Documentación Asambleas – Giramaster - Backend

En la presente documentación se encuentra la explicación de cada funcionalidad del proyecto presentado en la carpeta *ASAMBLEAS-GIRAMASTER-MAIN,* con el objetivo de identificar el funcionamiento de la aplicación y los alineamientos con los que está construida.

El backend está construido con **Laravel**, aprovechando su potente sistema de rutas, controladores, middleware y Eloquent ORM para facilitar la gestión de usuarios, reuniones, votaciones y reportes. Se emplean varias funcionalidades integradas como la autenticación, el envío de correos electrónicos, la exportación de archivos y la seguridad mediante el uso de hash en contraseñas. Además, la aplicación parece estar diseñada para interactuar con servicios externos, como Zoom, lo que permite un alto grado de integración y flexibilidad en su funcionamiento.

Índice

Documentación backend: En la carpeta backend se encuentra todo lo relacionado con la creación de la base de datos mediante controladores de php y además de esto la conexión con la API, en el backend encontramos carpetas tales como:

1. **Carpeta App:** Esta carpeta contiene…

   1.1 **Console:** Contiene Scripts ejecutables desde consola como comandos personalizados o tarjetas programadas (cron jobs)

   - **Kernel.php:** Este archivo, Kernel.php, es el núcleo para la consola de comandos en aplicaciones Laravel. Se encarga de gestionar los comandos Artisan y la programación de tareas automáticas.

   1.2 **Exceptions:** Manejo de excepciones personalizadas para gestionar errores específicos del sistema.

   - **Handler.php:** se encarga de centralizar el manejo de excepciones en la aplicación, definiendo qué errores registrar, qué datos proteger, y cómo presentarlos al usuario final. Es el núcleo del manejo de errores en Laravel.

   1.3 **Exports:** Clases para exportar datos, por ejemplo, en formatos como Excel, CSV o PDF.

   - **PlantillaExport.php:** Este código automatiza la generación de reportes en Excel con datos dinámicos. Aplica formatos y asegura que el archivo tenga una presentación profesional y organizada. Ideal para exportar información de usuarios o registros.

1.4 HTTP: Esta carpeta contiene...

1.4.1 Controllers: Manejo de rutas HTTP y de endpoints. Esta carpeta tiene otras dos carpetas en la carpeta API encontramos.

- **AuthController.php:** Define el controlador para gestionar funciones relacionadas con la autenticación y roles en una aplicación Laravel.

- **PoderesController.php:** Define el controlador para los poderes o acciones que tienen los usuarios dentro del aplicativo permitiendo la asignación de apoderados para representar a los propietarios y realizar votaciones o decisiones en su nombre.
- **ReportesController.php:** Este controlador en Laravel que maneja la lógica de generación de reportes de datos relacionados con "EmpresasPersonas".
- **ReunionesController.php:** Este controlador maneja las acciones relacionadas con la gestión de reuniones, como la creación, actualización, eliminación, y el manejo de datos relacionados, como votaciones, quorum y usuarios de la reunión.
- **UnidadesController.php:** Este controlador es el responsable de manejar las operaciones CRUD (crear, leer, actualizar, eliminar) relacionadas con las **unidades** dentro del sistema. Este controlador se encarga de recibir y procesar las solicitudes HTTP (GET, POST, PUT, DELETE) relacionadas con las unidades, las cuales son entidades clave en la aplicación. Además de las operaciones estándar de CRUD, también gestiona la subida y eliminación de imágenes asociadas a las unidades, y el manejo de los usuarios y roles vinculados.
- **UsuariosController.php:** Este controlador gestiona las operaciones relacionadas con los usuarios del sistema. Las operaciones incluyen la creación, actualización, eliminación, y el manejo de contraseñas.
- **ValidacionesController.php:** Este controlador maneja las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para la entidad **"Validaciones"** en tu aplicación.
- **VotacionesController.php:** Este controlador se encarga de gestionar las operaciones relacionadas con las votaciones en el sistema. Este controlador permite crear, gestionar, cerrar y eliminar votaciones, así como gestionar las preguntas y las opciones asociadas a las votaciones.
- **ZoomController.php:** Este controlador tiene la función principal de generar una **firma JWT** que es utilizada para autenticar a un usuario en una reunión de Zoom, utilizando las credenciales del SDK de Zoom.

