Extraits de code commentés

Clickandeat

1Connexion :

1.1 Connexion

```php
/**
 * Traite la connexion de l'utilisateur
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\RedirectResponse
 */
public function login(Request $request)
{
    // Valide les champs email et mot de passe
    $credentials = $request->validate([
        'email' => ['required', 'email'],
        'password' => ['required'],
    ]);

    // Tente d'authentifier l'utilisateur avec les identifiants fournis
    if (Auth::attempt($credentials)) {
        // Régénère la session pour éviter les attaques de fixation de session
        $request->session()->regenerate();

        $user = Auth::user();
        // Redirige selon le rôle de l'utilisateur
        if ($user->role === 'restaurateur') {
            return redirect('/restaurateur/dashboard');
        }
        // Redirection par défaut (client)
        return redirect('/client/dashboard');
    }

    // Retourne à la page précédente avec un message d'erreur si l'authentification échoue
    return back()->withErrors([
        'email' => 'Les identifiants fournis ne correspondent pas à nos enregistrements.',
    ])->onlyInput('email');
}
```

1.2 Inscription

```php
    public function create()
    {
        return view('auth.register');
    }

    /**
     * Traite l'inscription d'un nouvel utilisateur
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\RedirectResponse
     */
    public function store(Request $request)
    {
        // Valide les champs du formulaire d'inscription
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'confirmed', Rules\Password::defaults()],
            'role' => ['required', 'string', 'in:client,restaurateur'],
        ]);

        // Crée un nouvel utilisateur avec les données validées
        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password), // Hash le mot de passe
            'role' => $request->role,
        ]);

        // Déclenche l'événement Registered (pour l'email de bienvenue, etc.)
        event(new Registered($user));

        // Connecte automatiquement l'utilisateur après l'inscription
        Auth::login($user);

        // Redirige selon le rôle sélectionné
        if ($request->role === 'restaurateur') {
            return redirect('/restaurateur/dashboard');
        }

        // Redirection par défaut (client)
        return redirect('/client/dashboard');
    }
}
```

1.3 Changement de mot de passe

```php
public function create(Request $request): View
{
    return view('auth.reset-password', ['request' => $request]);
}

/**
 * Traite la demande de nouveau mot de passe.
 *
 * Cette méthode est appelée lorsque l'utilisateur soumet le formulaire de réinitialisation du mot de passe.
 * Elle valide les champs du formulaire, réinitialise le mot de passe de l'utilisateur et redirige vers la page de connexion.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\RedirectResponse
 * @throws \Illuminate\Validation\ValidationException
 */
public function store(Request $request): RedirectResponse
{
    // Valide les champs du formulaire
    // Les champs 'token', 'email' et 'password' sont requis
    // Le champ 'email' doit être un email valide
    // Le champ 'password' doit être confirmé et doit respecter les règles de mot de passe par défaut
    $request->validate([
        'token' => ['required'],
        'email' => ['required', 'email'],
        'password' => ['required', 'confirmed', Rules\Password::defaults()],
    ]);

    // Logique de réinitialisation du mot de passe
    // On utilise la facade Password pour réinitialiser le mot de passe de l'utilisateur
    // On passe les champs 'email', 'password', 'password_confirmation' et 'token' en paramètre
    // On utilise une fonction anonyme pour mettre à jour le mot de passe de l'utilisateur et déclencher l'événement PasswordReset
    $status = Password::reset(
        $request->only('email', 'password', 'password_confirmation', 'token'),
        function (User $user) use ($request) {
            // On met à jour le mot de passe de l'utilisateur
            $user->forceFill([
                'password' => Hash::make($request->password),
                'remember_token' => Str::random(60),
            ])->save();

            // On déclenche l'événement PasswordReset
            event(new PasswordReset($user));
        }
    );

    // Redirige vers la page de connexion avec un message de succès ou d'erreur
    // Si la réinitialisation du mot de passe a réussi, on redirige vers la page de connexion avec un message de succès
    // Sinon, on redirige vers la page précédente avec un message d'erreur
    return $status == Password::PASSWORD_RESET
                ? redirect()->route('login')->with('status', __($status))
                : back()->withInput($request->only('email'))
                    ->withErrors(['email' => __($status)]);
}
```

