**Instrucciones para iniciar el proyecto:**

**Página web principal:**

http://127.0.0.1:8000/

**Cosas a tener ejecutadas en el cuadro de comandos mientras trabajo para que funcione el proyecto:**

Servidor: **Php artisan serve**

Poner en funcionamiento el envío de emails en cola:

**php artisan queue:work --queue=default --tries=3**

Ver estilos de la hoja de estilos app.css (sin esto corriendo se verá la web sin estilos):

**npm run dev**

**Enlaces a visitar por el gerente para gestionar todo:**

Para añadir, editar o eliminar los platos de la carta:

<http://127.0.0.1:8000/platos/editar>

Para ver las reservas:

<http://127.0.0.1:8000/gerencia/reservas>

**Mapa de archivos proyecto Laravel**

Estructura general de la página:

-Routes/web.php

-resources/views/layout.blade.php //Esta creo que es como una plantilla que usan las 4 siguientes

-resources/views/inicio.blade.php

-resources/views/menu.blade.php

-resources/views/reservas.blade.php

-resources/views/contacto.blade.php

**Para la sección de contacto:**

**0).env**

Para editar la configuración del email

**1) routes/web.php (Rutas)**

Define por dónde entra la petición (GET para ver el formulario, POST para enviarlo).

Mantendremos tu ruta GET y añadiremos una POST.

Opcional: en vez de un closure, usar un Controlador nos da orden y escalabilidad.

**2)app/Http/Controllers/ContactController.php (Controlador)**

Contiene la lógica cuando llega el POST: recibe datos, delega la validación (Form Request), y decide qué responder (redirigir con éxito o con errores).

**3)app/Http/Requests/StoreContactRequest.php (Form Request)**

Validación en servidor de los datos del formulario. Así, aunque alguien intente saltarse la validación del navegador, el backend sigue protegiendo.

**4)App/Mail/ContactMessage.php**

Define cómo se construye el email: asunto, vista, datos, etc

**5)Ressources/views/emails/contacto.blade.php**

Es la vista del email. Es decir, la plantilla con el mensaje que recibiré.

**6)resources/views/contacto.blade.php (Vista)**

Es la vista del formulario en la pagina web.

Añadimos validación del lado del cliente (HTML5: required, minlength, maxlength, type="email", pattern para el teléfono) y un poquito de JS (contador de caracteres, bloqueo de doble envío).

**Estructura de la sección de reservas**

**1) routes/web.php**

**•** Archivo de rutas web del proyecto.

• Define rutas para /, /menu, /reservas (GET y POST) y /contacto.

• Incluye validación de datos y creación de reservas

**2)App/Http/Controllers/ReservaController.php**

• Define la lógica de creación de la reserva

**3)app/Models/Reserva.php**

• Modelo Eloquent para la tabla reservas.

• Define los campos que pueden asignarse de forma masiva ($fillable).

**4)****database/migrations/2025\_08\_164036\_create\_reservas\_table.php**

• Migración que crea la tabla reservas en la base de datos.

• Define columnas (fecha, hora, personas, etc.) y métodos up()/down().

**5)resources/views/reservas.blade.php**

• Vista Blade para el formulario de reserva.

• Muestramensajes de éxito y el formulario con selectores para fecha, personas y hora.

**6)App/Mail/ReservaConfirmada.php**

• Archivos de configuración para el envío automático del email:

**7)resources\views\emails/reserva\_confirmada.blade.php:**

la pantilla markdown del email

**Estructura de la sección menu:**

**M (Modelo)**

**1)app/Models/Plato.php**

Representa la **entidad de dominio** “Plato”.

**2)database/migrations/2025\_08\_16\_135645\_create\_platos\_table.php**

Define la **forma** de la tabla platos (nombre, precio, imagen\_path…).

**V (Vistas)**

**3)****resources/views/layout.blade.php**

Es tu **plantilla base** (layout). evita duplicar HTML en todas las páginas.

**4)resources/views/menu.blade.php**

Vista del **formulario** para crear platos (uso interno).

**C (Controlador)**

**5)app/Http/Controllers/PlatoController.php**

**Enrutado (no es parte de MVC, pero es el “pegamento”)**

**6)routes/web.php**

**Sección para el gerente para editar la carta:**

**7)resources/views/platos/create.blade.php**

**8)resources/views/platos/edit.blade.php**

**9)resources/views/platos/manage.blade.php** (es la que nos muestra la vista platos/editar)

**Idea clave (el “por qué” de todo esto)**

* **Modelo**: habla con la BD y representa tus datos.
* **Vista**: solo pinta.
* **Controlador**: traduce una petición HTTP en acciones sobre modelos y respuestas (vistas/redirects).
* **Rutas**: mapa URL → Controlador.
* **Migraciones**: versionan la estructura de tu BD (infraestructura).
* **Storage**: separa archivos de la carpeta pública y los sirve correctamente.

\*Con esto ya estás trabajando auténtico MVC en Laravel. Si quieres, en el siguiente paso te enseño a refactorizar validaciones a un FormRequest (todavía más limpio) y a extraer la lógica de subida de imagen a un método del Modelo para dejar el controlador aún más fino. ¿Te apetece?

**Archivos de la vista de reservas para gerencia**

-1)Routes/web.php

**Específicos para esta área:**

-2)Config/reservas.php

-3)app/Http/Controllers/GerenciaReservasController.php

**-4)resources/views/gerencia/reservas/index.blade.php:**

Vista interna para ver las reservas

**Creados cuando se creó la sección de reservas (osea ya estaba previamente creado):**

-5)database/migrations/2025\_08\_164036\_create\_reservas\_table.php

Esto lo puedo mejorar quizás haciendo que está sección esté oculta o protegida, para que no pueda entrar a esta url cualquiera. Lo mismo para la sección de añadir o editar platos del menu (esa es aún mas peligrosa porque da acceso a desbaratarte el menu).

**Archivos para la sección de inicio**

1) Routes/Web.php

2) resources/views/layout.blade.php

3) resources/views/inicio.blade.php

**Usuarios:**

* **Email**: hectorortizsuarez3@gmail.com
* **Password**: 123456 (encriptada con bcrypt).