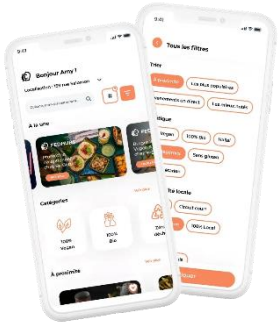
RAPPORT DE MISSION

Développeur back-end
FEDHUBS

MALLE TRAORE
2eme année de BTS
SIO SLAM
A L'ENC PARIS

A - Présentation du projet



Fedhubs est une société de services qui a pour objectif de transmettre tous les moyens digitaux aux commerçants afin de susciter l'intérêt des consommateurs.

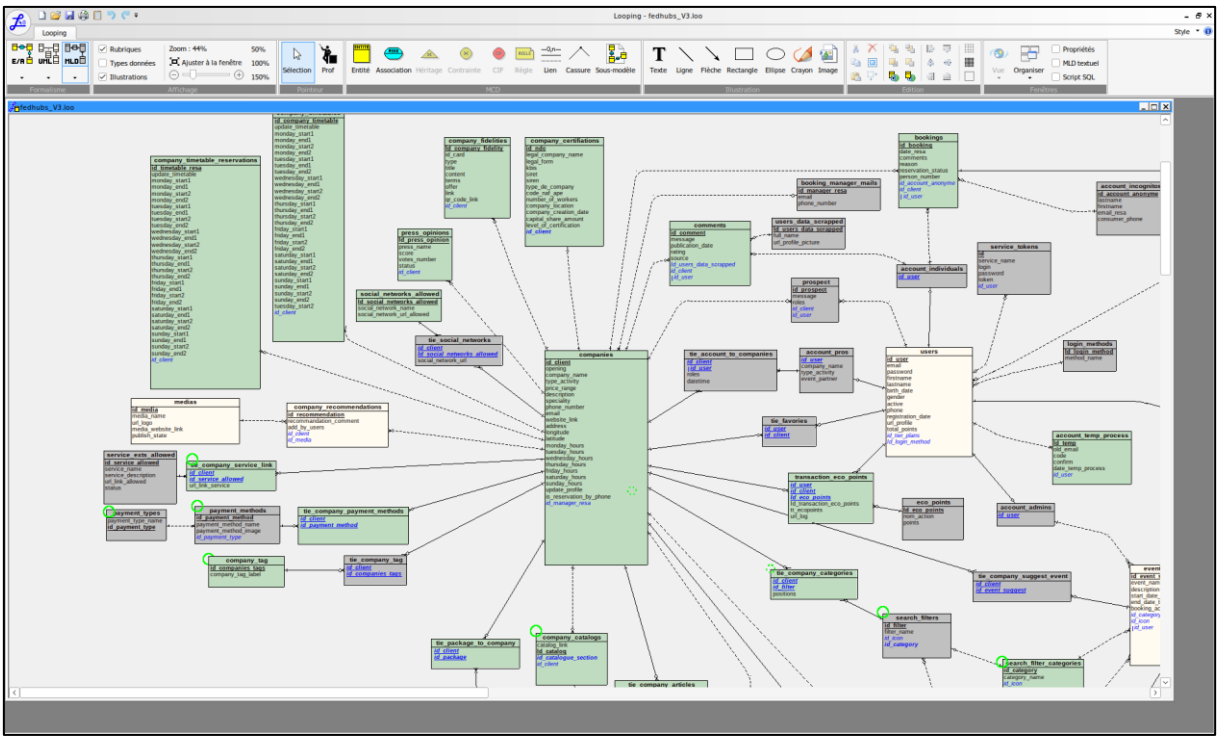
Tout se passe via l'application mobile qui recense les bars et restaurants combinant des concepts variés et qui organisent des animations culturelles.

