Documentación BackOffice Contract

Conceptos importantes :

- Frontend : Aplicación web desarrollada en React JS por Want, que corresponde al front del sistema Contract
- Backoffice : Aplicación web desarrollada en Laravel por Want, que corresponde al intermediario (en algunas ocasiones) entre el cliente administrador Contract y su backend y entre el frontend y el backend
- Backend : sistema provisto por Contract que maneja la lógica de negocio de la aplicación y desde el cual se consume la mayor parte de la data con la que trabaja el sistema

En el presente documento se entregan detalles técnicos sobre la implementación de la aplicación para el backoffice del sistema Contract, tales como variables de configuración de la aplicación, servicios a los cuales se encuentra conectada y las distintas apis que probé el sistema tanto al frontend como al backend.

La finalidad de este documento es facilitar a futuros desarrolladores o administradores de la plataforma su mantenimiento, además de entregar al cliente toda la información relevante sobre la implementación del sistema.

Descripción General de la aplicación

La aplicación se encuentra desarrollada el laravel 7.0 y base de datos MySQL. Fue desarrollada en php 7.2, sin embargo es compatible con versiones superiores. Dentro de las librerías utilizadas para dar soporte a funcionalidades de la aplicación encontramos:

- laravel/passport: utilizada para el funcionamiento de login desde el backoffice y el login provisto mediante api (consumido por el frontend) para usuarios finales. Para el caso del login de usuarios finales se implementó con el uso de tokens JWT.
- laravel/socialite: utilizada para la configuración del login de usuarios finales con redes sociales como Twitter, Facebook y Google

Para un mejor entendimiento de la plataforma es importante comprender que esta se compone de dos grandes conjuntos de funcionalidades. Por un lado tenemos el backoffice, el cual es la parte del sistema que se conecta directamente al backend provisto por contract y se encarga de obtener la información de espacios, zonas, edificios, plantas y demás, y entrega una vista amigable sobre la cual el usuario contract puede visualizar, editar, crear y eliminar lo que necesite.
Por otro lado, nos encontramos con todo el conjunto de apis que la aplicación facilita para el funcionamiento del frontend. Dentro de este conjunto encontramos el soporte de funcionalidades como el registro, login, manejo de sesiones activas, administración de cuenta por parte del usuario, entre otras funcionalidades.

Ambos conjuntos de funcionalidades serán descritos con mayor detalle para un mejor entendimiento.

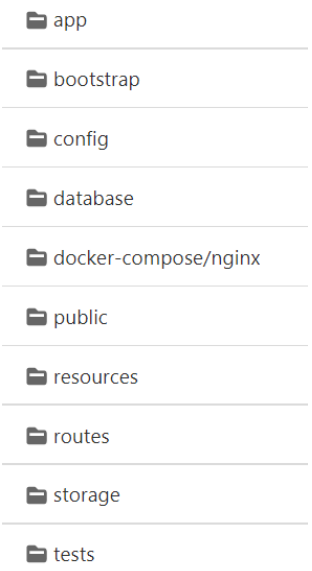
Variables de entorno de la aplicación

Las variables de entorno se encuentran configuradas en el archivo .env, ubicado en la raíz del proyecto. En el es posible configurar :

- APP_NAME : nombre del proyecto
- APP_DEBUG : activar o desactivar el debug de la aplicación (se recomienda siempre mantenerlo deshabilitado en el caso de producción)
- APP_URL : url de referencia de la aplicación
- DB_HOST : host de conexión a la base de datos (podría ser localhost si la base de datos se encuentra dentro de la misma máquina)
- DB_PORT : puerto de conexión a la base de datos
- DB_DATABASE : nombre de la base de datos de la aplicación
- DB_USERNAME : nombre de usuario para la conexión a la base de datos
- DB_PASSWORD : contraseña del usuario que se conectará a la base de datos
- MAIL_MAILER : servicio a utilizar para el envío de correo (por lo general se suele utilizar smtp)
- MAIL_HOST : host del servidor de correo
- MAIL_PORT : puerto del servidor de correo
- MAIL_USERNAME : nombre de usuario del servidor de correo
- MAIL_PASSWORD : contraseña para el acceso al servidor de correo
- MAIL_ENCRYPTION : tipo de encriptación a utilizar (por lo general se utiliza tls)
- MAIL_FROM_ADDRESS : (opcional) permite confirgurar el remitente de todos los correos a enviarse desde el sitio.
- MAIL_FROM_NAME : define el nombre de la aplicación que se incluirá dentro de cada correo
- CONFIRM_URL : url del frontend, de la vista de confirmación de cuenta a ser incluida dentro del correo enviado al usuario
- RESET_URL : url del frontend, de la vista para reestablecer contraseña, que será incluida en el correo al usuario
- CONTRACT_API_URL : url de la api del backend de contract, desde el cual se consume información de edificios, plantas, y demás
- SPACES_TOKEN : token de autenticación para la api del backend provisto por contract
- GOOGLE_CLIENT_ID : client id de app de Google necesaria para configurar el login/registro con esta rrss
- GOOGLE_CLIENT_SECRET : client secret de app de Google necesario para configurar el login/registro con esta rrss
- GOOGLE_REDIRECT_URL : url a la que redireccionará Google tras realizar la autenticación (esta ruta siempre debe ser la ruta base del frontend)
- FACEBOOK_APP_ID : app id de app de facebook necesaria para el funcionamiento del ligin/registro con Facebook

