

Parking

Le projet Parking consiste à créer une application web qui permet d'attribuer à chaque membre qui le demandait une place de parking numérotée.

Nous pouvons noter les besoins de cette manière :

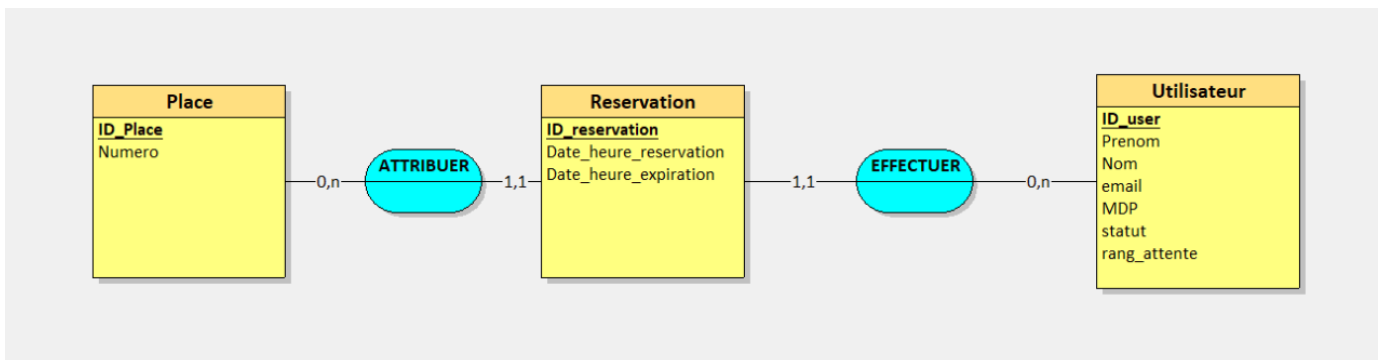
- ✚ Le front-office doit être sécurisé et n'accepter que les demandes du personnel des ligues. Les inscriptions au service de réservation de place doivent être validées (ou créées) par un administrateur.
- ✚ L'administrateur, seul utilisateur du back-office, doit pouvoir éditer la liste des places et gérer les inscriptions des utilisateurs.
- ✚ Lorsqu'un utilisateur en fait la demande, une place libre lui est attribuée aléatoirement et immédiatement par l'application, la réservation expire automatiquement au bout d'une durée par défaut déterminée par l'administrateur.
- ✚ Si une demande ne peut pas être satisfaite, l'utilisateur est placé en liste d'attente.
- ✚ L'utilisateur ne peut pas choisir la date à laquelle une place lui est attribué, les réservations sont toujours immédiates. Un utilisateur ne peut pas faire une demande de réservation s'il est en file d'attente ou qu'il occupe une place.
- ✚ Un utilisateur ou l'administrateur peuvent fermer une réservation avant la date d'expiration prévue. Une fois celle-ci expirée, l'utilisateur doit refaire une demande s'il souhaite obtenir une place.

Le but de cette itération ne porte que sur la documentation. C'est-à-dire nous avons pour objectif de réaliser :

- MCD
- Maquette pour l'application Web
- Plan du site avec URLs

Nous commençons à comprendre comment fonctionne GitHub.

- Voici notre MCD :

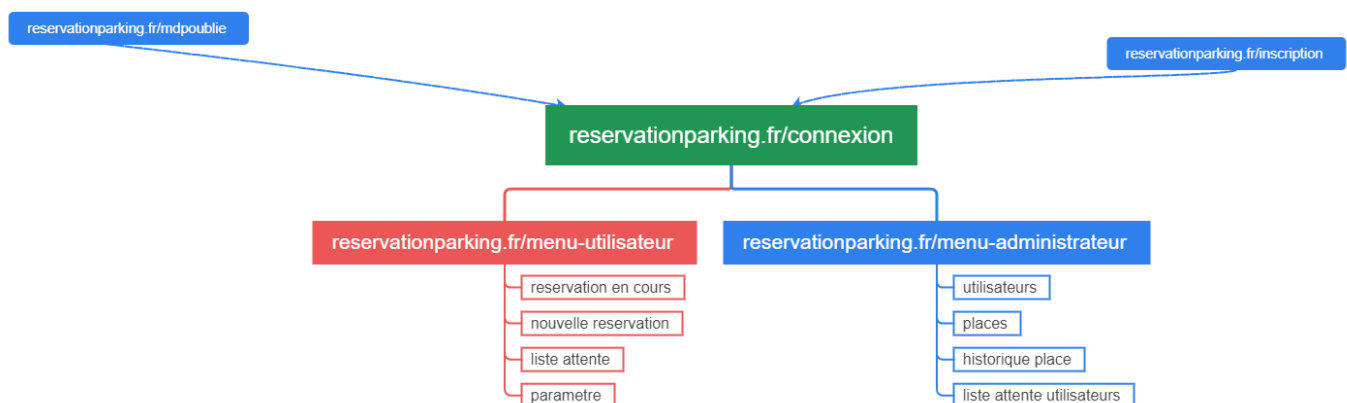


- Voici notre Maquette pour l'application web :