1 – Les ressources

- Outils de FedHubs
- Prérequis
- Installation et ajout du serveur dans XAMPP
- Structure et organisation de Laravel
- Gitlab et Git
- Mise en place de Sourcetree
- Models, Migrations et Seeders
- Routes et controllers
- BDD avec looping
- Explication et Création : CRUD / Relation
- Commandes utiles de php:artisan
- Présentation de phpMyAdmin
- Présentation d'Hoppscotch
- Lancer le Back-end
- ACL et packages
- Swagger
- Backup Laravel
- Test unitaire
- Cas d'usage des tests unitaire
- Les 2 mode de Matching

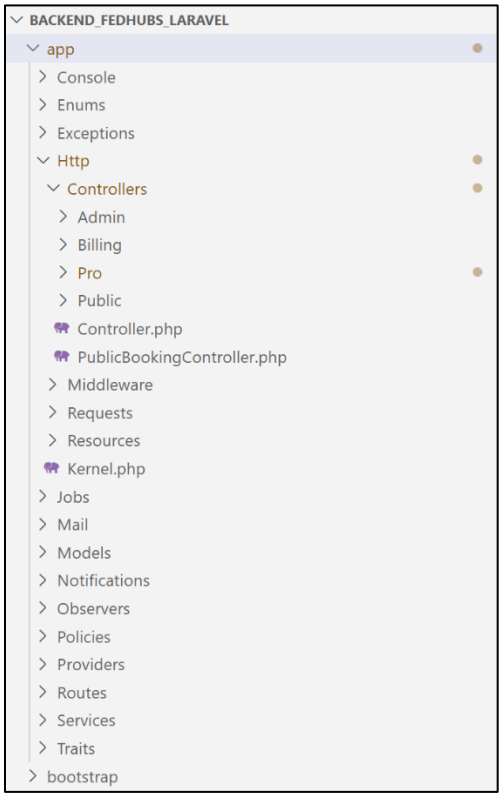
Toutes les informations concernant le projet étaient sur **Notion**, ça m'a permis de m'adapter très vite avec toutes les informations que j'avais à disposition bien avant le début du stage.

Voici le MEA sous format MLD :



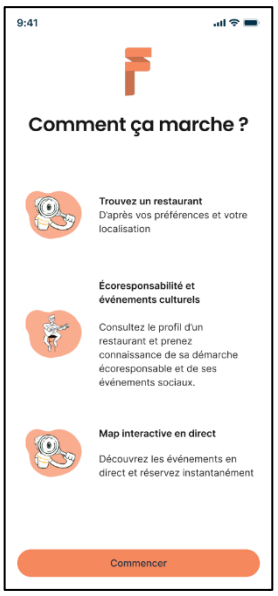
MALLE TRAORE
2eme année de BTS
SIO SLAM
A L'ENC PARIS

FEDHUBS



Voici l'architecture de l'application, c'est un projet Laravel.

Voici quelques captures d'écran de l'application (Figma)



B – Les missions réalisées

1 – Documentation des API

Je devais documenter les API qui concernaient la partie Public du projet. J'ai donc documenté avec **Swagger**, qui est un langage de description d'interface permettant de décrire des API exprimées à l'aide de JSON.

Pour cette fonction par exemple :

