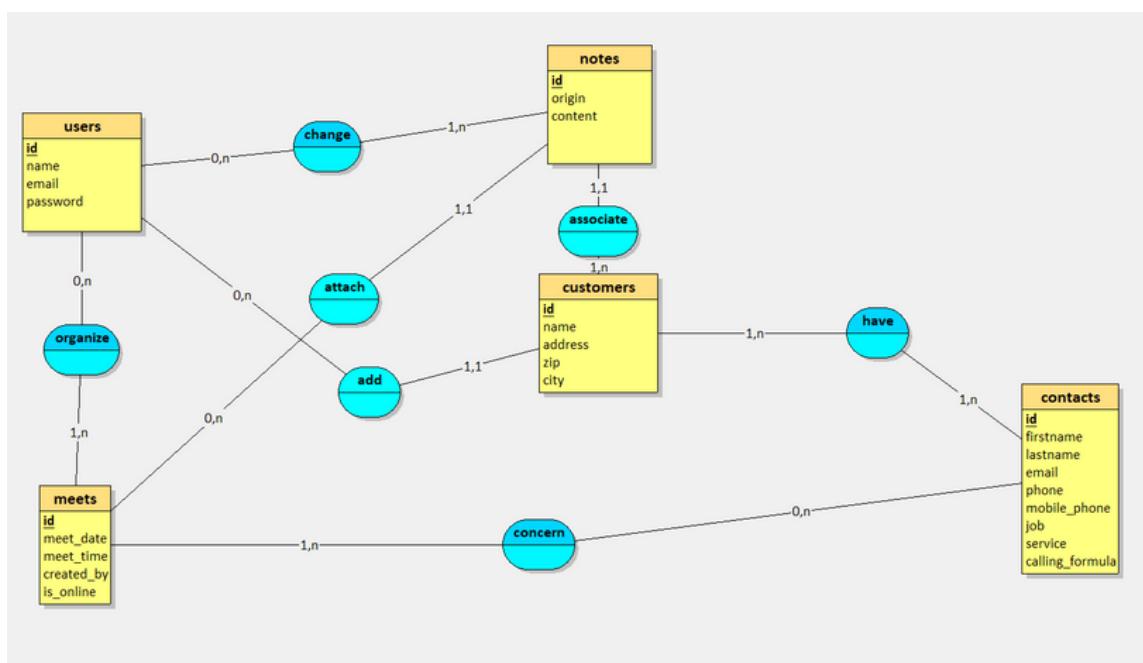
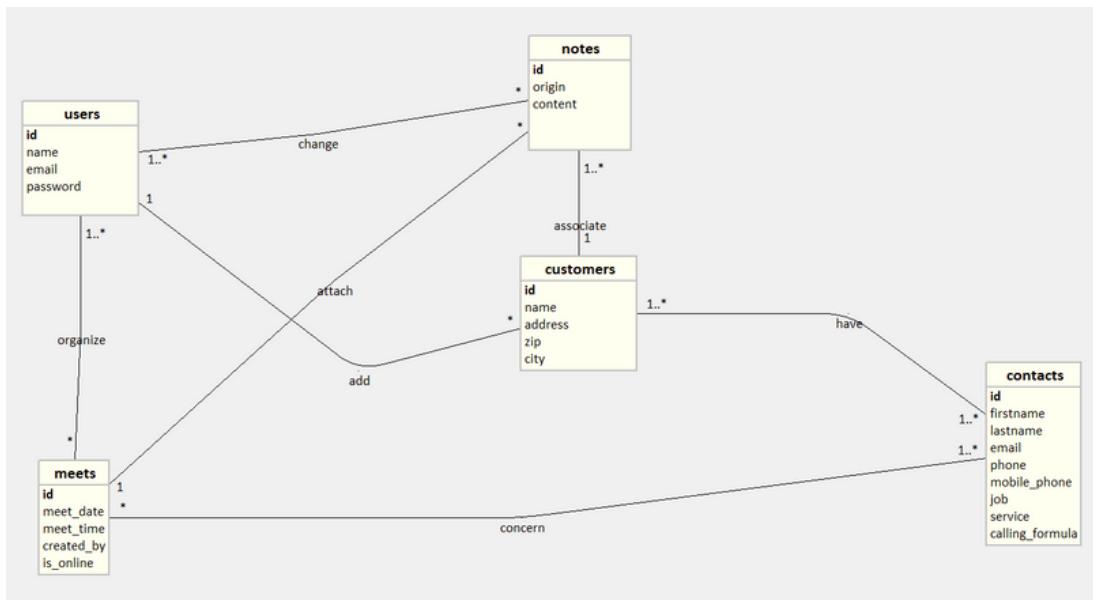