- FACEBOOK_APP_SECRET : app secret de app de Facebook necesaria para el funcionamiento del login/registro con Facebook
- FACEBOOK_REDIRECT : url a la que redireccionará Facebook tras realizar la autenticación (esta ruta siempre debe ser la ruta base del backoffice)
- TOKEN_LIFETIME : tiempo de duración del token del usuario final
- TOKEN_LIFETIME_UNIT : unidad de tiempo en la que es medido el valor ingresado en la variable anterior (day/minute/hour)
- TOKEN_LIFETIME_REFRESH : tiempo de validez que tendrá un token para volver a solicitar uno válido
- TOKEN_LIFETIME_UNIT_REFRESH :unidad de tiempo en la que es medido el valor ingresado en la variable anterior (day/minute/hour)

Estructura interna de archivos



- **app/Console/Commands/..**
  Dentro de es directorio encontrarán los distintos scripts que poblan las tablas con información de: edificios, categorías, espacios, subcategorías y zonas. Cada script está diferenciado y se puede identificar por el nombre de archivo.
- **app/Console/Kernel.php**
  Archivo que contiene la configuración de los scripts mencionados en el punto anterior. En él se define el periodo de ejecución de cada script
- **app/Http/Controllers/api/v1/AuthController.php**
  Controlador que contiene las funciones de las apis relacionadas con el proceso de registro y autenticación de un usuario. Ej: registro normal de usuario, registro con rrss, validación de cuenta, login, entre otras.

- **app/Http/Controllers/api/v1/CountryController.php**
  Controlador que contiene función de api que retorna el listado de países habilitados para el registro
- **app/Http/Controllers/api/v1/FloorController.php**
  Controlador que contiene función que permite la actualización de un plan desde el frontend de la aplicación. También contiene función que retorna el listado de usuarios activos en la aplicación, consumida por el backend.
- **app/Http/Controllers/api/v1/ResetPasswordController.php**
  Controlador que contiene funciones asociadas al proceso de recuperar contraseña. Ej: solicitar correo para recuperación, restablecer contraseña, entre otras.
- **app/Http/Controllers/Auth/…**
  Controladores que contienen la lógica de registro y autenticación de usuarios del backoffice. No se encontrará mayor detalle de la lógica, pues esta se encuentra implementada con una librería.
- **app/Http/Middleware/…**
  Directorio que contiene la lógica de validaciones del login de usuarios. Aquí se define bajo qué reglas un usuario se considera correctamente logueado y qué decisiones se toma tras el login exitoso o fallido.
- **app/Mail/uploadPlan.php**
  Plantilla que contiene estructura del correo que es enviado al usuario administrador por la solicitud de actualización de un plan.
- **app/Models/…**
  Directorio que contiene archivos con la lógica de clases de la bd. En laravel, cada tabla de la bd corresponde a una clase, y la definición de esta en el directorio Models, permite su manipulación en el resto del proyecto.
- **app/Notifications/…**
  Directorio que contiene archivos que establecen la estructura de contenido de los mail enviados al usuario tras los procesos de registro, confirmación de cuenta y recuperación de contraseña.
- **app/Orchid/…**
  Directorio que contiene archivos que definen parte de la estructura de contenido de los módulos del backoffice.
- **app/Untils/CustomValidations.php**
  Archivo que contiene la definición de funciones globales que son utilizadas en la aplicación.
- **app/Codes.php**
  Archivo que contiene la definición de códigos de respuesta manejado por las apis de la aplicación.
- **config/Contract.php**
  Archivo que contiene la llamada a variables de configuración del sistema, como la url del backend de Contract y el token de autenticación de la misma. Dentro del directorio config/… encontrarán también otras configuraciones más genéricas de la aplicación.
- **database/…**
  Directorio que contiene toda la información relacionada al levantamiento inicial de la