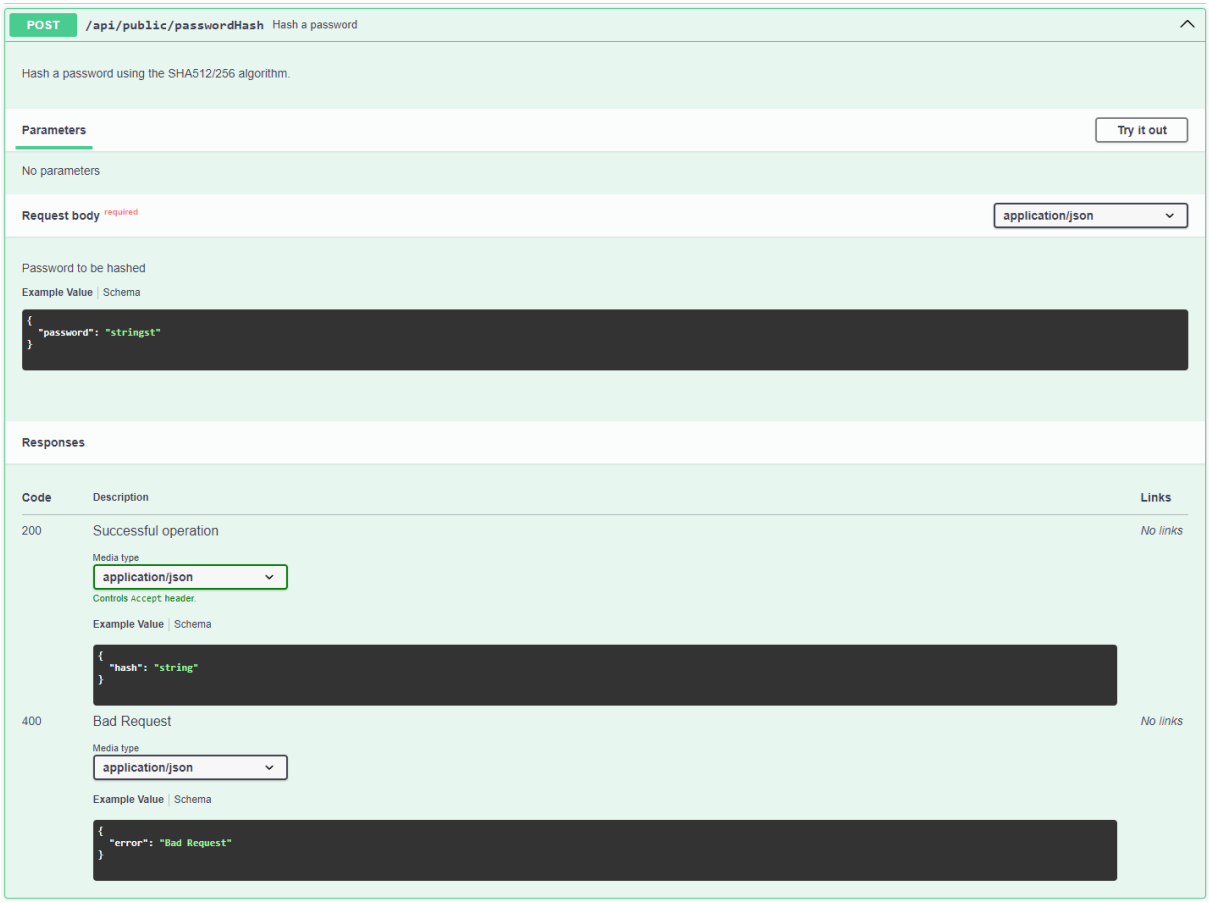
```php
public function passwordHash(Request $request)
{
    $data = $request->validate([
        "password" => ["required", "string", "min:8", "max:256"],
    ]);


    return response([
        "hash" => hash("sha512/256", $data['password']),
    ], 200);
}
```

Voici sa documentation Swagger :

```php
/**
 * @OA\Post(
 *     path="/api/public/passwordHash",
 *     summary="Hash a password",
 *     description="Hash a password using the SHA512/256 algorithm.",
 *     operationId="passwordHash",
 *     tags={"Public Service Management"},
 *     @OA\RequestBody(
 *         required=true,
 *         description="Password to be hashed",
 *         @OA\JsonContent(
 *             @OA\Property(property="password", type="string", minLength=8, maxLength=256, description="The password to be hashed"),
 *         ),
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Successful operation",
 *         @OA\JsonContent(
 *             @OA\Property(property="hash", type="string", description="Hashed password"),
 *         ),
 *     ),
 *     @OA\Response(
 *         response=400,
 *         description="Bad Request",
 *         @OA\JsonContent(
 *             @OA\Property(property="error", type="string", example="Bad Request"),
 *         ),
 *     ),
 * )
 */
```

Après avoir générer ma documentation, je peux me rendre sur le site de Swagger en local pour voir ce qu'il se passe :



2 – Réalisation des tests unitaires

J'ai eu l'occasion de faire plusieurs tests unitaires, voici l'un de mes test :

Pour cette partie de la fonction :

```php
public function register(RegisterRequest $request)
{
    $data = $request->validated();

    if (User::where('email', '=', $data["email"])->exists()) {
        return response(["message" => "Un compte est déjà associé à cette adresse mail."], 401);
    } // test réussi

    if (Tie_model_pseudonym::where('pseudonym', $data['pseudonym'])->exists()) {
        return $this->errorResponse('Pseudonym already exists');
    } // test réussi
```

