Documentacion API

Controlador de acceso:

Para la generación de token y validaciones he utilizado JWT, que es parecida a Passport.

controlador:

```
class AuthController extends Controller

{

    public function __construct()

    {

        $this->middleware('auth:api', ['except' => ['login',
'register']]);

    }



    /**

     * Valida la solicitud y luego intenta autenticar al usuario con
las credenciales proporcionadas.

     * Si la autenticación falla, devuelve un error 401. Si tiene
éxito, devuelve un mensaje de éxito

     * con el token

     *

     * @param Request request El objeto de la solicitud.

     *

     * @return Una ficha

     */
```

```php
public function login(Request $request)
{
    $request->validate([
        'email' => 'required|string|email',
        'password' => 'required|string',
    ]);

    $credentials = $request->only('email', 'password');
    $token = Auth::attempt($credentials, true);

    if (!$token) {
        return response()->json([
            'status' => 'error',
            'message' => 'Unauthorized',
        ], 401);
    }

    return response()->json([
        'status' => 'success',
        'token' => $token,
    ]);
}

/**
 * Crea un nuevo usuario, inicia sesión y devuelve un token
 *
 * @param Request request El objeto de la solicitud.
```

```php
     */
    public function register(Request $request)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6',
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        $token = Auth::login($user);
        return response()->json([
            'status' => 'success',
            // 'user' => $user,
            'token' => $token,
        ]);
    }

    /**
     * Cierra la sesión del usuario y devuelve una respuesta JSON
     *
     * @return Una respuesta JSON con un estado y un mensaje.
```

```
     */
    public function logout()
    {
        Auth::logout();
        return response()->json([
            'status' => 'success',
            'message' => 'Successfully logged out',
        ]);
    }

    /**
     * Toma el token de la solicitud y luego usa el método
`Auth::refresh()` para generar un nuevo
     * token
     *
     * @return El token está siendo devuelto.
     */
    public function refresh()
    {
        $token = Auth::refresh();
        return response()->json([
            'status' => 'success',
            // 'user' => Auth::user(),
            'token' => $token,
        ]);
    }
}
```

Controlador de Peticiones:

El controlador utiliza los modelos Empresas y EmpresasDiario para interactuar con la base de datos y obtener los datos necesarios.

El método index se encarga de obtener los datos de la tabla EmpresasDiario si la fecha proporcionada está dentro de los últimos 7 días, de lo contrario, obtiene los datos de la tabla Empresas.

Los datos se obtienen a través de una solicitud HTTP que incluye los parámetros opcionales empresa, fecha y hora. La fecha se convierte en un objeto de DateTime para poder calcular la diferencia con la fecha actual.

Si la fecha proporcionada está dentro de los últimos 7 días, el método recupera la hora máxima disponible en la tabla EmpresasDiario para cada una de las empresas proporcionadas. Si se proporciona una hora, se utiliza para obtener los registros correspondientes a esa hora. Si no se proporciona una hora, se obtienen todos los registros de la fecha especificada.

Si la fecha proporcionada no está dentro de los últimos 7 días, se obtienen los registros correspondientes de la tabla Empresas utilizando el parámetro opcional fecha.

El método devuelve una respuesta JSON con el estado de la solicitud y los datos obtenidos. En caso de un error en la solicitud, se devuelve una respuesta con un código de estado HTTP 500 y un mensaje de error.

```
class EmpresasController extends Controller
```

```php
{
    public function __construct()
    {
        $this->middleware('auth:api');
    }

    /**
     * Obtiene los datos de la tabla EmpresasDiario si la fecha está dentro de los últimos 7 días, en
     * caso contrario obtiene los datos de la tabla Empresas
     *
     * @param Request request El objeto de la solicitud.
     *
     * @return El método devuelve una respuesta JSON con el estado de la solicitud y los datos.
     */
    public function index(Request $request)
    {
        $empresas = $request->input('empresa');
        $fecha = $request->input('fecha');
        $hora = $request->input('hora');

        try {
            /* Convertir la fecha en una cadena y luego calcular la diferencia entre la fecha actual y
            la fecha pasada en la solicitud. */
            if (!empty($fecha)) {
                $fecha = new DateTime($fecha);
```

```
                $hoy = new DateTime();

                $maxFecha = EmpresasDiario::query()
                    ->whereIn('Empresa', $empresas)
                    ->max('Fecha');
                $maxFecha = new DateTime($maxFecha);

                if ($fecha > $maxFecha) {
                    $dias_diferencia = date_diff($maxFecha, $hoy);
                    $fecha = $maxFecha;
                } else {
                    $dias_diferencia = date_diff($fecha, $hoy);
                }

                $dias_diferencia = $dias_diferencia->format('%a');
            } else {
                $dias_diferencia = null;
            }

            if ($dias_diferencia != null && $dias_diferencia <= 7) {

                /* Obteniendo el máximo de horas de la tabla
EmpresasDiario. */
                if (!empty($hora)) {
                    $hora = EmpresasDiario::query()
```

```
                        ->whereIn('Empresa', $empresas)
                        ->where('Fecha', '>=', $fecha)
                        ->max('Hora');
                }

                $todos = [];
                foreach ($empresas as $empresa) {
                    /* Obteniendo el máximo de horas de la tabla
EmpresasDiario. */
                    if (!empty($hora)) {
                        $hora = EmpresasDiario::query()
                        ->where('Empresa', $empresa)
                        ->when($fecha != null, function ($query) use
($fecha) {
                            return $query->where('Fecha', '>=',
$fecha);
                        })
                        ->max('Hora');

                        $registros = EmpresasDiario::query()
                            ->where('Empresa', $empresa)
                            ->where('Fecha', '>=', $fecha)
                            ->where('Hora', $hora)
                            ->get();

                        $todos = array_merge($todos,
$registros->toArray());
                    }else{
```

```
                        $registros = EmpresasDiario::query()
                            ->where('Empresa', $empresa)
                            ->where('Fecha', '>=', $fecha)
                            ->get();

                        $todos = array_merge($todos,
$registros->toArray());
                    }
                }

                $todos = collect($todos);
            } else {
                $todos = Empresas::query()
                    ->whereIn('Empresa', $empresas)
                    ->when($fecha != null, function ($query) use
($fecha) {
                        return $query->where('Fecha', '>=', $fecha);
                    })
                    ->get();
            }

            return response()->json([
                'status' => 'success',
                'Empresas' => $todos,
            ]);
        } catch (\Exception $e) {
            return response()->json([
                'status' => 'error',
```

```
                'message' => $e->getMessage(),

            ], 500);

        }

    }

}
```

Cómo utilizar la API:

Para poder hacer peticiones primero debemos estar registrados y logueados para poder obtener un token con el que entonces permite realizar la petición (este token caduca a los 20 minutos).

Después de obtener el token para hacer peticiones el formato debe ser así:

localhost/api/empresas?empresas[]=bbva

Siempre debe ser /api/empresas? ; luego añadimos tantas empresas como queramos solicitar ejemplo:

localhost/api/empresas?empresa[]=bbva&empresa[]=inditex&empresa=ferrovial