Integración de Acceso Tamaulipas (FusionAuth) en Aplicaciones Laravel

□□ Introducción

El presente manual tiene como objetivo establecer una guía práctica para la implementación de **Acceso Tamaulipas**, plataforma institucional basada en **FusionAuth**, como proveedor de autenticación en proyectos Laravel, utilizando el paquete **Socialite Providers – FusionAuth** como mecanismo de integración con Laravel Socialite.

Acceso Tamaulipas permite centralizar la autenticación de usuarios mediante flujos basados en **OAuth2/OpenID Connect**, facilitando que las aplicaciones Laravel deleguen el inicio de sesión, validación de identidad y administración de sesiones a una plataforma externa especializada. Esto resulta especialmente útil en ecosistemas donde existen múltiples sistemas administrativos o aplicaciones internas que requieren compartir un mismo esquema de acceso, manteniendo una administración centralizada de usuarios, aplicaciones, roles y permisos.

A través de esta integración, Laravel no se encarga directamente de validar credenciales como usuario y contraseña, sino que redirige al usuario hacia **Acceso Tamaulipas** para realizar el proceso de autenticación. Una vez concluido el inicio de sesión, la plataforma devuelve a la aplicación la información necesaria para identificar al usuario y permitir su acceso conforme a las reglas internas del sistema.

El uso de **Socialite Providers – FusionAuth** simplifica este proceso, ya que proporciona un proveedor compatible con Laravel Socialite para conectarse con FusionAuth de forma estructurada, reutilizable y alineada con las prácticas comunes del framework. Esto permite mantener una implementación limpia, desacoplada y fácil de replicar en distintos proyectos Laravel.

Este manual describe la configuración necesaria para integrar **Acceso Tamaulipas** en una aplicación Laravel, incluyendo la instalación de dependencias, configuración de variables de entorno, registro del proveedor, definición de rutas, implementación del controlador de autenticación y consideraciones generales para el manejo del usuario autenticado dentro del sistema.

La finalidad de esta guía es proporcionar una base estándar para que los proyectos Laravel puedan integrarse con **Acceso Tamaulipas** de manera consistente, segura y mantenible, reduciendo la duplicidad de lógica de autenticación y favoreciendo una arquitectura institucional más ordenada, escalable y preparada para futuros módulos o aplicaciones.

⚙ Configuración Técnica de Integración

Antes de instalar el módulo de autenticación, es importante considerar que la **Agencia de Innovación e Inteligencia Digital** cuenta con un repositorio privado para la distribución de librerías desarrolladas en **PHP**, orientadas principalmente a proyectos construidos con **Laravel**.

Este repositorio permite centralizar paquetes institucionales reutilizables, como módulos, componentes y librerías internas, evitando duplicar lógica entre proyectos y facilitando la estandarización del desarrollo.

Para poder instalar dependencias privadas de la AIIDT mediante Composer, es necesario configurar previamente las credenciales de acceso al repositorio privado a nivel global o a nivel de proyecto, según corresponda.

Consideración de red

Para instalar las librerías privadas de la AIIDT, es necesario estar conectado a la **red de Gobierno** o contar con acceso mediante **VPN a la red interna**, ya que el repositorio privado no se encuentra disponible públicamente en internet.

Una vez instalada la librería y descargadas las dependencias necesarias, se podrá trabajar normalmente en el proyecto Laravel.

Configuración de autenticación para Composer

Antes de ejecutar la instalación de las librerías, se debe registrar el token de acceso al repositorio privado mediante el siguiente comando:

```
composer config --global bearer.172.21.88.204:8002 "pkdt-srJNRh9XpMQreeh8uybq1euUOYhSTZkAqlhKXy6xcc1e389f"
```

Configuración del repositorio en `composer.json`

Además de configurar el token de acceso, se debe validar que el archivo `composer.json` del proyecto cuente con la configuración del repositorio privado.