1.4.2 Auth: Contienen controladores relacionados con el manejo de la autenticación y gestión de usuarios en el sistema.

- **ForgotPasswordController:** Este controlador se encarga de manejar el proceso de restablecimiento de contraseñas mediante el envío de correos electrónicos. Utiliza el trait `SendsPasswordResetEmails` para facilitar la implementación de esta funcionalidad, y está protegido por un middleware que restringe su uso únicamente a usuarios invitados (no autenticados).
- **LoginController.**php: Este controlador es responsable de manejar la autenticación de usuarios en la aplicación. Usa el trait `AuthenticatesUsers`, que incluye métodos para iniciar sesión, validar credenciales y gestionar redirecciones. Después de un inicio de sesión exitoso, redirige a los usuarios a la ruta `/home`. Aplica un middleware que permite el acceso únicamente a usuarios no autenticados, excepto para la funcionalidad de cierre de sesión.
- **RegisterController.**php: Este controlador maneja el registro de nuevos usuarios en la aplicación. Utiliza el trait `RegistersUsers` para implementar métodos estándar de registro. Este controlador valida los datos de entrada mediante reglas de validación, asegura que las contraseñas estén encriptadas antes de guardarlas en la base de datos, y redirige a los usuarios a `/home` tras un registro exitoso. Aplica el middleware `guest` para limitar el acceso solo a usuarios no autenticados.
- **ResetPasswordController.**php: Este controlador maneja las solicitudes de restablecimiento de contraseñas. Utiliza el trait ResetsPasswords, que incluye métodos estándar para esta funcionalidad. Tras un restablecimiento exitoso, los usuarios son redirigidos a la ruta /home. Además, el middleware guest asegura que solo los usuarios no autenticados puedan interactuar con este controlador.
- **VerificationController.**php: Este controlador gestiona la verificación de correos electrónicos para usuarios recién registrados. Usa el trait VerifiesEmails para proporcionar métodos predefinidos, como la confirmación de verificación y el reenvío de correos. Incluye middleware para garantizar que: Solo usuarios autenticados puedan verificar sus correos, las solicitudes de verificación estén firmadas, las solicitudes de verificación y reenvío estén limitadas para prevenir abuso. Tras la verificación, los usuarios son redirigidos a /home.

- **Controller.**php: El archivo Controller.php sirve como controlador base para todos los demás controladores en la aplicación. Proporciona funcionalidades comunes mediante los *traits* de Laravel: autorización, validación de solicitudes y manejo de trabajos en cola. Este archivo es el punto de partida para extender la lógica de los controladores específicos.

1.4.3 Middleware: Contiene archivos que son responsables de manejar ciertas acciones o verificaciones antes o después de que se ejecute el código de los controladores. Los *middlewares* son filtros que permiten verificar si el usuario tiene permisos adecuados, si se encuentran en el modo de mantenimiento, si la solicitud es válida, entre otros aspectos.

- **Authenticate.php:** Este middleware se utiliza para proteger rutas que requieren que el usuario esté autenticado. Si el usuario no está autenticado, no se le redirige a una vista, sino que se le responde con un mensaje JSON indicando que no tiene acceso.
- **CheckForMaintenanceMode.php:** Este middleware es usado para habilitar un modo de mantenimiento para la aplicación. Cuando la aplicación está en mantenimiento, generalmente, las solicitudes a todas las rutas son bloqueadas y los usuarios ven una página de mantenimiento. Sin embargo, si alguna ruta está definida en el array $except, esa ruta se mantendrá accesible, incluso durante el mantenimiento.
- **Cors.php:** Este middleware maneja las solicitudes CORS (Cross-Origin Resource Sharing). CORS es un mecanismo que permite a los navegadores realizar solicitudes de un dominio a otro (entre diferentes orígenes). Laravel no habilita CORS de manera predeterminada, por lo que este middleware lo implementa para permitir que las solicitudes de recursos (como una API) sean compartidas entre orígenes diferentes.
- **EncryptCookies.php:** Este middleware ayuda a asegurar la privacidad de las cookies en tu aplicación, encriptándolas antes de enviarlas al cliente y desencriptándolas cuando se reciben en solicitudes posteriores.
- **RedirectIfAuthenticated.php:** Este middleware hará que los usuarios que ya estén autenticados sean redirigidos a /home en lugar de ver las páginas de login o registro.
- **TrimStrings.php:** Este middleware ayuda a mantener la integridad de los datos eliminando los espacios innecesarios de las cadenas, asegurando que los datos recibidos sean más consistentes y evitando errores comunes que puedan surgir por los espacios extra en los formularios de los usuarios.
- **TrustProxies.php:** Este middleware asegura que Laravel pueda manejar adecuadamente las cabeceras de los proxies, lo que es especialmente importante en aplicaciones que están detrás de un proxy inverso o balanceador de carga. Permite que Laravel obtenga correctamente la IP del cliente real y otros detalles sobre la solicitud, a pesar de que el tráfico esté siendo redirigido a través de un proxy o balanceador de carga.
- **VerifyCsrfToken.php:** Este middleware ayuda a proteger tu aplicación de ataques CSRF, asegurando que todas las solicitudes a tu servidor provengan de usuarios legítimos. La cookie XSRF-TOKEN facilita el proceso, permitiendo que las solicitudes incluyan un token de verificación. Solo se deben excluir de esta verificación las rutas que realmente lo necesiten, como en el caso de APIs o servicios externos.