Voici les tests :

```php
// Fonctionnelle
public function test_it_registers_with_existing_mail()
{
    // Créez un utilisateur existant dans la base de données
    $existingUser = User::factory()->create();

    // Envoyez une requête d'inscription avec la même adresse e-mail
    $response = $this->postJson('/api/auth/register', [
        'email' => $existingUser->email,
        'password' => 'password',
        'login_method_id' => 1,
        'isPro' => '0',
        'pseudonym' => 'example',
        'firstname' => 'John',
        'lastname' => 'Doe',
        'address' => '123 Main St',
        'phone' => '55512340000000',
        'token' => 'example-token',
        'user_id' => 1,
    ]);

    // Assurez-vous que la réponse contient le message d'erreur attendu
    $response->assertStatus(401)
        ->assertJson([
            'message' => 'Un compte est déjà associé à cette adresse mail.'
        ]);
}
```

```php
// Fonctionnelle
public function test_it_registers_with_existing_pseudonym()
{
    // Créez un utilisateur existant dans la base de données
    $existingPseudonym = Tie_model_pseudonym::create([
        'model_id' => 1,
        'pseudonym' => 'pseudo',
        'type' => 'company',
    ]);

    // Envoyez une requête d'inscription avec la même adresse e-mail
    $response = $this->postJson('/api/auth/register', [
        'email' => 'email@email.com',
        'password' => 'password',
        'login_method_id' => 1,
        'isPro' => '0',
        'pseudonym' => $existingPseudonym->pseudonym,
        'firstname' => 'John',
        'lastname' => 'Doe',
        'address' => '123 Main St',
        'phone' => '55512340000000',
        'token' => 'example-token',
        'user_id' => 1,
    ]);

    $response->assertStatus(400)
    ->assertJson([
        'status' => 'error',
        'message' => 'Pseudonym already exists',
        'data' => null
    ]);
}
```

J'exécute les tests avec la commande suivante : php artisan test

```
PS D:\wamp64\www\backend_fedhubs_laravel> php artisan test --filter RegisterUserTest
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\Http\v1\Public\User\RegisterUserTest
  ✓ it registers with existing mail
  ✓ it registers with existing pseudonym

  Tests:  2 passed
  Time:   5.49s
```