En caso de no existir, agregar la siguiente configuración al final del archivo, antes del cierre de la última llave `}`:

```
"repositories": {
    "packistry": {
        "type": "composer",
        "url": "http://172.21.88.204:8002/r/aiidt-librerias"
    }
}
```

Ejemplo de referencia dentro del archivo `composer.json`:

```
{
    "name": "laravel/laravel",
    "type": "project",
    "require": {
        "php": "^8.2"
    },
    "repositories": {
        "packistry": {
            "type": "composer",
            "url": "http://172.21.88.204:8002/r/aiidt-librerias"
        }
    }
}
```

Configuración de `secure-http`

Debido a que actualmente el repositorio privado se encuentra publicado mediante `http`, es necesario agregar la siguiente configuración dentro de la llave `config` del archivo `composer.json`:

```
"secure-http": false
```

Ejemplo de referencia:

```
{
    "config": {
        "optimize-autoloader": true,
        "preferred-install": "dist",
        "sort-packages": true,
        "secure-http": false
    }
}
```

> **Nota:** Esta configuración permite que Composer pueda descargar paquetes desde repositorios que no utilizan HTTPS. De momento es necesaria para consumir el repositorio privado interno de la AIIDT.

Instalación de dependencias

Para habilitar la autenticación mediante **FusionAuth** en un proyecto Laravel, es necesario instalar el **módulo de autenticación desarrollado por la AIIDT**, el cual contiene la estructura base requerida para gestionar el flujo de inicio de sesión, retorno de autenticación, cierre de sesión, controlador principal y rutas necesarias para la integración con FusionAuth.

Este módulo permite estandarizar la implementación de autenticación en los proyectos Laravel, evitando duplicar lógica entre aplicaciones y facilitando el mantenimiento centralizado del proceso de autenticación institucional.

Para su instalación, el proyecto debe contar previamente con la librería de módulos de Laravel:

```
composer require nwidart/laravel-modules:^12.0
```

Posteriormente, se deben instalar las siguientes dependencias mediante **Composer**:

```
composer require joshbrw/laravel-module-installer:^2.0
```

```
composer require aiidt/autenticacion-module:^1.1
```

La primera dependencia, `joshbrw/laravel-module-installer`, permite la instalación automática de módulos dentro de la estructura del proyecto Laravel. Esta librería se encarga de ubicar correctamente el módulo dentro del directorio correspondiente, de acuerdo con la configuración del proyecto.

La segunda dependencia, `aiidt/autenticacion-module`, corresponde al módulo de autenticación como tal. Este módulo contiene la lógica necesaria para integrar Laravel con FusionAuth, incluyendo rutas, controladores y configuraciones base requeridas para completar el flujo de autenticación.

Se recomienda **no modificar directamente el módulo de autenticación**, ya que al ser administrado como una librería instalada mediante Composer, cualquier cambio local puede perderse al reinstalar dependencias, actualizar el paquete o desplegar nuevamente el proyecto. En caso de requerir ajustes específicos, estos deberán realizarse mediante configuración externa, extensión del comportamiento o personalización desde el proyecto consumidor, sin alterar el código fuente del módulo.

Con esta configuración, el proyecto queda preparado para incorporar el flujo de autenticación centralizada con FusionAuth de forma ordenada, reutilizable y alineada con el estándar institucional de integración.

Configuración de servicios

En el archivo `config/services.php`, se debe reemplazar o agregar la configuración de `fusionauth` con la siguiente estructura:

```
'fusionauth' => [
    'client_id' => env('FUSIONAUTH_CLIENT_ID'),
    'client_secret' => env('FUSIONAUTH_CLIENT_SECRET'),
    'redirect' => env('FUSIONAUTH_REDIRECT_URI'),
    'base_url' => env('FUSIONAUTH_BASE_URL'),
    'tenant_id' => env('FUSIONAUTH_TENANT_ID'),
    'redirect_home' => env('FUSIONAUTH_REDIRECT_HOME'),
    'url_logout' => env('FUSIONAUTH_URL_LOGOUT'),
],
```

Esta configuración permite que Laravel obtenga desde el archivo `.env` los datos necesarios para comunicarse con FusionAuth, incluyendo las credenciales de la aplicación, la URL base del servicio, el tenant correspondiente, la ruta de retorno después del inicio de sesión y la URL utilizada para cerrar sesión.