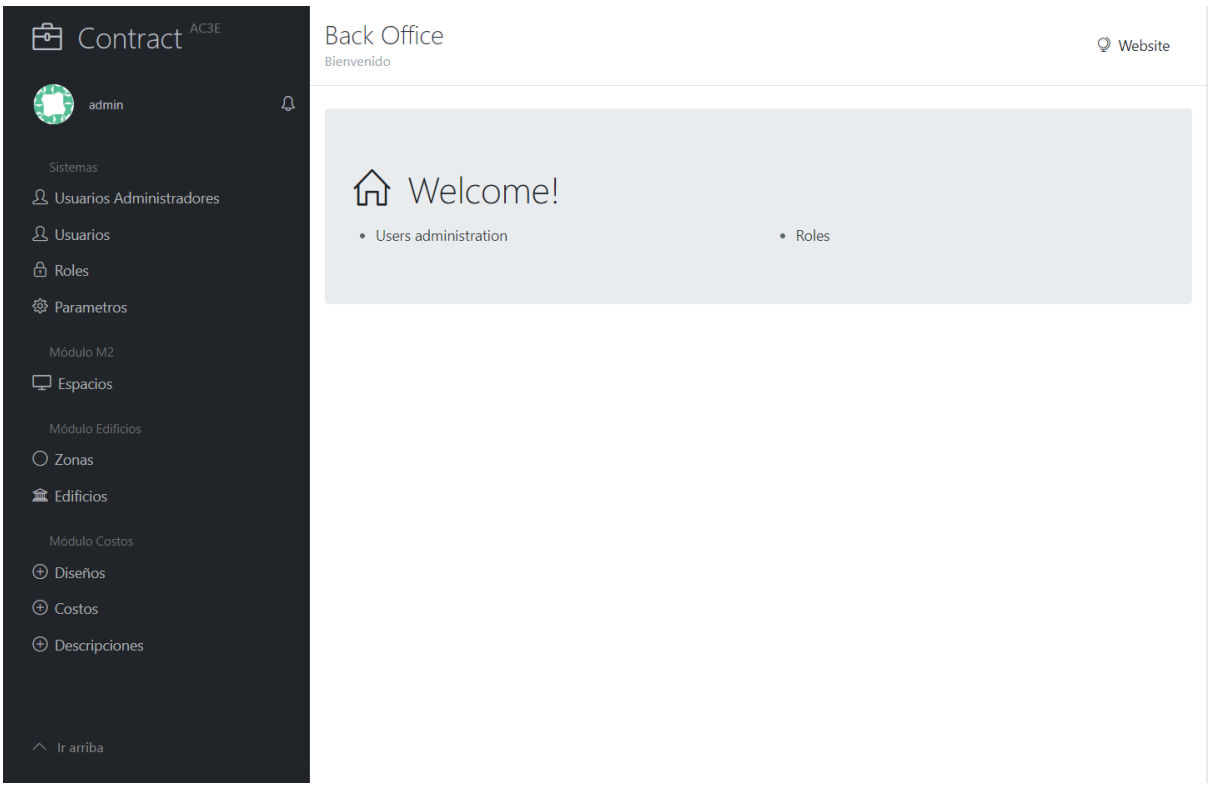
estructura de la bd. Contiene las migraciones definidas para la bd, en el directorio database/migrations/…, y también el listado de información inicial para poblar ciertas tablas, en los directorios database/data/… y database/seeds/…

- **public/…**
  Corresponde al directorio raíz desde el cual corre la aplicación, en el se configura el index.php y htaccess de la aplicación.
- **resources/…**
  Directorio que contiene todo lo relacionado a la parte visual del backoffice. Desde archivos con la estructura de las vistas resources/views/…, hasta archivos de estilos y js que trabajan sobre las mismas.
- **routes/…**
  Directorio que contiene la lógica de ruteo de la aplicación. Dentro de routes/api/v1/api.php encontramos la lista de apis disponibles para ser consumidas por el frontend, junto con su estructura de llamada a funciones. Por otro lado, en el archivo routes/platform.php encontramos la lógica de rutas que conforman el backoffice.
- **storage/…**
  Directorio en el cual se almacenan archivos que son procesados por la aplicación, como por ejemplo archivos de planos a actualizar. También se almacenan archivos de logs de la aplicación. Adicionalmente dentro de este directorio es posible encontrar las llaves públicas y privadas con las cuales se construye la lógica de tokens de autenticación de la aplicación (storage/oauth-private.key y storage/oauth-public.key).
- **.env**
  Archivo que contiene las variables globales posibles de configurar de la aplicaicón.