- **Kernel.php:** Configura cómo los middleware de tu aplicación Laravel gestionan las solicitudes HTTP. Define un conjunto de middleware global, grupos para rutas web y de API, y middleware que se pueden asignar a rutas específicas. Además, el archivo establece el orden de ejecución de los middleware no globales. Estos middleware son fundamentales para gestionar sesiones, autenticación, protección CSRF, control de acceso, y mucho más, mejorando la seguridad y el rendimiento de la aplicación.

1.5 Imports: Esta carpeta almacena la clase encargada de realizar las importaciones de los usuarios al sistema.

- **UsersImport.php:** Este archivo es una **clase de importación** en Laravel que se encarga de procesar datos desde un archivo Excel (o archivo con datos tabulados) y almacenarlos en la base de datos. El archivo implementa las interfaces ToCollection y WithHeadingRow, que indican que el archivo se va a leer como una colección de filas y que la primera fila del archivo contiene los encabezados (nombres de las columnas).

1.6 Mail: Esta carpeta contiene las clases para gestionar el envío de correos electrónicos personalizados, como notificaciones, invitaciones o reportes.

- **CorreoLibre.php:** Este archivo es responsable de generar un correo electrónico utilizando el sistema de correo de Laravel. Este correo tiene como propósito notificar a los usuarios sobre una reunión, con información personalizada basada en los datos que se le pasan al constructor de la clase. Utiliza la vista emails.notificacion-asamblea para estructurar el contenido del correo.
- **InvitacionAsamblea.php:** Este archivo es el encargado de gestionar la creación de un correo de invitación a una asamblea. El correo está diseñado para ser enviado a los usuarios, con información sobre la asamblea y datos de acceso personalizados.
- **NotificacionEntrevista.php:** Este archivo contiene información sobre la entrevista que se va a realizar y está destinado a ser enviado a una persona específica, incluyendo detalles sobre la entrevista y posiblemente sobre la ubicación de la misma.
- **NotificacionLlamada.php:** Este archivo es el que envía a una persona especifica un correo para informarle sobre la llamada.
- **NotificacionVisita.php:** Este archivo envía la notificación a una persona con los detalles de la visita domiciliaria.
- **PoderDenegado.php:** Este archivo es responsable de gestionar el envío de un correo que notifica a un usuario sobre la denegación de su solicitud de poder. Este correo incluye el asunto, la dirección de correo de destino y el motivo de la denegación.

1.7 Models: Esta carpeta contiene los diversos modelos de datos que interactúan con bases de datos en el contexto de un sistema de gestión de reuniones, votaciones, usuarios, y otras entidades relacionadas con la administración de una plataforma.

- **Apoderado.php:** Este archivo define un modelo Eloquent para interactuar con la tabla adm_apoderado_copro en la base de datos. Este modelo permite gestionar los apoderados y sus datos asociados, como el nombre, cédula y correo electrónico. Además, establece una relación "belongsTo" con el modelo UsuariosReuniones, lo que implica que cada apoderado está vinculado a un usuario de reunión específico. Esto facilita la recuperación de datos relacionados entre ambas tablas de manera eficiente y segura. El archivo también asegura la protección contra la asignación masiva al especificar explícitamente qué campos son asignables de manera masiva.
- **CorreosEnviados.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_correos_enviados en la base de datos. Este modelo permite gestionar los correos enviados, almacenando información como el tipo de correo, asunto, mensaje, receptor y el identificador de la reunión asociada. Los campos definidos en la propiedad $fillable permiten la asignación masiva de valores a estos atributos de forma segura. Este archivo facilita la manipulación de los registros de correos enviados en la base de datos, permitiendo consultas y operaciones relacionadas con este modelo.
- **DetalleQuorum.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_detalle_quorum en la base de datos. Este modelo gestiona la información relacionada con los detalles del quórum en las reuniones, como la identificación del quórum, el usuario de la reunión, el apoderado, las fechas y horas de ingreso y salida, y la información de conexión. Además, define dos relaciones con otros modelos: Quorum y UsuariosReuniones, lo que permite acceder a los detalles del quórum y los usuarios asociados de forma fácil. Este archivo facilita la manipulación de registros detallados del quórum en la base de datos.
- **DocPoderes.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_doc_poderes en la base de datos. Este modelo gestiona la información relacionada con los documentos de poderes, como el ID del poder, el nombre del archivo, la extensión, la ruta del archivo en el sistema y su tamaño. Este archivo facilita la manipulación de registros de documentos de poderes en la base de datos, permitiendo el acceso y modificación de los datos de manera sencilla.
- **InformeWpp.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_wpp_enviados en la base de datos. Este modelo maneja la información relacionada con los informes enviados a través de WhatsApp (WPP), almacenando la fecha de envío, el ID del usuario que envió el informe, y el ID de la reunión asociada. Proporciona una manera fácil de acceder y manipular los registros de estos informes en la base de datos.
- **Logos.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_logos_unidades en la base de datos. Este modelo gestiona la información de los logotipos asociados a las unidades, almacenando el ID de la unidad, el nombre

del archivo del logo, su extensión, la ruta del archivo y su tamaño. Proporciona una forma estructurada de acceder y manipular estos datos dentro de la base de datos.

- **OpcionesRespuestas.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_opciones_respuesta en la base de datos. Este modelo gestiona las opciones de respuesta relacionadas con preguntas de votación, incluyendo detalles como la opción de respuesta, el nombre de la imagen asociada, la extensión y la ruta del archivo. Además, establece una relación de uno a muchos con el modelo RespuestasVotaciones, lo que permite acceder fácilmente a todas las respuestas asociadas a una opción específica.
- **Poderes.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_poderes en la base de datos. Este modelo representa los poderes otorgados a un usuario para una reunión específica, incluyendo detalles como el estado, la cédula, correo, apartamento, y si el usuario es propietario o apoderado. Además, establece varias relaciones con otros modelos: un poder pertenece a una reunión de usuario (UsuariosReuniones), puede tener múltiples documentos asociados (DocPoderes) y puede estar relacionado con varios poderes denegados (PoderesDenegados). Estas relaciones permiten una fácil manipulación y consulta de los datos relacionados.
- **PoderesDenegados.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_poderes_denegados en la base de datos. Este modelo representa la información sobre los poderes que han sido denegados, incluyendo el motivo de la denegación y una referencia al poder específico al que pertenece (pk_poderes).
- **PreguntasVotaciones.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_preguntas_votaciones en la base de datos. Este modelo representa las preguntas de votación y su relación con las opciones de respuesta (OpcionesRespuestas) y las respuestas de votación (RespuestasVotaciones). Además, se especifican los campos que pueden ser asignados masivamente, y las relaciones de uno a muchos con otros modelos.
- **Proposiciones.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_proposiciones en la base de datos. Este modelo representa las proposiciones asociadas a reuniones y usuarios, y establece una relación de pertenencia con el modelo UsuariosReuniones. Además, se especifican los campos que pueden ser asignados masivamente.
- **QR.php:** Este archivo define un modelo Eloquent para interactuar con la tabla adm_qr_usuarios en la base de datos. Este modelo maneja los registros relacionados con los códigos QR de los usuarios, y define los campos que se pueden asignar masivamente, como el ID de usuario, nombre, extensión y la ruta del archivo.
- **Quorum.php:** Este archivo define un modelo Eloquent para interactuar con la tabla adm_quorum en la base de datos. Este modelo maneja los registros relacionados con los quorum de las reuniones, permitiendo la gestión de información como el nombre, la fecha, la hora y el porcentaje de participación. Además, establece las relaciones con otros modelos, como Reuniones (un quorum pertenece a una reunión) y DetalleQuorum (un quorum puede tener varios detalles asociados).