Variables de entorno

En el archivo `.env` del proyecto Laravel, se deben agregar las siguientes variables de entorno:

```
FUSIONAUTH_CLIENT_ID=tu_client_id
FUSIONAUTH_CLIENT_SECRET=tu_client_secret
FUSIONAUTH_REDIRECT_URI=${APP_URL}/autenticacion/callback
FUSIONAUTH_BASE_URL=https://acceso.tamaulipas.gob.mx
FUSIONAUTH_TENANT_ID=d9220956-99f7-431f-8de1-de37e65488a8
FUSIONAUTH_REDIRECT_HOME="/inicio"
FUSIONAUTH_URL_LOGOUT="${FUSIONAUTH_BASE_URL}/oauth2/logout?client_id=${FUSIONAUTH_CLIENT_ID}"
```

Estas variables permiten parametrizar la integración sin modificar directamente el código fuente del proyecto, facilitando su mantenimiento y despliegue entre distintos ambientes.

Consideraciones importantes

Es importante tomar en cuenta las siguientes consideraciones antes de realizar la integración:

No se deben modificar las siguientes variables:

```
FUSIONAUTH_REDIRECT_URI=${APP_URL}/autenticacion/callback
FUSIONAUTH_BASE_URL=https://acceso.tamaulipas.gob.mx
```

```
FUSIONAUTH_TENANT_ID=d9220956-99f7-431f-8de1-de37e65488a8
```

La variable `FUSIONAUTH_REDIRECT_URI` define la ruta de callback a la que FusionAuth redirigirá al usuario después de completar el proceso de autenticación. Esta URI debe coincidir exactamente con la configurada en FusionAuth, ya que cualquier diferencia en dominio, protocolo, ruta o prefijo puede provocar errores durante el inicio de sesión.

Asimismo, `FUSIONAUTH_BASE_URL` y `FUSIONAUTH_TENANT_ID` corresponden a la configuración institucional del servicio de autenticación, por lo que deben conservarse sin cambios para garantizar que la aplicación se conecte al tenant correcto.

Registro del Provider de Socialite

Para que Laravel Socialite pueda reconocer y utilizar el proveedor de **FusionAuth**, es necesario registrar el provider correspondiente dentro del archivo:

```
app/Providers/EventServiceProvider.php
```

En este archivo, se debe agregar la siguiente configuración **solo en caso de que no exista previamente**:

```
protected $listen = [
    \SocialiteProviders\Manager\SocialiteWasCalled::class => [
        \SocialiteProviders\FusionAuth\FusionAuthExtendSocialite::class . '@handle',
    ],
];
```

Configuración del Modelo `User`

Para almacenar correctamente la información recibida desde **FusionAuth**, es necesario actualizar el modelo `User` del proyecto Laravel.

Agregar campos a `$fillable`

En el archivo del modelo `User`, normalmente ubicado en:

```
app/Models/User.php
```

se deben agregar los siguientes campos dentro del arreglo `$fillable`:

```
protected $fillable = [
    'name',
```

```
    'email',
    'profile_photo_url',
    'password',
    'fusionauth_id',
    'fusionauth_access_token',
    'fusionauth_refresh_token',
    'data',
    'registration_data',
];
```

Estos campos permiten guardar la información básica del usuario, así como los identificadores, tokens y datos adicionales provenientes de FusionAuth.

Agregar casting de datos

También se debe agregar o actualizar la propiedad `$casts` dentro del mismo modelo:

```
protected $casts = [
    'data' => 'array',
    'registration_data' => 'array',
];
```

Esto permite que los campos `data` y `registration_data` sean tratados automáticamente como arreglos en Laravel, evitando estar haciendo `json_decode()` y `json_encode()`.

Con esta configuración, el modelo `User` quedará preparado para manejar la información extendida del usuario autenticado mediante FusionAuth de forma limpia y estructurada.

Modificación de la tabla `users`

Para que Laravel pueda almacenar correctamente la información que regresa **FusionAuth** durante el proceso de autenticación, es necesario agregar campos adicionales a la tabla `users`.

Estos campos permiten guardar el identificador único del usuario en FusionAuth, los tokens de autenticación, la fotografía de perfil y la información adicional del usuario y su registro dentro de la aplicación.