Backoffice

El sistema backoffice corresponde a la interfaz visual provista para que los usuarios administradores puedan revisar toda la información que se cruza desde el frontend y el backend. La ruta de acceso de este sistema, corresponde a la ruta base en donde se monte el proyecto.

A continuación se describen las distintas secciones del sistema.



- Usuarios administradores : Sección que muestra el listado completo de usuarios administradores del sistema. Es posible listar, editar, eliminar y crear un nuevo usuario administrador, el cual se almacena directamente dentro de la base de datos del backoffice. Estos usuarios son los que tendrán permiso de acceder al backoffice.
- Usuarios : Sección que muestra el listado completo de usuarios finales de la plataforma. Es posible listar, editar, eliminar y crear un nuevo usuario final. A pesar de tener todas estas funcionalidades, se espera que el registro y edición de estos usuarios se realice por lo general desde el front, por medio de una api provista para esa funcionalidad. Todos los usuarios finales son almacenados dentro de la base de datos del backoffice.
- Roles : Sección que lista los roles disponibles dentro del sistema. Esta sección es 100% informativa, pues actualmente existe sólo 1 rol y es el rol por defecto con el que se crean todos los usuarios administradores. El fin de la implementación de este módulo, es permitir la implementación de un sistema de roles a futuro. Todos los roles son almacenados dentro de la base de datos del backoffice.
- Parámetros : Sección que lista los parámetros administrables que posee el sistema. Actualmente sólo se encuentra administrable el correo al cual son enviadas todas las solicitudes de actualización de planos dentro de un edificio.
- Espacios : Sección que lista los espacios disponibles dentro del sistema. Es posible listar, filtrar, editar, eliminar y agregar un nuevo espacio. La información de espacios es obtenida desde el backend mediante la consulta a una api rest, la cual es consultada una vez y almacenada en una tabla independiente dentro del backoffice,

con el fin de evitar consultas al servicio por cada acceso a esta sección de la plataforma. Este espejo de la información es actualizado periódicamente, además de entregar la opción de sincronizar en el momento toda la data.

- Zonas : Sección que lista las zonas disponibles dentro del sistema. Es posible listar, filtrar, editar, eliminar y agregar una nueva zona. La información de cada zona es obtenida desde el backend mediante la consulta a una api rest. la cual es consultada una vez y almacenada en una tabla independiente dentro del backoffice, con el fin de evitar consultas al servicio por cada acceso a esta sección de la plataforma. Este espejo de la información es actualizado periódicamente, además de entregar la opción de sincronizar en el momento toda la data.
- Edificios : Sección que lista los edificios y plantas disponibles dentro del sistema. Es posible listar, filtrar, editar, eliminar y agregar un nuevo edificio. La información de cada edificio es obtenida desde el backend mediante la consulta a una api rest. la cual es consultada una vez y almacenada en una tabla independiente dentro del backoffice, con el fin de evitar consultas al servicio por cada acceso a esta sección de la plataforma. Este espejo de la información es actualizado periódicamente, además de entregar la opción de sincronizar en el momento toda la data.
- Diseños : El módulo de diseños facilita la interfaz para la carga de una planilla (.xlsx) al sistema backend mediante una api rest.
- Costos : El módulo de diseños facilita la interfaz para la carga de una planilla (.xlsx) al sistema backend mediante una api rest.
- Descripción de costos: El módulo de diseños facilita la interfaz para la carga de una planilla (.xlsx) al sistema backend mediante una api rest.