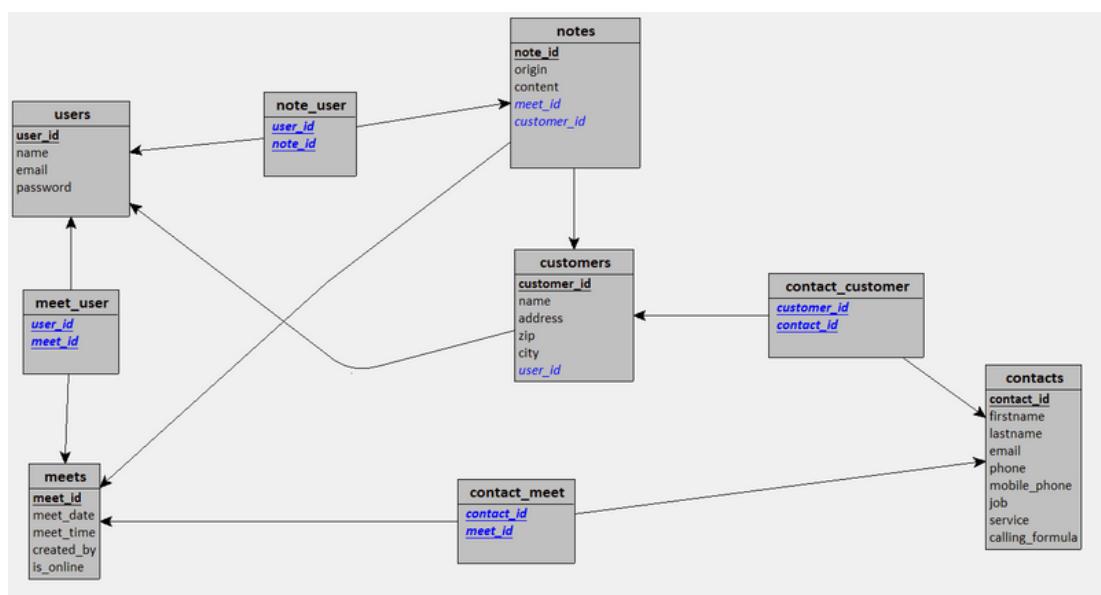


Diagramme MCD (Model Conceptuel de Données)



## Diagramme UML (Unified Modeling Language)



## Diagramme MLD (Modèle Logique des Données)

## 2.6) IHM (Interface Homme-Machine)

Ici nous avons une ébauche de la futur interface mobile qui sera utilisé par les employés de l'entreprise, ainsi qu'une version web

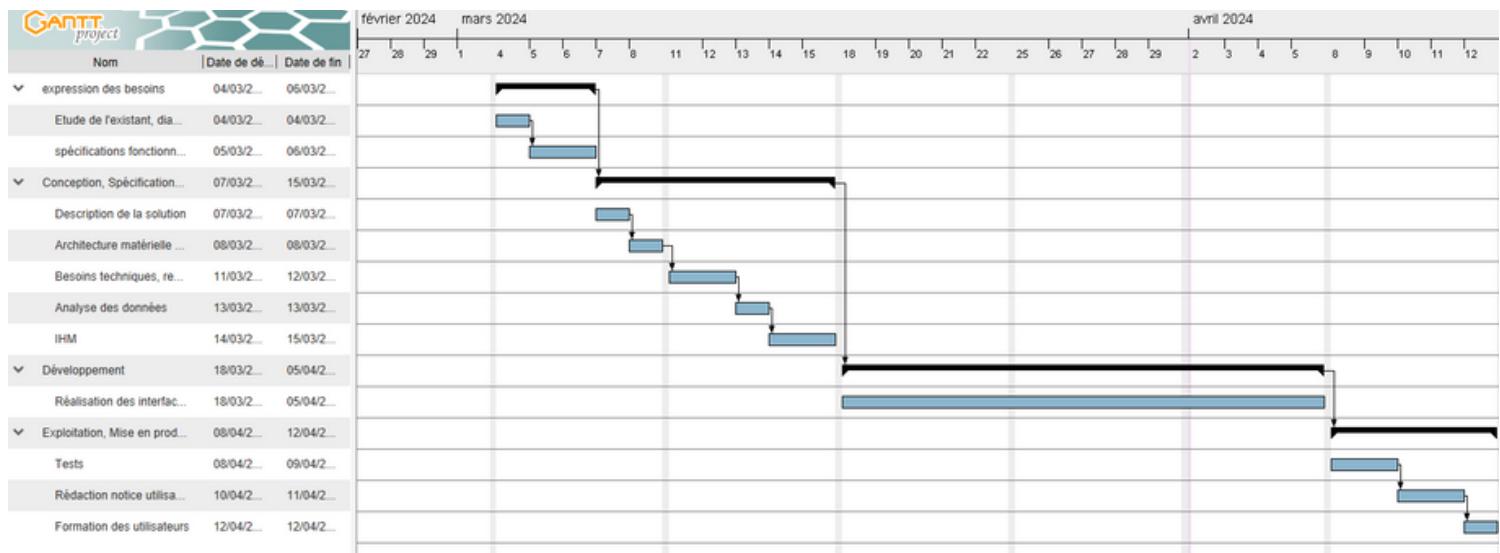


## Maquette interface Web

Réalisation d'une maquette sur Figma pour l'interface web, que je ne réaliserais pas pendant ma période de stage par manque de temps.

A wireframe of a CRM web application. At the top, there is a header bar with the 'CRM' logo on the left and four menu items: 'Contact' (with a phone icon), 'Groupes' (with two people icons), 'Sociétés' (with a building icon), and 'Agenda' (with a calendar icon). To the right of these is a search bar with the placeholder 'rechercher' and an 'ok' button. Below the header is a large input form for a new contact. It contains several pairs of input fields: 'Contact' and 'Téléphone', 'Société' and 'E-mail', 'Titre' and 'Adresse', 'service' and 'Ville', and 'Formule' and 'code postal'. At the bottom of the page, there is a navigation bar with three tabs: 'Activités', 'Historiques', and 'Notes'.

## 2.7) Conduite de projet



L'équipe est composé de mon tuteur et moi même, chaque fin de tache sera vérifié puis validé par le tuteur, tout ceci sur une durée de 6 semaines.

**03**

---

**DÉVELOPPEMENT**

# Création d'une migration

Une migration en Laravel est une classe permettant de gérer et de versionner une table de la base de données. L'ensemble des migrations représente la structure de la BDD. Dedans, nous spécifions pour chaque champ son nom, son type et s'il est nullable (false par défaut).