Opción A: Mediante migración

Esta opción es la **recomendada**, especialmente cuando el proyecto se encuentra en etapa inicial o cuando aún no existen usuarios registrados en la base de datos.

La migración de la tabla `users` deberá contemplar la siguiente estructura:

```
public function up(): void
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password')->nullable();
        $table->rememberToken();

        $table->string('fusionauth_id', 36)->unique();
        $table->text('fusionauth_access_token');
        $table->text('fusionauth_refresh_token')->nullable();
        $table->string('profile_photo_url', 500)->nullable();
        $table->json('data')->nullable();
        $table->json('registration_data')->nullable();

        $table->timestamps();
        $table->softDeletes();
    });
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::dropIfExists('users');
}
```

Con esta modificación, la tabla `users` queda preparada para recibir y persistir la información necesaria para operar con FusionAuth dentro del proyecto Laravel.

Los campos agregados cumplen la siguiente función:

Campo	Descripción
`fusionauth_id`	Identificador único del usuario dentro de FusionAuth.
`fusionauth_access_token`	Token de acceso generado durante el proceso de autenticación.

Campo	Descripción
`fusionauth_refresh_token`	Token utilizado para renovar la sesión o el acceso, cuando aplique.
`profile_photo_url`	URL de la fotografía de perfil del usuario.
`data`	Información adicional del usuario proveniente de FusionAuth.
`registration_data`	Información relacionada con el registro del usuario dentro de la aplicación en FusionAuth.

Es importante considerar que el campo `password` se define como `nullable`, ya que la autenticación no será administrada directamente por Laravel, sino por FusionAuth. En otras palabras.

Opción B: Mediante SQL

Como alternativa, también es posible agregar los campos requeridos directamente mediante una consulta SQL sobre la tabla `users`.

Sin embargo, esta opción **no es la recomendada**, ya que el cambio no queda registrado dentro del sistema de migraciones de Laravel. Esto implica que, en cada nuevo despliegue, reinstalación del proyecto o preparación de ambiente, será necesario ejecutar manualmente el query para asegurar que la tabla tenga la estructura correcta.

```
ALTER TABLE users
    MODIFY COLUMN password VARCHAR(255) NULL,
    ADD COLUMN fusionauth_id VARCHAR(36) NOT NULL UNIQUE AFTER remember_token,
    ADD COLUMN fusionauth_access_token TEXT NOT NULL AFTER fusionauth_id,
    ADD COLUMN fusionauth_refresh_token TEXT NULL AFTER fusionauth_access_token,
    ADD COLUMN profile_photo_url VARCHAR(500) NULL AFTER fusionauth_refresh_token,
    ADD COLUMN data JSON NULL AFTER profile_photo_url,
    ADD COLUMN registration_data JSON NULL AFTER data,
    ADD COLUMN deleted_at TIMESTAMP NULL AFTER updated_at;
```

Esta consulta realiza las siguientes acciones:

Acción	Descripción
`MODIFY COLUMN password VARCHAR(255) NULL`	Permite que el campo `password` pueda ser nulo, ya que la autenticación será gestionada por FusionAuth.
`fusionauth_id`	Agrega el identificador único del usuario en FusionAuth.
`fusionauth_access_token`	Agrega el token de acceso generado durante el inicio de sesión.
`fusionauth_refresh_token`	Agrega el token de renovación, cuando aplique.

Acción	Descripción
`profile_photo_url`	Agrega el campo para almacenar la URL de la fotografía de perfil del usuario.
`data`	Agrega un campo JSON para guardar información adicional del usuario.
`registration_data`	Agrega un campo JSON para guardar información relacionada con el registro del usuario en la aplicación.
`deleted_at`	Agrega soporte para eliminación lógica mediante `SoftDeletes`.

> **Nota:** Si la tabla `users` ya cuenta con alguno de estos campos, el query puede generar errores por columnas duplicadas. En ese caso, se deberán validar previamente los campos existentes antes de ejecutar la consulta.

