Securing Your Web Service with JSON Web Token (JWT)

NOTE: Prior to following the steps detailed below, you must successfully integrate into your team project implementation the code that was provided in the `AA-files.zip`.

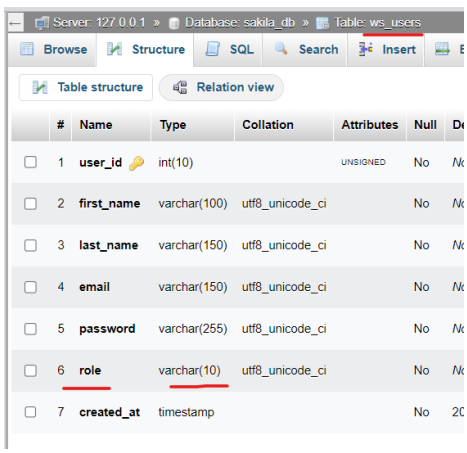
A) Finalizing the integration of the AA layer:

To complete what we have started to integrate a token-based authentication and authorization layer around your Web service, follow the steps below:

1. Open the .htaccess file and uncomment the last line (just remove the # character as shown below).

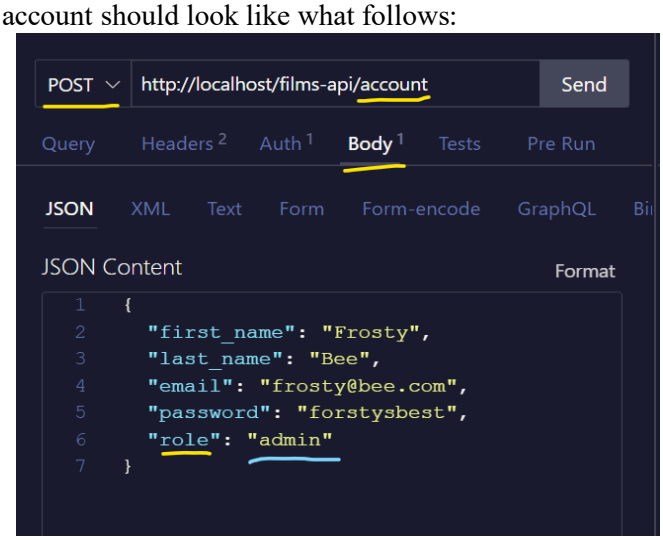
```
# TODO: enable slim-jwt-auth related .
#RewriteRule .* - [env=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
```

2. Add a new column to the `ws_users` table named `role` as shown below. Note that this new field will hold the value that determines what the client application's account can do: **`admin`** (or write: for POST|PUT|DELETE operations), and **`general`** (or read, for GET requests).