Disponible sur ce github

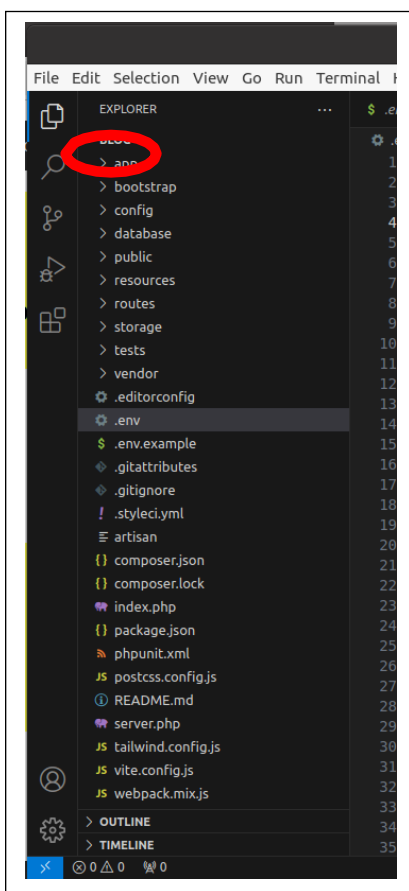
<https://github.com/thegoodone14/parking/tree/main/maquette>

- Voici Plan du site avec URLs :



Une fois les éléments de documentations effectués, on peut commencer à coder. Pour cela, il faut préparer l'environnement de développement en installant ou en ayant les éléments suivants :

- Machine Linux (préférence Ubuntu)
- Avoir PHP & Composer & MySQL
- Avoir Laravel
- Avoir un bon IDE : VSCode
- Avoir GitHub-Desktop



Le répertoire "BLOG" englobe l'intégralité de notre projet. Ce dossier est soumis à un Versionnage, et toute information que nous ne Souhaitons pas inclure dans le versionnage est spécifiée dans un fichier « .env ».

Un fois installé, nous devons apprendre ce qu'est Laravel et comment cela fonctionne.

Nous avons compris la théorie du système de routes, vues, model et de Controller. Mais cela fut complexe de gérer aux premiers abords.

Nous avons commencé par créer une page index.html :

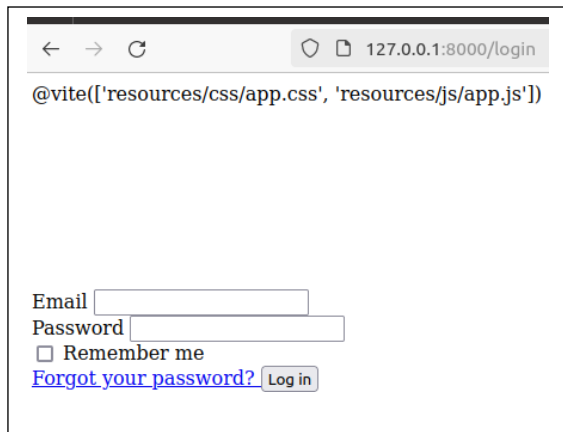
VUES :

Les vues servent à afficher les données d'une application web à l'utilisateur final, en fournissant une interface utilisateur.

Voici l'accueil de notre site codé en HTML & CSS se trouvant sur la vue « toto.blade.php » :

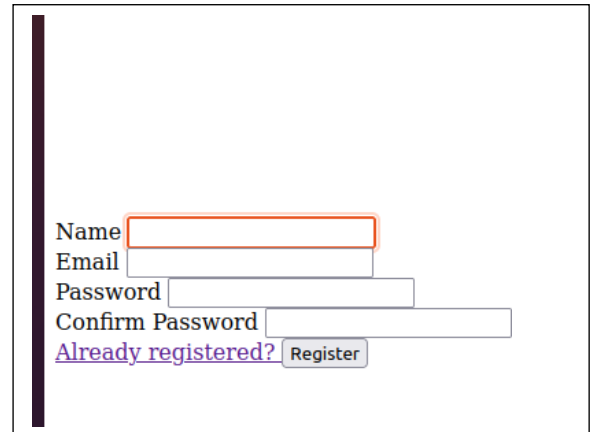


Pour notre page connexion, nous avons utilisé le package Breeze :



A screenshot of a web browser window showing a login page. The address bar displays the URL `127.0.0.1:8000/login`. The page content includes a Vite script at the top: `@vite(['resources/css/app.css', 'resources/js/app.js'])`. Below this, there are input fields for 'Email' and 'Password'. A checkbox labeled 'Remember me' is positioned below the password field. At the bottom left, there is a blue link that says 'Forgot your password?'. At the bottom right, there is a 'Log in' button.

Page de connexion



A screenshot of a web browser window showing a registration page. The page contains input fields for 'Name', 'Email', 'Password', and 'Confirm Password'. Below the 'Confirm Password' field, there is a purple link that says 'Already registered?'. To the right of this link is a 'Register' button.

Page d'inscription

ROUTES

Les routes dans Laravel sont comme des panneaux indicateurs sur une autoroute. Elles indiquent à Laravel quelle action doit être exécutée lorsque quelqu'un visite une URL spécifique sur votre site Web. Voici nos routes :

```
require __DIR__.'/auth.php';
Route::get('/menu_utilisateur', function () {
    return view('menu_utilisateur')
})->middleware(['auth'])->name('menu_utilisateur');

Route::middleware(['auth', 'is_admin'])->group(function () {
    Route::resource('places', PlaceController::class);
    Route::get('/admin/dashboard', function () {
        return view('admin.dashboard');
    })->name('admin.dashboard');

    Route::get('/admin/users', [AdminController::class, 'users'])->name('admin.users');
    Route::get('/admin/places', [AdminController::class, 'places'])->name('admin.places');
    Route::get('/admin/places/history', [AdminController::class, 'placesHistory'])->name('admin.places.history');
    Route::get('/admin/waitlist', [AdminController::class, 'waitlist'])->name('admin.waitlist');
});
```

```
// Route::middleware(['auth'])->group(function () {

    Route::get('/reservations/create', [ReservationController::class, 'create'])->name('reservations.create');

    Route::resource('reservations', ReservationController::class);
    // Toutes les autres routes qui nécessitent une authentification

    route::get('/menu-utilisateur', function () {
        return View('menu_utilisateur');
    });

    route::get('/menu-utilisateur/reservation-en-cours', function () {
        return View('reservation_en_cours');
    });

    route::get('/menu-utilisateur/nouvelle-reservation', function () {
        return View('nouvelle_reservation');
    });

    Route::get('/waitlist', [ReservationController::class, 'waitlist'])->name('waitlist');
    Route::get('/reservations/waitlist', [ReservationController::class, 'waitlist'])->name('reservations.waitlist');

    route::get('/menu-utilisateur/parametre', function () {
        return View('parametre');
    });
});
//});
```

CONTROLLEURS

Voici notre contrôleur, qui gère le processus de création de nouvelles réservations dans votre application. Elle assure également que les réservations antérieures sont nettoyées, évitant ainsi toute confusion ou problème avec les réservations expirées.

```
class ReservationController extends Controller
{
    // Enregistrer une nouvelle réservation dans la base de données
    public function store(Request $request)
    {
        // Récupérer l'ID de l'utilisateur authentifié
        $userID = auth()->user()->id;

        // Supprimer les réservations antérieures à la date actuelle
        Reservation::where('Fin_Reserv', '<', now())->delete();

        // Récupérer un ID_Place disponible
        $availablePlace = Place::whereNotExists(function ($query) {
            $query->select(DB::raw(1))
                ->from('reservations')
                ->whereColumn('reservations.place_id', 'places.id')
                ->orWhere('reservations.Fin_Reserv', '<', now());
        })->value('id');

        if (!$availablePlace) {
            // Aucune place disponible, rediriger vers la liste d'attente
            return ('error, Aucune place disponible pour effectuer la
réservation.');
```

DATABASE

Bien sûr nous devons créer notre base de données. Nous la nommerons « parking ». Nous créerons les tables grâce à la commande suivante :

php artisan make:migration nom_de_la_table

Ce dossier sera visible sur /database/migration/*nom du fichier*.php

```
<?php

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

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('prenom');
            $table->string('nom');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->boolean('statut')->default(0);
            $table->integer('rang_attente')->nullable();
            $table->rememberToken();
            $table->timestamps();
        });
    }

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


Nous avons également pu créer une page pour demander une nouvelle réservation, voir nos réservations en cours ou encore la liste d'attente avant qu'on nous attribue une place de parking et qui est coordonnée avec la base de données :



Page « Réservation en cours »

Réservations en cours

Date et heure de réservation	Date et heure d'expiration	Place
2024-03-27 15:26:20	2024-03-28 15:26:20	Place 1
2024-03-27 15:27:29	2024-03-28 15:27:29	Place 2
2024-03-27 15:27:34	2024-03-28 15:27:34	Place 3
2024-03-28 08:51:17	2024-03-28 08:53:17	Place 4

Page « Demander une nouvelle réservation »

Demander une nouvelle réservation

Cliquez sur le bouton ci-dessous pour lancer une demande de réservation :

Demander une réservation

Reçu de notre demande :