- **RespuestasVotaciones.php:** Este archivo define define un modelo Eloquent para interactuar con la tabla adm_respuestas_votaciones en la base de datos. Este modelo maneja las respuestas a las votaciones, almacenando información como el usuario que respondió, la pregunta de la votación, la opción seleccionada y la hora de la respuesta. Además, establece relaciones con los modelos PreguntasVotaciones (una respuesta pertenece a una pregunta de votación) y UsuariosReuniones (una respuesta pertenece a un usuario de reunión).
- **Reuniones.php:** Este archivo define un modelo Eloquent para interactuar con la tabla adm_reuniones en la base de datos. Este modelo maneja la información relacionada con las reuniones, como la cantidad de copropietarios, los enlaces de acceso, las fechas y horas de inicio y fin, entre otros detalles. Además, establece diversas relaciones con otros modelos, como UsuariosReuniones (una reunión puede tener varios usuarios), Quorum (una reunión puede tener varios quórums), CorreosEnviados (una reunión puede tener varios correos asociados), Temas (una reunión puede tener varios temas) y Votaciones (una reunión puede tener varias votaciones). También tiene una relación con el modelo Unidades, indicando que una reunión pertenece a una unidad específica.
- **Roles.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_roles en la base de datos. Este modelo se utiliza para manejar la información relacionada con los roles, como el nombre y la descripción del rol. La propiedad $fillable permite asignar masivamente los atributos nombre y descripcion para la creación o actualización de registros en la base de datos.
- **Temas.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_temas en la base de datos. Este modelo representa los temas asociados con una reunión y permite gestionar atributos como el tema, estado, y la relación con las proposiciones asociadas a cada tema. La propiedad $fillable permite asignar masivamente los atributos tema, estado, y pk_reuniones para la creación o actualización de registros en la base de datos. Además, se define una relación uno a muchos con el modelo Proposiciones, indicando que un tema puede tener varias proposiciones asociadas.
- **Unidades.php:** Este archivo define un modelo Eloquent que interactúa con la tabla adm_unidades en la base de datos. Este modelo representa las unidades asociadas con una reunión y permite gestionar atributos como el nombre, nit, direccion, pagina, telefono_port, telefono_admin, nom_admin, correo_unidad, y correo_admin. La propiedad $fillable permite asignar masivamente estos atributos para la creación o actualización de registros en la base de datos. Además, el archivo define dos relaciones. La primera es una relación uno a muchos con el modelo UsuariosUnidades, indicando que una unidad puede tener múltiples usuarios asociados. La segunda relación es con el modelo Logos, especificando que una unidad puede tener múltiples logos asociados.
- **User.php:** Este archivo define el modelo User que interactúa con la tabla adm_usuarios en la base de datos. El modelo se utiliza para gestionar la información de los usuarios, como su nombre, correo electrónico, teléfono, contraseñas y estado. Utiliza los traits Notifiable y HasApiTokens para habilitar notificaciones y autenticación API con Laravel Passport.

- **UsuariosReuniones.php:** Este archivo define el modelo UsuariosReuniones, que representa la relación entre los usuarios y las reuniones en la base de datos. Esta relación es de muchos a muchos, y el modelo permite acceder a detalles de la reunión, los roles de los usuarios y otros datos relacionados.
- **UsuariosRoles.php:** Este archivo define el modelo UsuariosRoles, que representa la relación entre los usuarios y los roles que se les asignan en el sistema. Cada usuario puede tener un rol asociado, y este modelo se utiliza para gestionar esa relación.
- **UsuariosUnidades.php:** Este archivo define el modelo UsuariosUnidades, que representa la relación entre los usuarios y las unidades a las que están asignados, así como su rol dentro de esa unidad. Este modelo gestiona la información relacionada con la asignación de usuarios a unidades y sus roles.
- **Validaciones.php:** Este archivo define el modelo Validaciones, que representa las validaciones que pueden estar disponibles en el sistema. El modelo gestiona las validaciones que se pueden aplicar, posiblemente en el contexto de formularios u otros procesos que requieran validación de datos.
- **Votaciones.php:** Este archivo define el modelo Votaciones, que representa las votaciones dentro del sistema. El modelo gestiona las votaciones asociadas a reuniones, permitiendo acceder a sus preguntas, respuestas y el estado de la votación.

1.8 Notifications: En esta carpeta se encuentra el archivo que tiene como propósito la gestión y envío de notificaciones personalizadas a los usuarios. En este caso, está diseñada específicamente para la funcionalidad de restablecimiento de contraseñas. En Laravel, la carpeta Notifications generalmente almacena las clases que definen notificaciones personalizadas, las cuales pueden enviarse a los usuarios a través de diferentes canales, como correo electrónico, base de datos, SMS, entre otros.

- **MailResetPasswordNotifications.php:** Este archivo se usa para enviar una notificación por correo electrónico a un usuario cuando se solicita un restablecimiento de contraseña. El correo electrónico incluirá un enlace para que el usuario pueda restablecer su contraseña.

1.9 Providers: Esta carpeta se utiliza para almacenar **proveedores de servicios (service providers)**. En Laravel, los proveedores de servicios son la forma principal de enlazar clases o funcionalidades dentro del contenedor de servicios (service container)

- **AppServiceProvide.php:** Este archivo es un proveedor de servicios en Laravel. En el método register(), se configura la longitud predeterminada de las columnas de tipo string en las migraciones de la base de datos a 191 caracteres, lo cual es útil para evitar problemas con índices de bases de datos en algunas versiones de MySQL. También está comentada una línea que, si se descomentara, haría que Laravel Passport ignore sus migraciones al ejecutar las migraciones de la base de datos. El método boot() está vacío y se utiliza para inicializar servicios o configurar cosas adicionales al inicio de la aplicación.
- **AuthServiceProvider.php:** Este archivo es responsable de registrar y configurar servicios relacionados con la autenticación y autorización en la aplicación. En este caso, se utiliza **Laravel Passport** para la autenticación API, configurando las rutas necesarias para los tokens de acceso y actualización. Además, se especifica que los tokens de acceso y los tokens de actualización expiren después de 10 horas.
- **BroadcastServiceProvider.php:** Este archivo está encargado de configurar el sistema de transmisión de eventos en tiempo real en la aplicación utilizando **Laravel Broadcasting**. El método boot registra las rutas necesarias para el broadcasting y carga los canales definidos en el archivo routes/channels.php, donde se especifican los canales a través de los cuales se enviarán los eventos en tiempo real.
- **EventServiceProvider.php:** Este archivo define los eventos y sus respectivos listeners en la aplicación. En este caso, se escucha el evento Registered (cuando un usuario se registra en el sistema) y, como respuesta, se ejecuta el listener SendEmailVerificationNotification, que envía un correo de verificación al usuario. El método boot garantiza que cualquier lógica adicional relacionada con los eventos se registre correctamente.
- **RouteServiceProvider.php:** Este archivo define cómo se agrupan y configuran las rutas de la aplicación. Incluye métodos para definir tanto las rutas web como las de la API. En particular, las rutas web se configuran con la protección CSRF y el manejo de sesiones, mientras que las rutas API están pensadas para ser sin estado (stateless), utilizando middleware para manejar la autenticación y el control de acceso. El archivo también define el espacio de nombres para los controladores de las rutas.

2. **Carpeta Boostrap:** Esta carpeta contiene el almacenamiento de la caché y lo más importante, el archivo de arranque del sistema (app.php)

    - **App.php:** El archivo app.php establece y configura la instancia principal de la aplicación Laravel, enlazando los componentes clave y permitiendo que la aplicación maneje tanto solicitudes web como de consola de manera eficiente.

3. **Carpeta config:** En esta carpeta contiene archivos de configuración que controlan varios aspectos del comportamiento y la estructura del sistema. Estos archivos permiten configurar de manera centralizada las diversas opciones del framework, como la base de datos, el sistema de caché, las sesiones, las vistas, y muchos otros componentes.

    - **app.php: E**ste archivo permite configurar y gestionar aspectos fundamentales de la aplicación Laravel, como su entorno, idiomas, servicios y dependencias.
    - **auth.php:** Este archivo es clave para gestionar la autenticación en Laravel, permitiendo configurar cómo los usuarios acceden y manejan su información en la aplicación.
    - **broadcasting.php:** Este archivo es la configuración para la transmisión de eventos en Laravel (config/broadcasting.php). Laravel permite "transmitir" eventos a través de diferentes canales, como websockets o servicios externos como Pusher.
    - **database.php:** Este archivo se encuentra dentro de la carpeta config en un proyecto Laravel y tiene como propósito configurar las conexiones de base de datos que la aplicación utilizará. Se especifican detalles como el tipo de base de datos (MySQL, SQLite, PostgreSQL, SQL Server), las credenciales de conexión y opciones adicionales para cada base de datos. El archivo también configura una base de datos Redis para la aplicación, especificando parámetros como el host, el puerto y la base de datos utilizada para operaciones generales y para el caché. Además, se establece la tabla migrations, que mantiene un registro de las migraciones realizadas en la base de datos, y el valor de la conexión predeterminada, que puede ser ajustado mediante el archivo .env.
    - **cache.php:** Este archivo configura las opciones para que Laravel gestione la caché de manera eficiente y flexible según las necesidades de la aplicación.
    - **excel.php:** Este archivo se configura para optimizar el proceso de importar y exportar datos en diferentes formatos, garantizando flexibilidad y personalización según las necesidades del proyecto.
    - **filesystem.php:** Este archivo configura las opciones de almacenamiento y define cómo manejar los archivos tanto localmente como en la nube.
    - **hashing.php:** Este archivo configura las opciones de seguridad relacionadas con el almacenamiento de contraseñas. Permite elegir entre diferentes algoritmos de hash y controlar sus parámetros para ajustar el nivel de seguridad en función de los recursos disponibles.