2 Utilisateur :

2.1 Visuel utilisateur

```php
class DashboardController extends Controller
{
    /**
     * Affiche la page du tableau de bord en fonction du rôle de l'utilisateur.
     *
     * Cette méthode vérifie le rôle de l'utilisateur connecté et affiche la vue correspondante.
     *
     * @return \Illuminate\View\View
     */
    public function index()
    {
        // Récupère l'utilisateur connecté
        $user = Auth::user(); // Récupérer l'utilisateur actuellement authentifié

        // Récupère les commandes récentes de l'utilisateur
        $orders = Order::where('user_id', $user->id)->latest()->take(5)->get();

        // Récupère les réservations récentes de l'utilisateur
        $reservations = Reservation::where('user_id', $user->id)->latest()->take(5)->get();

        // Vérification du rôle de l'utilisateur
        if ($user->isAdmin()) {
            // Si l'utilisateur est un admin, afficher la vue pour l'admin
            // avec les données des commandes et réservations récentes
            return view('dashboard.admin', compact('user', 'orders', 'reservations'));
        } elseif ($user->isRestaurateur()) {
            // Si l'utilisateur est un restaurateur, afficher la vue pour le restaurateur
            // avec les données des commandes et réservations récentes
            return view('dashboard.restaurateur', compact('user', 'orders', 'reservations'));
        } elseif ($user->isClient()) {
            // Si l'utilisateur est un client, afficher la vue pour le client
            // avec les données des commandes et réservations récentes
            return view('dashboard.client', compact('user', 'orders', 'reservations'));
        }

        // Si aucun rôle ne correspond, rediriger ou afficher une page d'erreur
        return redirect('/')->with('error', 'Accès interdit');
    }
}
```

2.2 Affichage de la page des restaurants

```php
class RestaurantsController extends Controller
{
    /**
     * Affiche la liste de tous les restaurants
     * @return \Illuminate\View\View
     */
    public function index()
    {
        // Récupère tous les restaurants avec leur utilisateur associé
        $restaurants = Restaurant::with('user')->get();
        return view("restaurants.index", ['restaurants' => $restaurants]);
    }
```

2.3 Affichage des détails d'un restaurant

```php
public function show($id)
{
    // Récupère le restaurant avec son utilisateur associé
    $restaurant = Restaurant::with('user')->findOrFail($id);
    return view('restaurants.show', ['restaurant' => $restaurant]);
}
```

2.4 Formulaire de modification des restaurants

```php
public function edit($id)
{
    // Récupère le restaurant à éditer
    $restaurant = Restaurant::findOrFail($id);
    // Récupère tous les utilisateurs ayant le rôle restaurateur pour lier le restaurant
    $users = User::where('role', 'restaurateur')->get();
    return view('restaurants.edit', ['restaurant' => $restaurant, 'users' => $users]);
}
```

3 Items

3.1 Récupération de tous les plats

```php
public function index()
{
    // Récupère tous les plats avec pagination et la relation category
    $items = Item::with('category')->paginate(10);
    return view('items.index', compact('items'));
}
```

3.2 Création d'un nouveau plat

```php
public function create()
{
    // Récupère toutes les catégories pour le formulaire de création
    $categories = Category::all();
    return view('items.create', compact('categories'));
}
```

3.3 Affichage du formulaire :

```php
/**
 * Affiche le formulaire d'édition d'un plat.
 *
 * @param  \App\Models\Item  $item
 * @return \Illuminate\Http\Response
 */
public function edit(Item $item)
{
    // Récupère toutes les catégories pour le formulaire d'édition
    $categories = Category::all();
    return view('items.edit', compact('item', 'categories'));
}
```

4 Payments

4.1 Récupération des commandes

```php
public function show($orderId)
{
    // Récupère la commande de l'utilisateur connecté
    $order = Order::where('user_id', Auth::id())->findOrFail($orderId);
    return view('payment.stripe', compact('order'));
}
```

4.2 Affichage de la page

```php
public function show($orderId)
{
    // Récupère la commande de l'utilisateur connecté
    $order = Order::where('user_id', Auth::id())->findOrFail($orderId);
    return view('payment.stripe', compact('order'));
}
```

4.3 Lancement du payment

```php
public function checkout(Request $request, $orderId)
{
    // Récupère la commande de l'utilisateur connecté
    $order = Order::where('user_id', Auth::id())->findOrFail($orderId);

    // Initialise la clé secrète Stripe
    Stripe::setApiKey(config('services.stripe.secret'));

    // Crée une session Stripe Checkout
    $session = StripeSession::create([
        'payment_method_types' => ['card'],
        'line_items' => [[
            'price_data' => [
                'currency' => 'eur',
                'unit_amount' => (int)($order->total_amount * 100),
                'product_data' => [
                    'name' => 'Commande #' . $order->id,
                ],
            ],
            'quantity' => 1,
        ]],
        'mode' => 'payment',
        'success_url' => route('payment.success', $order->id),
        'cancel_url' => route('payment.cancel', $order->id),
        'customer_email' => Auth::user()->email,
    ]);

    // Redirige l'utilisateur vers la page de paiement Stripe
    return redirect($session->url);
}
```