Apis rest Backoffice

A continuación se listan todas apis rest provistas por el backoffice, las cuales son consumidas tanto por el frontend como el backend

GET /api/countries

Descripción : Retorna el listado de países disponibles para el registro de usuarios. Consumida por el frontend en el módulo de registro

Success Response


```
{
   "status_code": 2000,
   "countries": [
      {
         "id": 26,
         "name": "Bolivia",
         "short_code": "bo"
      },
      {
         "id": 29,
```

```
            "name": "Brazil",
            "short_code": "br"
        }...
    ]
}
```

POST /api/user/register

Descripción : Api de registro para usuarios finales consumida por el frontend en el módulo de registro

Request

```
{
    "country_id":{{id del país}},
    "email": {{email usuario}},
    "last_name": {{apellido del usuario}},
    "name": {{nombre del usuario}},
    "password": {{contraseña}},
    "password_confirmation": {{confirmación contraseña}}
}
```

Success Response

```
{
    "status_code": 2001,
    "user": [
            "name":{{nombre}},
            "email": {{email}},
            ...
    ]
}
```

Error Response

```
{
    "status_code": 1001,
    "message": {{descripción del error}}
}
```

POST /api/user/social

Descripción : Api para el login/registro mediante redes sociales. Consumida por el frontend en el módulo de login

Request

```
{
    "access_token":{{token entregado por la red social}},
    "provider": {{red social utilizada (google/facebook/linkedin}}
```

}
Success Response
{
"status_code": 2000,
"access_token": {{token JWT de acceso al sistema}},
"status_code": {{tipo de token}},
"user": [
"name":{{nombre}},
"email": {{email}},
...
]
}

Error Response 1
{
"status_code": 1005,
"message": "Invalid credentials"
}

Error Response 2
{
"status_code": 1007,
"message": "User is not active."
}

POST /api/user/login

Descripción : Api para login de usuarios finales, consumida por el frontend en la vista de login

Request
{
"email":{{email usuario}},
"password": {{contraseña usuario}}
}

Success Response
{
"status_code": 2000,
"access_token": {{token JWT de acceso al sistema}},
"status_code": {{tipo de token}},
"user": [
"name":{{nombre}},
"email": {{email}},
...
]

}
Error Response 1
{
"status_code": 1001,
"message": {{descripción del error}}
}

Error Response 2
{
"status_code": 1002,
"message": "Unauthorized"
}

Error Response 3
{
"status_code": 1007,
"message": "User is not active."
}

Error Response 4
{
"status_code": 1008,
"message": "User is not confirmed"
}

POST /api/user/confirm

Descripción : Api para la confirmación de cuenta de un usuario final. Consumida por el fronted en vista de confirmación de cuenta

Request
{
"token":{{token para validación}},
}

Success Response
{
"status_code": 2000,
"user": [
"name":{{nombre}},
"email": {{email}},
...
]
}

Error Response 1

```
{
   "status_code": 1001,
   "message": {{descripción del error}}
}
```

Error Response 2

```
{
   "status_code": 1005,
   "message": "This token is invalid."
}
```

Error Response 3

```
{
   "status_code": 1004,
   "message": "No user with that e-mail address."
}
```

POST /api/refresh

Descripción : Api para la renovación del token de sesión. Dentro del encabezado (authorization) de la consulta, debe enviar un token válido (Bearer)

Success Response

```
{
   "status_code": 2000,
   "user": [
         "name":{{nombre}},
         "email": {{email}},
         ...
   ],
  "access_token":{{token de acceso}},
  "type_token": "Bearer",
  "expires:at": {{fecha de expiración del token}},
}
```

POST /api/upload-plan

Descripción : Api para la actualización del plano de un edificio, es consumida por el front en la sección de actualización de plano. Dentro del encabezado (authorization) de la consulta, debe enviar el último token válido del usuario (Bearer). La solicitud es enviada por correo al mail configurado en los parámetros de sistema del backoffice.

Request

```
{
   "building_name":{{nombre del edificio}},
   "address": {{dirección del edificio}},
```

```
    "country":{{país}},
    "city": {{ciudad}},
    "link:at": {{base64 del archivo de la planta}},
}
```

Success Response

```
{
    "message": "Solicitud exitosa"
}
```

GET /api/active-users

Descripción : Api que retorna el listado completo de usuarios finales activos en el sistema. Esta api es consumida por el backend de Contract y para consultarla es necesario incluir dentro del encabezado (authorization) un token válido del usuario (Bearer)

Success Response

```
{
    "status_code": 2000,
    "activeUsers": {{array usuarios finales activos en la plataforma}}
}
```

POST /api/user/password/email

Descripción : Api creada para enviar un email con link para la recuperación de contraseña de la cuenta de un usuario. Consumida por el frontend en la vista de recuperación de contraseña

Request

```
{
    "email":{{email del usuario que desea recuperar contraseña}}
}
```

POST /api/user/password/reset

Descripción : Api provista para la actualización de contraseña de un usuario. Consumida por el frontend en la vista de recuperación de contraseña

Request

```
{
    "token":{{token de recuperación}},
    "password":{{nueva contraseña}},
}
```