- **loggin.php:** Este archivo de configuración permite gestionar y personalizar el comportamiento de los registros de tu aplicación, proporcionando múltiples formas de almacenamiento y manejo, desde archivos locales hasta servicios externos.
- **mail.php:** Este archivo se encarga de configurar todos los parámetros necesarios para que Laravel pueda enviar correos electrónicos a través de un servidor de correo, con opciones para usar diferentes métodos de autenticación, seguridad y personalización de los correos (como el diseño en Markdown). Es un archivo esencial si tu aplicación requiere el envío de correos electrónicos, como para notificaciones, registros de usuario, reset de contraseñas, etc.
- **queue.php:** Este archivo configura cómo y dónde Laravel procesará las tareas en segundo plano. Puedes usar diferentes controladores de cola como base de datos, Redis, Beanstalkd, o el servicio SQS de AWS. También permite definir un comportamiento específico cuando los trabajos de la cola fallan, como registrar los fallos en una base de datos para hacer un seguimiento.
- **services.php:** Este archivo se encarga de almacenar las credenciales y configuraciones para integrar servicios de terceros en la aplicación, como servicios de correo, pagos y otros. Cada sección de este archivo está relacionada con un servicio específico, y su propósito es permitir que la aplicación se conecte y utilice esos servicios con las credenciales proporcionadas en el archivo .env (variables de entorno).
- **session.php:** Este archivo configura el comportamiento de las sesiones en una aplicación Laravel. Aquí se definen varias opciones relacionadas con cómo Laravel maneja las sesiones, qué controlador utilizará para almacenarlas y cómo se gestionarán las cookies de sesión.
- **view.php:** Este archivo controla el almacenamiento y la ubicación de las vistas y sus versiones compiladas en Laravel, ayudando a gestionar y optimizar el rendimiento del sistema de plantillas.

4. **Carpeta database:** Esta carpeta está destinada a almacenar todo lo relacionado con la base de datos de la aplicación. Esta carpeta contiene subcarpetas y archivos que permiten manejar la estructura de la base de datos, las migraciones, las semillas, y las factories, entre otras configuraciones. Vamos a analizar sus principales subcarpetas y archivos para entender mejor su propósito.

   4.1 Carpeta factories: En esta carpeta se crea la tabla adm_roles con columnas para clave primaria (pk_roles), nombre, descripción, y marcas de tiempo. Proporciona un método para eliminar la tabla si es necesario (down). Es parte del proceso de control de versiones de la base de datos en Laravel.

   4.2 Carpeta migrations: Se encuentran los archivos de creación de las tablas en la base de datos.

   4.3 Carpeta seeds: Se encuentran las inserciones a la base de datos.

5. **Carpeta public:** En esta carpeta public se almacenan los archivos estáticos tales como los estilos css, imágenes, pdf de los informes, y scripts de javascript.

    - **Index.php:** Este archivo es el punto de entrada principal para las solicitudes HTTP en una aplicación Laravel. Se encuentra en la carpeta public y actúa como el front controller, manejando todas las peticiones que llegan al servidor.

6. **Carpeta resources:** En esta carpeta encontramos algunos recursos del proyecto tales cómo el cambio de lenguaje a inglés en algunas variables tipo fecha que se le muestran al usuario, también la creación o generación de pdf en la aplicación, y el envío de correos electrónicos.

    6.1 Js: Esta carpeta es responsable de manejar la lógica del frontend, integrando Vue y configurando dependencias como Bootstrap, Axios y CSRF. Sirve como base para crear aplicaciones interactivas y conectarlas con el backend de Laravel.

    - **App.js:** En require('./bootstrap'): Carga el archivo bootstrap.js, que inicializa librerías y configuraciones globales como Axios y jQuery. Vue.component, registra un componente de Vue llamado <example-component> desde el archivo components/ExampleComponent.vue. new Vue, crea una instancia de Vue que controla la parte del DOM asociada al ID #app.
    - **Boostrap.js:** Lodash, proporciona funciones útiles para trabajar con arrays, objetos y otros datos. Bootstrap, incluye funcionalidades como modales, pestañas y otros componentes. Axios, configura solicitudes HTTP al backend, asegurándose de que se envíen los tokens CSRF automáticamente. CSRF Token, garantiza que las solicitudes sean seguras contra ataques de falsificación.

    6.2 Lang: En esta carpeta se encuentran dos carpetas una dónde se almacenan unas variables en español y la otra estas mismas variables, pero con su traducción al inglés.

6.3 Views: Carpeta que contiene los informes y las notificaciones de los correos que se envíen.

- Emails**:** Esta carpeta contiene las plantillas Blade utilizadas para construir correos electrónicos en aplicaciones Laravel.

  - **Notificacion-asamblea.blade.php:** Este archivo genera un correo HTML para enviar notificaciones relacionadas con eventos de asambleas en una unidad residencial.
  - **Notificacion-entrevista.blade.php:** Este archivo genera una plantilla HTML sencilla que se utiliza para notificar el resultado de una entrevista a una persona.
  - **Notificacion-llamada.blade.php:** Este archivo genera una plantilla HTML para notificar sobre la necesidad de realizar una llamada a Selección Global con el propósito de ampliar información sobre una persona específica.
  - **Notificacion-visita.blade.php:** Este código genera una plantilla HTML para enviar una notificación sobre una visita domiciliaria programada.
  - **Poder-generado.blade.php:** Este archivo genera una plantilla HTML genera una notificación de denegación de poder.