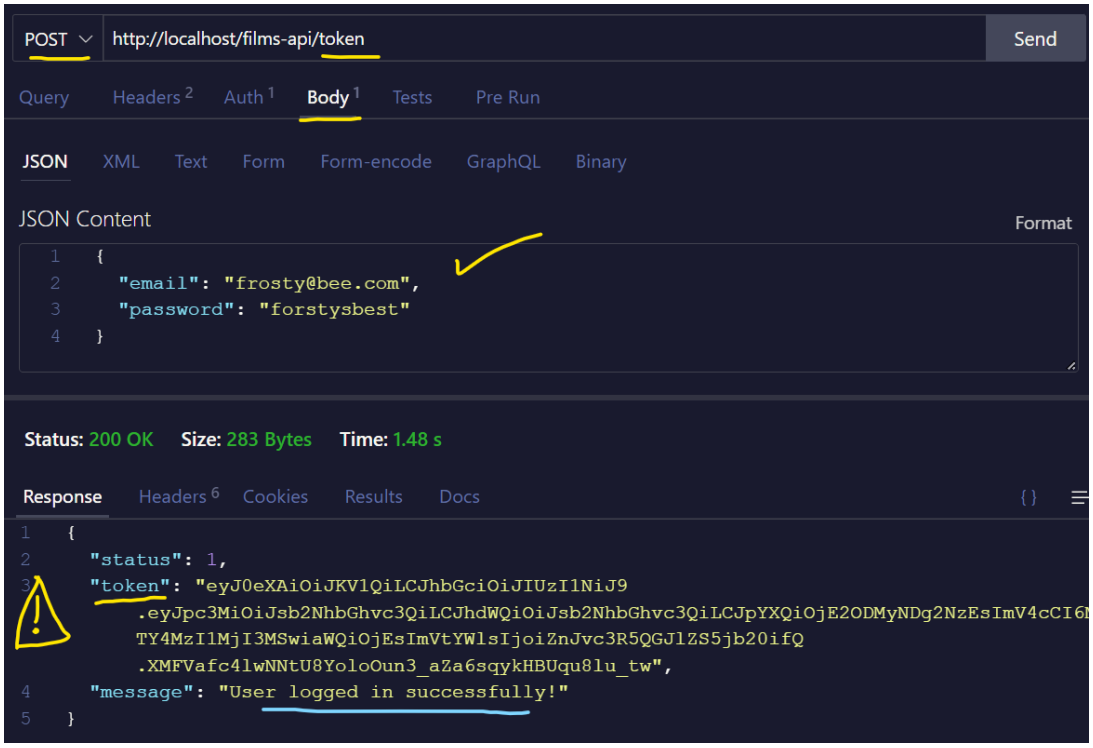


#	Name	Type	Collation	Attributes	Null	De
1	user_id	int(10)		UNSIGNED	No	No
2	first_name	varchar(100)	utf8_unicode_ci		No	No
3	last_name	varchar(150)	utf8_unicode_ci		No	No
4	email	varchar(150)	utf8_unicode_ci		No	No
5	password	varchar(255)	utf8_unicode_ci		No	No
6	role	varchar(10)	utf8_unicode_ci		No	No
7	created_at	timestamp			No	20

3. Modify the code of the `/account`'s callback to add support for the new `role` field (it must be inserted into the `ws_users` table along with the supplied credentials). A request for creating an account should look like what follows:



4. Modify the code of the `/token`'s callback: the value of the `role` field that is associated with the account that has been just authenticated must be included in the payload of the JWT token (along with user id and email)

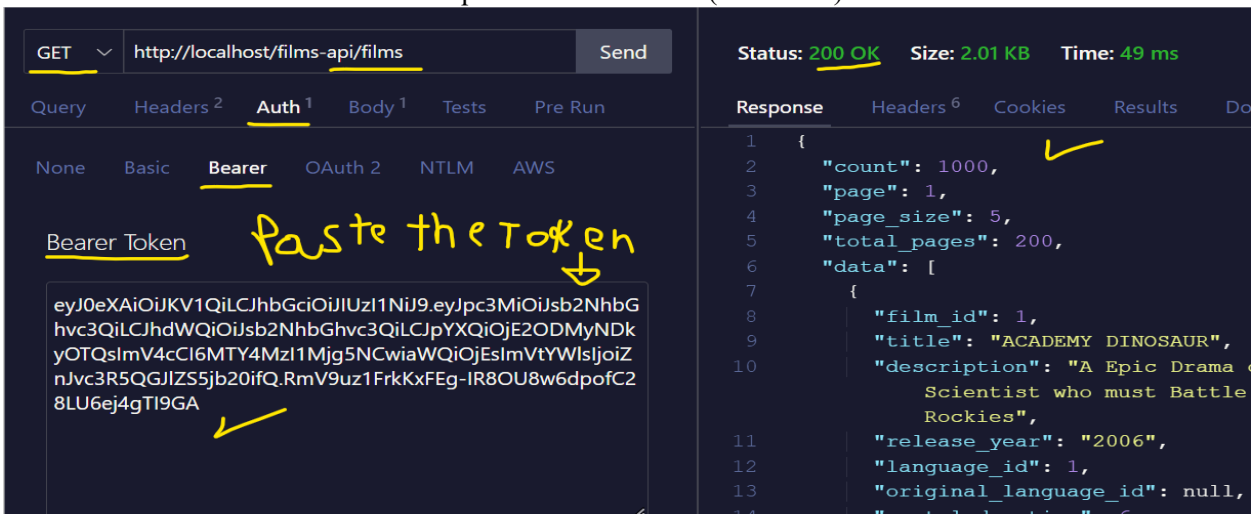


5. Use the latest version of the provided `JWTManager.php` file.
6. Add the provided `JWTAuthMiddleware.php` to your Slim application.
7. Create a new instance of this middleware and add it to your Slim's $app object in `index.php`
8. Add the following line to your `index.php`:
   `define('APP_JWT_TOKEN_KEY', 'APP_JWT_TOKEN');`
9. Add the following to your `index.php`

```
//-- