Backend:
Como ya se menciono, el backend sera programado con el framework de Laravel. Se hara uso de migraciones (en lo mas posible). Para facilitar la migracion de desarrollo a produccion (Ver documentacion de Laravel).

Laravel solo sera usado como API REST para las peticiones ya sea del Frontend del sitio o desde cualquier otro acceso remoto.

Autentificacion:
La autentificacion para el uso API sera mediante tokens creados con JWT en Laravel. Con estos tokens el usuario se identificara desde el front. Enviado el token en la cabecera de la peticion identificado como "*Authorization*".

Informacion sobre JWT:
https://github.com/tymondesigns/jwt-auth

Para la autenticacion existe ya un metodo de login, logout y actualizacion de credenciales en:
app\Http\Controllers\AuthenticateController.php

Middlewares de JWT

Informacion sobre middlewares de Laravel:
https://laravel.com/docs/5.5/middleware

JWT tiene sus propios middleware para verificar que exista un token en cada una de las peticiones si es que se requiere.

Esta ultima mencionada tiene un alias para ser llamado "*jwt.auth*".

Roles y permisos
La implementacion de roles y permisos para que los usuarios puedan solo acceder a ciertas funciones del API son manejados en este proyecto por la librería Laravel-permission.

Informacion sobre Laravel-permission:
https://github.com/spatie/laravel-permission

Middlewares de Laravel-permission

Informacion sobre middlewares de Laravel:
https://laravel.com/docs/5.5/middleware

Se crearon para este proyecto dos middlewares para verificar si el usuario identificado tiene permisos para acceder a la accion API que esta solicitando. Sus alias son:

Para verificar si el usuario loggeado pertenece al rol
auth.role:ROL_A_BUSCAR

Donde ROL_A_BUSCAR sera el rol que deseas verificar. Ejemplo:
auth.role:administrador

Para verificar si un usuario tiene un permiso en especifico
auth.perm:PERMISO_A_BUSCAR

Donde PERMISO_A_BUSCAR sera el permiso que deseamos verificar. Ejemplo:
auth.perm:view_users

Providers

En este proyecto ya estan preconfigurados 2 providers, que serviran para facilitar el uso de varias funciones. En este caso existe un provider llamado "*Images*" y otro llamado "*Documents*". Estos nos ayudaran a manejar dichos archivos junto con sus URL's.

Provider Images
Con este provider manejaremos cualquier tipo de imagen. Con el se puede tener la gestion de cualquier imagen en la base de datos y su ruta fisica. Para ver su codigo fuente puedes encontrarlo en:
app\Providers\ImageServiceProvider.php

Provider Documents
Con este provider manejaremos cualquier tipo de documentos. Con el se puede tener la gestion de cualquier documento en la base de datos y su ruta fisica. Para ver su codigo fuente puedes encontrarlo en:
app\Providers\DocumentServiceProvider.php

Notas:
Es necesario saber que el uso de composer esta para que el peso de los proyectos sean lo mas liviano posible. Por lo que despues de hacer cualquier accion en composer debes de saber que la carpeta "*vendor*" debe de estar en el servidor en el que se esta desarrollando.

Errores comunes:

- Method Tymon\JWTAuth\Commands\JWTGenerateCommand::handle() does not exist
Este error es directamente del desarrollador de la librería, la solucion es bastante sencilla, modificando uno de los archivos a la librería. Solucion:
https://github.com/tymondesigns/jwt-auth/issues/1298