# Parte 1

Crear un apiRest con las siguientes funcionalidades:
***POST → ruta [login]***:
Se envían el *correo* y la *clave* (parámetro ***obj_json***) a la ruta login por método POST.
Se invocará al método de instancia *VerificarUsuario* de la clase *Verificadora*.
Si el usuario existe en la base de datos, se creará un **JWT** (método de clase *CrearJWT*, de la clase *Autentificadora*) con todos los datos del usuario y 5 minutos de validez.
Se retornará un **JSON**({*jwt*} y status 200).
Si el usuario no existe en la base de datos, retornará el *jwt* nulo y el *status* 403.

```
$app->post('/login[/]', \Verificadora::class . ':VerificarUsuario')
```

# Parte 2

Agregar un middleware (método de instancia *ValidarParametrosUsuario* de la clase *Verificadora*) que permita el paso al próximo calleable si:
Existe el parámetro ***obj_json***, si no informar del error.
Existe el atributo *correo*, si no informar del error.
Existe el atributo *clave*, si no informar del error.
El middleware retornará un **JSON** con *mensaje* (sólo si hubo errores) y *status* 403 o el resultado del calleable y *status* 200.

```
$app->post('/login[/]', \Verificadora::class . ':VerificarUsuario')->add(\Verificadora::class . ':ValidarParametrosUsuario'
```

# Parte 3

***GET → ruta [login/test]***:
Se enviará, en el encabezado *HTTP*, un **JWT** (parámetro ***token***) y se devolverá el *payload* (con la información del usuario) que contiene el token.
Para ello, se invocará al método de instancia *ObtenerDataJWT*, de la clase *Verificadora*, que invocará al método de clase *ObtenerPayLoad($token)*, que retornará un **JSON** ({*éxito:true/false, payload:null/datos, mensaje:””/texto de error*}) a partir del token que recibe como parámetro.

```
$app->get('/login/test', \Verificadora::class . ':ObtenerDataJWT')
```

# Parte 4

Agregar un middleware (método de instancia *ChequearJWT* de la clase *Verificadora*) que permita el paso al próximo calleable si:
Existe el token del encabezado *HTTP* (***token***) y el mismo es válido, si no informar del error.
El middleware retornará un JSON con *éxito:true/false, mensaje:””/texto de error y datos: nulo o la información del próximo calleable*.

```
$app->get('/login/test', \Verificadora::class . ':ObtenerDataJWT')->add(\Verificadora::class . ':ChequearJWT');
```

# Parte 5

Generar un ***CRUD*** (create-read-update-delete) para la tabla *cds*, de la base *cdcols*.
***Grupo → [json_bd]***

***GET → [/].***
Se invocará al método de instancia *TraerTodos* de la clase *CD*, retornará, en formato **JSON**, el contenido completo de la tabla cds. Utilizar PDO.

***GET → [/{id}]***
Se invocará al método de instancia *TraerUno* de la clase *CD*, retornará, en formato **JSON**, el contenido del registro que coincida con el id pasado como parámtetro en la tabla *cds*. Utilizar PDO.

***POST → [/]***
Se invocará al método de instancia *Agregar* de la clase *CD*, retornará, en formato **JSON**, el mensaje de lo acontecido y en el atributo *id_agregado*, el valor del ID del registro agregado o null si no pudo agregarse a la tabla cds. Utilizar PDO.

***PUT → [/]***
Se invocará al método de instancia *Modificar* de la clase *CD*, retornará, en formato **JSON**, el mensaje de lo acontecido. Utilizar PDO.

***DELETE → [/]***
Se invocará al método de instancia *Eliminar* de la clase *CD*, retornará, en formato **JSON**, el mensaje de lo acontecido y en el atributo *cantidad*, la cantidad de registros afectados al ejecutar un delete a la tabla cds. Utilizar PDO.

```php
$app->group('/json_bd', function (RouteCollectorProxy $grupo) {

  $grupo->get('/', \cd::class . ':TraerTodos');

  $grupo->get('/{id}', \cd::class . ':TraerUno');

  $grupo->post('/', \cd::class . ':Agregar');

  $grupo->put('/', \cd::class . ':Modificar');

  $grupo->delete('/', \cd::class . ':Eliminar');

});
```

# Parte 6

Agregar, a nivel de grupo (***json_bd***), el middleware *ChequearJWT*, descrito en la parte 4. Además, agregar los siguientes middlewares a:

***POST → [/]***

Agregar el middleware *ValidarParametrosCDAgregar* (método de instancia de la clase *Verificadora*) que permita el paso al próximo calleable si:
Existe el parámetro ***titulo***, si no informar del error.
Existe el parámetro ***cantante***, si no informar del error.
Existe el parámetro ***anio***, si no informar del error.
El middleware retornará un **JSON** con *mensaje* (sólo si hubo errores) y *status* 403 o el resultado del calleable y *status* 200.

***PUT → [/]***

Agregar el middleware *ValidarParametrosCDModificar* (método de instancia de la clase *Verificadora*) que permita el paso al próximo calleable si:
Existe el parámetro ***obj***, si no informar del error.
Existe el atributo *id*, si no informar del error.
Existe el atributo *titulo*, si no informar del error.
Existe el atributo *cantante*, si no informar del error.
Existe el atributo *anio*, si no informar del error.
El middleware retornará un **JSON** con *mensaje* (sólo si hubo errores) y *status* 500 o el resultado del calleable y *status* 200.

***DELETE → [/]***

Agregar el middleware *ValidarParametrosCDEliminar* (método de instancia de la clase *Verificadora*) que permita el paso al próximo calleable si:
Existe el parámetro ***obj***, si no informar del error.
Existe el atributo *id*, si no informar del error.
El middleware retornará un **JSON** con *mensaje* (sólo si hubo errores) y *status* 500 o el resultado del calleable y *status* 200.

```php
$app->group('/json_bd', function (RouteCollectorProxy $grupo) {

  $grupo->get('/', \cd::class . ':TraerTodos');

  $grupo->get('/{id}', \cd::class . ':TraerUno');

  $grupo->post('/', \cd::class . ':Agregar')->add(\Verificadora::class . ':ValidarParametrosCDAgregar');

  $grupo->put('/', \cd::class . ':Modificar')->add(\Verificadora::class . ':ValidarParametrosCDModificar');

  $grupo->delete('/', \cd::class . ':Eliminar')->add(\Verificadora::class . ':ValidarParametrosCDBorrar');

})->add(\Verificadora::class . ':ChequearJWT');
```