Prueba de integración

Después de completar la instalación de dependencias, configuración de servicios, variables de entorno, registro del provider, actualización del modelo `User` y modificación de la tabla `users`, se puede realizar una prueba inicial para validar que la integración con **FusionAuth** esté funcionando correctamente.

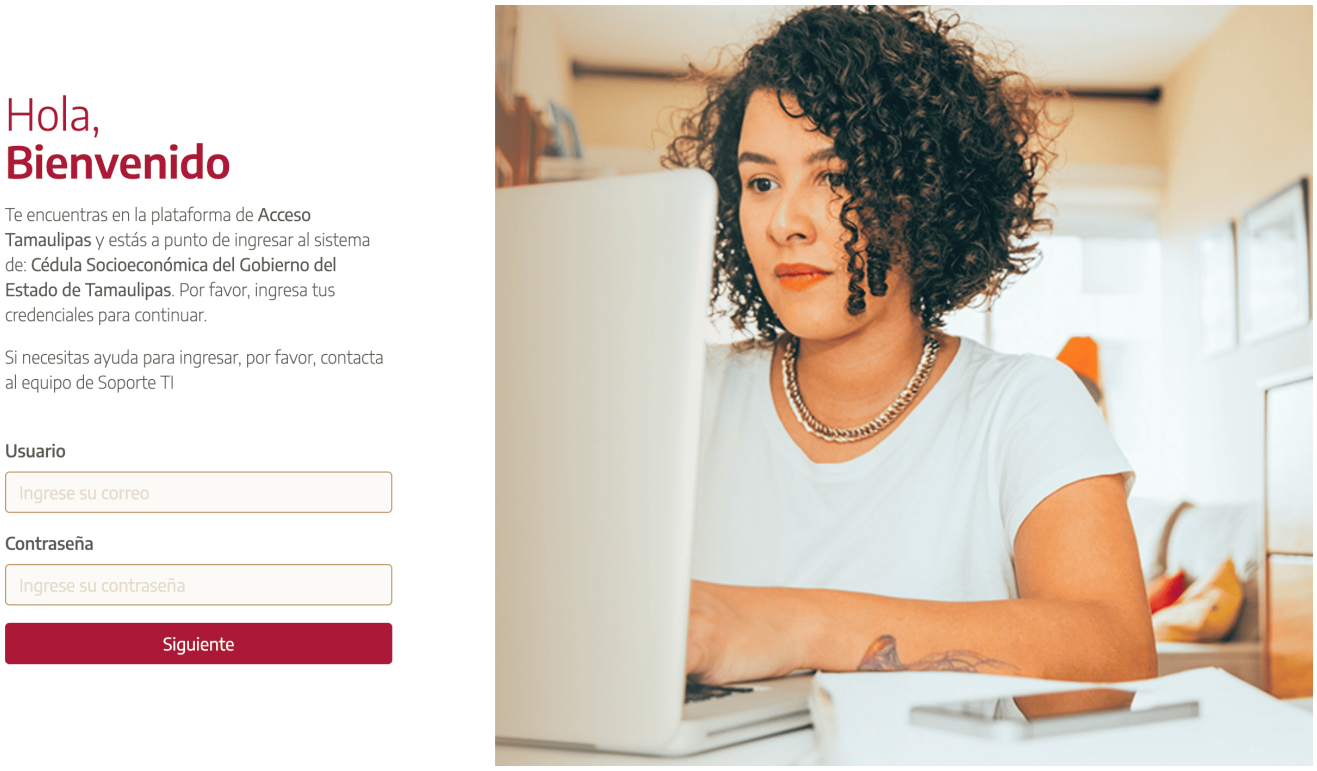
Para ello, se debe ingresar desde el navegador a la siguiente URL:

```
http://${base_url}/autenticacion/login
```

Donde `${base_url}` corresponde al dominio o URL base donde se encuentra publicado el proyecto Laravel.

Al acceder a esta ruta, la aplicación deberá redirigir automáticamente al formulario de inicio de sesión de **FusionAuth**.

La pantalla esperada debe visualizarse de la siguiente manera:



Si se muestra el login de FusionAuth, significa que la aplicación Laravel está redirigiendo correctamente al proveedor de autenticación y que la configuración base del flujo de inicio de sesión se encuentra operativa.

En caso de que no se visualice esta pantalla, o se presente algún error durante la redirección, se recomienda consultar la sección de **Troubleshooting**, donde se describen los errores más comunes y sus posibles soluciones.

□□ Troubleshooting

Esta sección tiene como objetivo ayudar a identificar y resolver los errores más comunes que pueden presentarse durante la integración de **FusionAuth** con proyectos Laravel mediante el módulo de autenticación y Socialite Provider.

Antes de revisar errores específicos, se recomienda validar que se hayan completado correctamente los siguientes puntos:

1. Instalación de dependencias mediante Composer.
2. Configuración de `config/services.php`.
3. Configuración de variables de entorno en `.env`.
4. Registro del provider de Socialite.
5. Configuración del modelo `User`.
6. Modificación de la tabla `users`.
7. Coincidencia exacta de la URI de callback entre Laravel y FusionAuth.

No redirige al login de FusionAuth

Problema

Al ingresar a la ruta: `http://${base_url}/autenticacion/login`

la aplicación no redirige al formulario de inicio de sesión de FusionAuth, muestra una pantalla en blanco, error 404 o permanece en la misma página.

Posibles causas

- El módulo de autenticación no está instalado correctamente.
- Las rutas del módulo no fueron registradas.
- El proyecto no reconoce el módulo de autenticación.
- La URL base del proyecto no corresponde con la URL utilizada en el navegador.
- El caché de rutas o configuración de Laravel está desactualizado.

Solución recomendada

Ejecutar los siguientes comandos dentro del proyecto Laravel:

```
php artisan optimize:clear
php artisan route:clear
php artisan config:clear
php artisan cache:clear
```

```
composer dump-autoload
```

Después, validar que la ruta exista ejecutando:

```
php artisan route:list | grep autenticacion
```

Se debe verificar que existan rutas relacionadas con:

```
/autenticacion/login
/autenticacion/callback
```

Si las rutas no aparecen, revisar que el módulo de autenticación se encuentre instalado correctamente dentro de la estructura de módulos del proyecto.

Error por URI de callback incorrecta

Problema

FusionAuth muestra un error relacionado con la URL de redirección o callback.

Puede presentarse como:

```
Invalid redirect_uri
```

o

```
The redirect_uri is not valid
```

Posibles causas

- La variable FUSIONAUTH_REDIRECT_URI no coincide exactamente con la configurada en FusionAuth.
- Se está usando http en lugar de https, o viceversa.
- La aplicación tiene un prefijo en la URL y no fue considerado.
- La variable APP_URL está mal configurada.
- Hay una diagonal / extra o faltante en la URL.

Solución recomendada

Validar en el archivo .env:

```
APP_URL=https://dominio-del-proyecto.gob.mx
FUSIONAUTH_REDIRECT_URI=${APP_URL}/autenticacion/callback
```

La URI generada debe coincidir exactamente con la configurada en FusionAuth.

Ejemplo:

```
https://dominio-del-proyecto.gob.mx/autenticacion/callback
```

También se recomienda limpiar caché de configuración:

```
php artisan config:clear
php artisan optimize:clear
```

Importante: la URI de callback debe coincidir exactamente. Para FusionAuth, `http://sistema.gob.mx/callback` y `https://sistema.gob.mx/callback` son rutas diferentes.

Error 500 después del callback

Problema

El usuario inicia sesión en FusionAuth, pero al regresar a Laravel se muestra un error 500.

Posibles causas

- Faltan columnas en la tabla users.
- El modelo User no tiene los campos agregados en $fillable.
- Los campos data o registration_data no tienen casting como array.
- La base de datos no acepta valores nulos en campos requeridos.
- El campo password no permite valores nulos.
- Error al guardar los tokens de FusionAuth.
- Solución recomendada

Validar que la tabla users tenga los siguientes campos:

```
fusionauth_id
fusionauth_access_token
fusionauth_refresh_token
profile_photo_url
data
registration_data
```