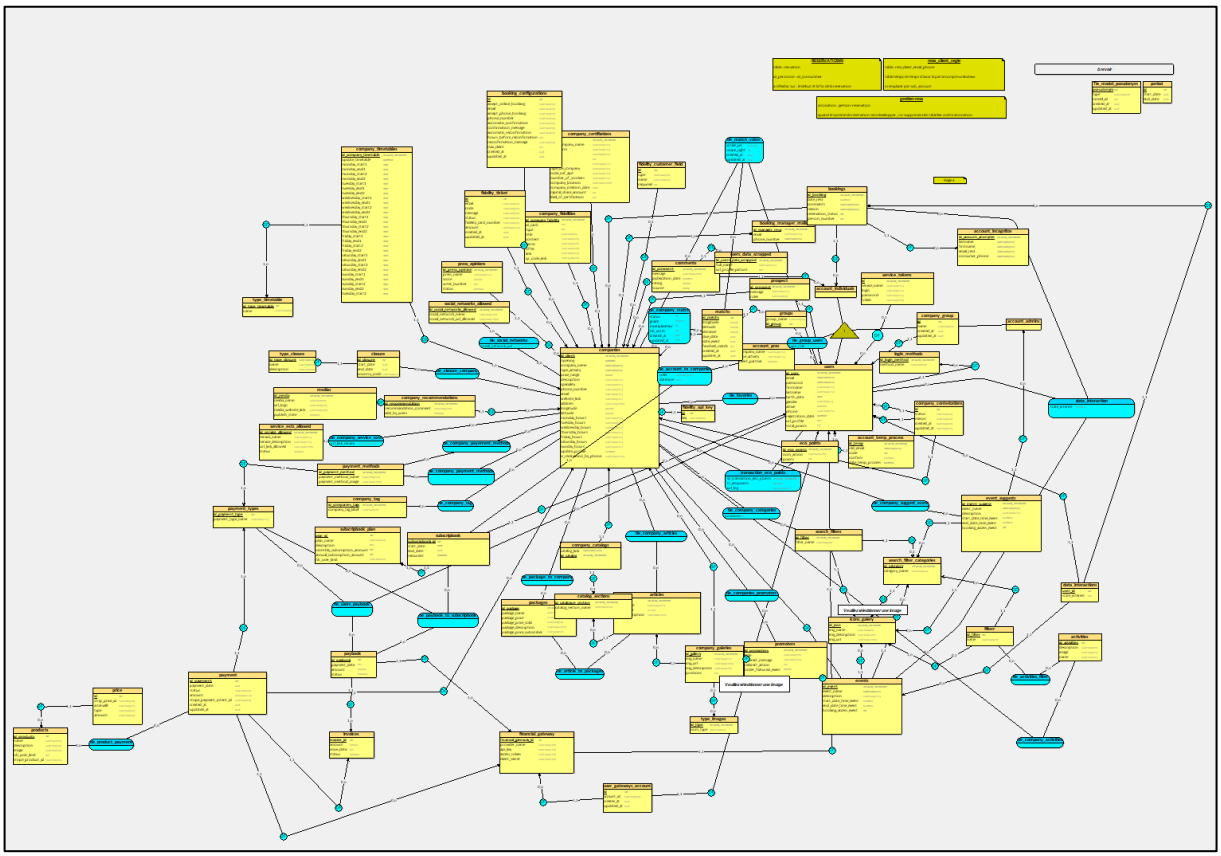
3 – Mise à jour du MEA

Le MEA qu'on avait à disposition n'était pas totalement à jour car il y'a eu quelques petites modifications par la suite. J'ai donc eu pour but de mettre à jour le MEA, avec les nouvelles tables qui ont été créée dans la base de données. Ce n'était pas compliqué car les tables qui n'étaient pas présente sur le MEA, étaient présente dans le répertoire Model de l'application.

MALLE TRAORE
2eme année de BTS
SIO SLAM
A L'ENC PARIS

FEDHUBS

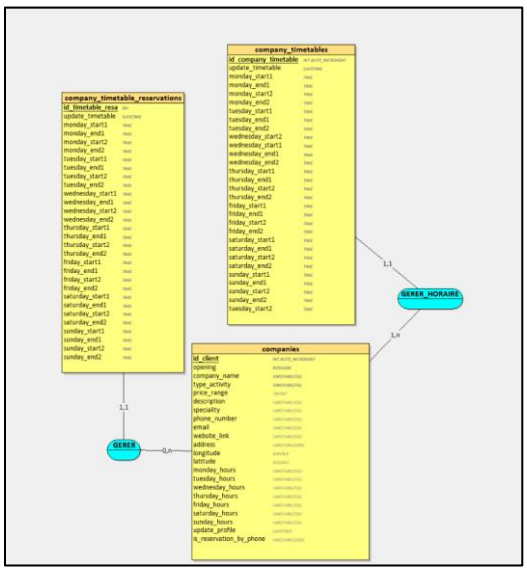
J'ai donc fait du repérage avec mon collègue, et nous avons reconstruit le MEA.
Le voici :



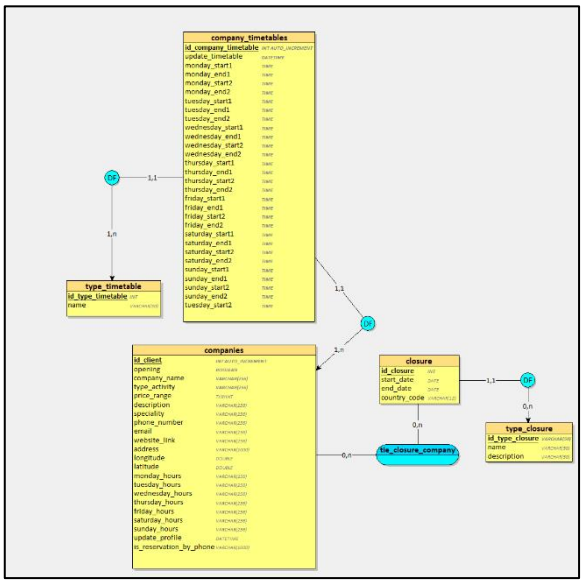
4 – Optimisation du code

Lors de notre précédente mission, nous avions fusionné deux entités pour optimiser la base de données.

Voici le MEA avant :



Et voici le MEA après :



MALLE TRAORE
2eme année de BTS
SIO SLAM
A L'ENC PARIS

FEDHUBS

Nous avions donc modifié le code pour que cela corresponde aux modifications que nous avions apportées au MEA.