6.4 PDF:

- **Informe-poderes.blade.php:** Este archivo HTML genera una página de notificación de poderes para una asamblea virtual. La página muestra los siguientes elementos:

  - Encabezado con el logo de "Giramaster" y el nombre de la empresa.
  - Título que muestra a qué unidad de reunión pertenecen los poderes (extraído dinámicamente de la variable {{ $reunion['unidad']['nombre'] }}).
  - Tabla que lista los detalles de los apoderados, como su estado, nombre, cédula, apartamento y el apartamento del que ceden su poder. Los datos se extraen de la variable dinámica $usuarioreunion utilizando un bucle @foreach.
  - Información de contacto de la empresa, al final de la página, con enlaces a su sitio web y correo.
  - Los estilos están definidos dentro del archivo y aplican formato a la tabla, haciendo uso de Bootstrap para el diseño y presentación de la página.

- **Informe-quoron.blade.php:** Este código genera un informe detallado sobre la asistencia y coeficiente de los copropietarios en una reunión virtual. A través de variables PHP y una estructura de bucles foreach, se calcula el coeficiente de asistencia y se muestra en tablas junto con información de conexión, hora de entrada y salida, nombres de los usuarios, y su rol.
- **Informe-tema.blade.php:** Este código es parte de una vista en Laravel, que renderiza un informe de una reunión con propuestas, y permite visualizar su estado (revisada o no). La página también está estilizada de manera que se vea bien en diferentes dispositivos gracias al uso de Bootstrap.
- **Informe-votacion.blade.php:** Este código representa un informe de votación generado dinámicamente, diseñado para mostrar los resultados de una votación en un formato tabular, con detalles de las opciones, coeficientes, porcentajes, y la información de los votantes.
- **Qr_usuarios.blade.php:** El archivo genera dinámicamente una página de códigos QR para cada copropietario, mostrando sus datos personales junto con la imagen del código QR generado. El uso de {{ $reunion['unidad']['nombre'] }} y {{ $usuario['apartamento'] }} indica que este archivo es parte de una aplicación de Laravel, lo que significa que los datos se están pasando desde el backend (PHP) al frontend (HTML).
- **Welcomeblade.php:** Este código constituye la página de inicio por defecto que se genera al crear una nueva aplicación Laravel, ofreciendo enlaces a recursos de aprendizaje y otros servicios útiles del ecosistema Laravel.

7. **Carpeta routes:** Esta carpeta contiene los archivos de definición de rutas para la aplicación. Laravel tiene varias rutas definidas para manejar solicitudes HTTP de diferentes tipos (GET, POST, PUT, DELETE, etc.), y estas rutas se gestionan en archivos dentro de esta carpeta.

    - **Api.php:** Este archivo es parte del sistema de rutas de la aplicación, en el cual se definen las rutas API que gestionan diversas funcionalidades relacionadas con la autenticación, usuarios, reuniones, poderes, votaciones y reportes. Estas rutas están organizadas en grupos y usan middleware para garantizar la seguridad y el control de acceso, permitiendo que solo los usuarios autenticados puedan acceder a ciertas funciones.
    - **Channels.php:** El código define un canal de difusión para usuarios, donde el canal se denomina App.User.{id}. Esto indica que cada usuario tendrá su propio canal, basado en su ID. La función de autorización ($user, $id) verifica si el usuario autenticado (representado por $user) puede escuchar el canal. La autorización es exitosa si el ID del usuario autenticado coincide con el ID proporcionado en el canal ({id}).

- **Console.php:**
  - Artisan::command('inspire', function () {...}): Registra un comando de consola personalizado llamado inspire. Este comando ejecutará la función definida dentro del closure cuando se invoque en la terminal.

  - Inspiring::quote(): Llama al método quote de la clase Inspiring, que devuelve una cita inspiradora aleatoria.

  - $this->comment(): Utiliza el método comment para mostrar un mensaje en color gris en la consola, que en este caso es la cita inspiradora.

  - ->describe('Display an inspiring quote'): Define una descripción que se muestra cuando se consulta la lista de comandos disponibles, proporcionando contexto sobre lo que hace el comando.

  - **web.php:** Route::get('/', function () {...}): Define una ruta HTTP de tipo GET para la URL raíz de la aplicación (/). Esta ruta se activa cuando un usuario accede a la página principal del sitio web.

  - return view('welcome'): Cuando esta ruta es visitada, se devuelve la vista llamada welcome. Esto normalmente se refiere a una vista en el directorio resources/views/welcome.blade.php.