Validar que el campo password permita valores nulos:

```
DESCRIBE users;
```

En el modelo User, revisar que existan los campos en $fillable:

```
protected $fillable = [
    'name',
    'email',
    'profile_photo_url',
    'password',
    'fusionauth_id',
    'fusionauth_access_token',
    'fusionauth_refresh_token',
    'data',
    'registration_data',
];
```

También revisar que los campos JSON tengan casting:

```
protected $casts = [
    'data' => 'array',
    'registration_data' => 'array',
];
```

Finalmente, revisar el log de Laravel:

```
tail -f storage/logs/laravel.log
```

Error: Class not found

Problema

Laravel muestra un error similar a:

```
Class "SocialiteProviders\FusionAuth\FusionAuthExtendSocialite" not found
```

o

```
Class not found
```

Posibles causas

- No se instaló correctamente el paquete del provider de FusionAuth.
- Composer no actualizó el autoload.
- El provider fue registrado con un namespace incorrecto.
- El archivo EventServiceProvider.php tiene una referencia mal escrita.

Solución recomendada

Validar que las dependencias estén instaladas:

```
composer show | grep fusionauth
composer show | grep socialite
```

Regenerar el autoload:

```
composer dump-autoload
php artisan optimize:clear
```

Revisar que en `app/Providers/EventServiceProvider.php` exista la siguiente configuración:

```
protected $listen = [
    \SocialiteProviders\Manager\SocialiteWasCalled::class => [
        \SocialiteProviders\FusionAuth\FusionAuthExtendSocialite::class . '@handle',
    ],
];
```

Error de configuración de services.php

Problema

La aplicación no puede iniciar el flujo de autenticación o genera errores relacionados con configuraciones vacías.

Posibles causas

- No existe la entrada fusionauth en config/services.php.
- Alguna variable del .env está vacía.
- Se modificó el nombre de una variable.
- El caché de configuración de Laravel sigue usando valores anteriores.

Solución recomendada

Validar que `config/services.php` tenga la siguiente estructura:

```
'fusionauth' => [
    'client_id' => env('FUSIONAUTH_CLIENT_ID'),
    'client_secret' => env('FUSIONAUTH_CLIENT_SECRET'),
    'redirect' => env('FUSIONAUTH_REDIRECT_URI'),
    'base_url' => env('FUSIONAUTH_BASE_URL'),
    'tenant_id' => env('FUSIONAUTH_TENANT_ID'),
    'redirect_home' => env('FUSIONAUTH_REDIRECT_HOME'),
    'url_logout' => env('FUSIONAUTH_URL_LOGOUT'),
],
```

Después, revisar las variables en .env:

```
FUSIONAUTH_CLIENT_ID=tu_client_id
FUSIONAUTH_CLIENT_SECRET=tu_client_secret
FUSIONAUTH_REDIRECT_URI=${APP_URL}/autenticacion/callback
FUSIONAUTH_BASE_URL=https://acceso.tamaulipas.gob.mx
FUSIONAUTH_TENANT_ID=d9220956-99f7-431f-8de1-de37e65488a8
FUSIONAUTH_REDIRECT_HOME="/inicio"
FUSIONAUTH_URL_LOGOUT="${FUSIONAUTH_BASE_URL}/oauth2/logout?client_id=${FUSIONAUTH_CLIENT_ID}"
```

Limpiar configuración:

```
php artisan config:clear
php artisan optimize:clear
```

Error: Client ID o Client Secret incorrecto

Problema

FusionAuth no permite completar el inicio de sesión o marca error relacionado con la aplicación cliente.

Posibles causas

- FUSIONAUTH_CLIENT_ID incorrecto.
- FUSIONAUTH_CLIENT_SECRET incorrecto.
- Las credenciales corresponden a otra aplicación.
- El usuario intenta autenticarse contra un tenant o aplicación distinta.

Solución recomendada

Validar que los valores del .env correspondan exactamente a los configurados en FusionAuth:

```
FUSIONAUTH_CLIENT_ID=tu_client_id
FUSIONAUTH_CLIENT_SECRET=tu_client_secret
```

También confirmar que el usuario tenga acceso a la aplicación configurada en FusionAuth.