Pour affecter les modifications sur la BDD, il faut utiliser la commande *php artisan migrate*

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('customers', function (Blueprint $table) {
            //crée automatiquement une colonne qui est primary keys et autoincrement
            $table->id();
            $table->string('name');
            $table->string('address');
            $table->string('zip');
            $table->string('city');
            $table->unsignedBigInteger('created_by');
            //crée la colonne created_at et updated_at de type timestamps
            $table->timestamps();
            //implémentation de la clé étrangère
            $table->foreign('created_by')->references('id')->on('users');

        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('customers');
    }
};
```

# Création d'un modèle

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\BelongsToManyRelationship;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasOne;

5 references | 0 implementations
class Customer extends Model
{
    use HasFactory;

    //indique les propriétés modifiables
    0 references
    protected $fillable = [
        'name',
        'address',
        'zip',
        'city',
        'created_by',
    ];
}

//un customer peut avoir 0 ou plusieurs contact
0 references | 0 overrides
public function contacts(): BelongsToMany
{
    return $this->belongsToMany(Contact::class);
}

//un customer possède un utilisateur (un utilisateur crée un customer)
0 references | 0 overrides
public function user(): HasOne
{
    return $this->hasOne(User::class);
}
```

Un modèle en Laravel est une classe PHP qui représente une table de la base de données : il fait la liaison. Laravel utilise des conventions de nommage qui permettent de développer plus rapidement, notamment sur les clés primaires et étrangères. Dans le cas où nous souhaitons déroger à ses conventions, c'est dans le modèle que nous spécifierons les changements.

Nous spécifions aussi quelques propriétés et les méthodes faisant les relations entre modèles.

# Création d'une route

```
<?php

use App\Http\Controllers\ContactController;
use App\Http\Controllers\CustomerController;
use App\Http\Controllers\MeetController;
use App\Http\Controllers>NoteController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
| */

//Les routes permettent d'acheminer les requêtes vers les contrôleurs
Route::apiResource('/contacts', ContactController::class);
Route::apiResource('/meets', MeetController::class);
Route::apiResource('/customers', CustomerController::class);
Route::apiResource('/notes', NoteController::class);
```

Une route en Laravel est un mécanisme de mapping qui associe une URL spécifique à une action ou une fonction de l'application, permettant ainsi de diriger les requêtes HTTP vers les contrôleurs appropriés.

# Création d'une form request

Une "form request" en Laravel est une classe qui encapsule et représente les données entrantes d'une requête HTTP, permettant ainsi leur traitement et leur validation avant de les utiliser dans le contrôleur.

```
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

2 references | 0 overrides
class UpdateContactRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
0 references | 0 overrides
    public function authorize(): bool
    {
        return false;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>/string>
     */
0 references | 0 overrides
    public function rules(): array
    {
        return [
            'email' => 'sometimes|required|email:rfc',//on peut valider que si le champ est présent/qu'il soit non-nul/doit respecter le format d'un mail
            'firstname' => 'sometimes|required',
            'lastname' => 'sometimes|required',
            'phone' => "sometimes|required|phone:FR",
            'mobile_phone' => "sometimes|required|phone:FR",
            'function' => 'sometimes|required',
            'service' => 'sometimes|required',
            'calling_formula' => 'sometimes|required',
        ];
    }
}
```

# Création d'un controller

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\StoreCustomerRequest;
use App\Http\Requests\UpdateCustomerRequest;
use App\Models\Customer;
use App\Models>Note;
use Illuminate\Http\Request;

2 references | 0 implementations
class CustomerController extends Controller
{
    //montrer plusieurs customer
    0 references | 0 overrides
    public function index()
    {
        //
    }

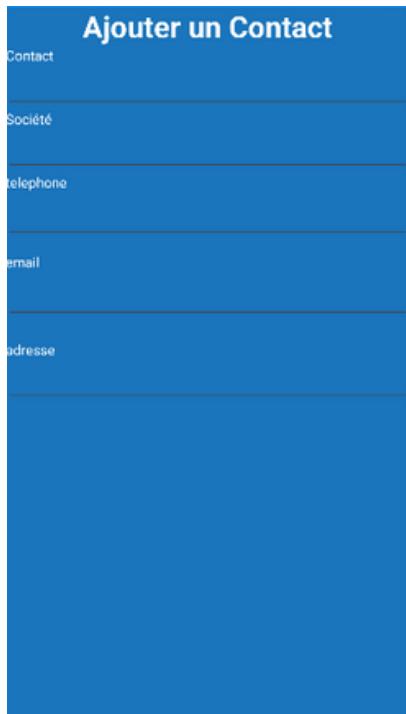
    //pour ajouter un customer
    0 references | 0 overrides
    public function store(StoreCustomerRequest $request)
    {
        Customer::create($request->validated());
    }

    //montrer un seul customer en fonction de l'id
    0 references | 0 overrides
    public function show(Customer $customer)
    {
        return response()->json($customer);
    }

    //pouvoir mettre a jour un customer
    0 references | 0 overrides
    public function update($request, $note)
    {
        $note->update($request->validated());
    }
}
```

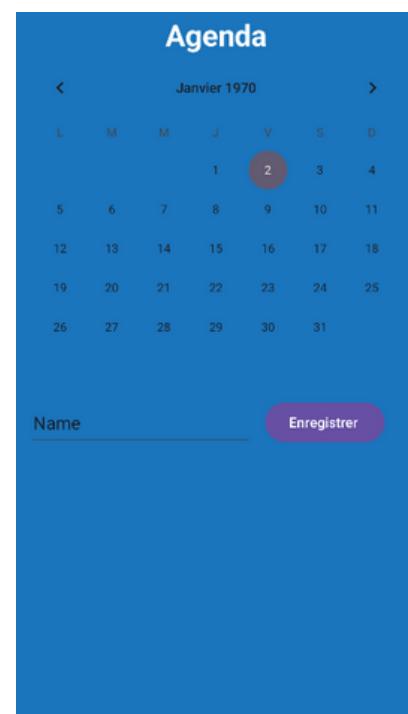
Un controller en Laravel est une classe PHP qui organise la logique métier de l'application en traitant les requêtes HTTP et en contrôlant les interactions entre le modèle et la vue. Dans le cas de notre API, les contrôleurs renvoient les données sous forme de JSON.

# Application android



ici nous avons la page dans laquelle les utilisateurs pourront ajouter ajouter un contact

ici nous avons l'agenda dans lequel les utilisateurs pourront ajouter des rendez vous



## 3.3) Difficultés rencontrées

Pour commencer des difficultés ont été rencontrées sur Laravel notamment sur la mise en place de concepts qui n'ont pas été vu en cours, également pour Android studio. Des difficultés matérielles ont également été rencontrées pour Android Studio qui provoque des problèmes sur l'ordinateur (freeze).

# 04

---

## **EXPLOITATION ET MISE EN PRODUCTION**

## 4.1) Test

le logiciel s'appelle Postman permet de faire des tests de pouvoir savoir si l'envoi des données avec l'api s'effectue correctement

The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:8000/api/contacts/1`. The response is successful (200 OK) with a duration of 604 ms and a size of 539 B. The response body is displayed in Pretty JSON format:

```
1 "id": 1,
2 "firstname": "oui",
3 "lastname": "non",
4 "email": "test@gmail.com",
5 "phone": "33625482315",
6 "mobile_phone": "6647247153",
7 "function": "directeur",
8 "service": "achat",
9 "calling_formula": "mme",
10 "created_at": "2024-03-05T13:55:02.000000Z",
11 "updated_at": "2024-03-05T13:55:02.000000Z"
```

## Bilan du stage

Le stage a été centrer sur le développement d'une API et a nécessité l'utilisation de laravel ce qui ma permit d'approfondir beaucoup plus les connaissance en laravel que je possédais déjà, de pouvoir voir des choses qui pourront dans le futur me servir dans les études supérieur ou dans la vie active/monde du travail