Después de modificar estas variables, ejecutar:

```
php artisan config:clear
php artisan optimize:clear
```

Error al guardar el usuario autenticado

Problema

El login se completa en FusionAuth, pero Laravel no puede crear o actualizar el usuario en la base de datos.

Posibles causas

- Falta el campo fusionauth_id.
- fusionauth_id está marcado como único y ya existe un usuario duplicado.
- El correo ya existe con otro fusionauth_id.
- Faltan campos en $fillable.
- Las columnas JSON no existen o no son compatibles con la base de datos.
- Los tokens son demasiado largos para el tipo de columna definido.

Solución recomendada

Validar la estructura de la tabla:

```
DESCRIBE users;
```

Confirmar que los tokens estén definidos como `TEXT`:

```
fusionauth_access_token TEXT
fusionauth_refresh_token TEXT
```

Validar si ya existe el usuario:

```
SELECT id, name, email, fusionauth_id
FROM users
```

```
WHERE email = 'correo@dominio.gob.mx'
   OR fusionauth_id = 'id-de-fusionauth';
```

Si existe un conflicto de datos, revisar si corresponde actualizar el registro existente o depurar el registro duplicado.

Recomendación final

Si después de aplicar las soluciones anteriores el problema continúa, se recomienda recopilar la siguiente información antes de solicitar soporte:

- URL del sistema donde se presenta el error.
- Captura de pantalla del error.
- Fragmento relevante del archivo storage/logs/laravel.log.
- Valor configurado en APP_URL.
- Valor configurado en FUSIONAUTH_REDIRECT_URI.
- Confirmación de que la URI de callback coincide con FusionAuth.
- Resultado de php artisan route:list | grep autenticacion.
- Confirmación de que la tabla users contiene los campos requeridos.

Con esta información será más sencillo identificar el origen del problema y aplicar una solución puntual.

⚠ Consideraciones Finales

La implementación de **Acceso Tamaulipas (Fusionauth)** en proyectos Laravel permite centralizar el proceso de autenticación y mantener un esquema institucional uniforme para el acceso a las aplicaciones. Esto facilita la administración de usuarios, reduce la duplicidad de lógica entre sistemas y mejora el mantenimiento general de los proyectos.

Antes de iniciar la integración, se deberá solicitar formalmente la **creación de la aplicación dentro de Acceso Tamaulipas (Fusionauth)**, con la finalidad de que se generen y proporcionen las llaves de acceso necesarias para la conexión del sistema, tales como el `client_id` y `client_secret`.

Dicha solicitud deberá realizarse mediante correo electrónico dirigido a **[nombre del área o dirección correspondiente]**, al correo **[correo@tamaulipas.gob.mx]**, indicando la información general del sistema que se desea integrar, el ambiente correspondiente y la URL donde será publicado el proyecto.

Una vez creada la aplicación en FusionAuth, se deberán configurar las variables de entorno correspondientes en el archivo `.env` del proyecto Laravel, utilizando las llaves de acceso proporcionadas. Es importante conservar correctamente la configuración definida para FusionAuth, especialmente las variables relacionadas con la URL base, el tenant y la URI de callback, ya que estos valores deben coincidir con la configuración registrada en FusionAuth.

Asimismo, se recomienda utilizar migraciones para cualquier modificación en la estructura de la base de datos, ya que esto permite mantener trazabilidad de los cambios y facilita los despliegues en nuevos ambientes. Aunque es posible realizar ajustes mediante SQL directo, esta opción debe considerarse únicamente como alternativa temporal o de emergencia.

El módulo de autenticación debe mantenerse sin modificaciones directas, ya que al ser administrado como una dependencia mediante Composer, cualquier cambio local puede perderse durante actualizaciones o reinstalaciones. En caso de requerir ajustes específicos, se deberán implementar desde el proyecto consumidor o mediante configuraciones externas.

Finalmente, antes de liberar la integración en un ambiente productivo, se deberá validar que el flujo completo funcione correctamente: redirección al login de FusionAuth, autenticación del usuario, retorno mediante callback, registro o actualización del usuario en Laravel, cierre de sesión y validación de acceso al sistema.

Con estas consideraciones, el proyecto quedará preparado para operar con una autenticación centralizada, mantenible y alineada con el estándar institucional.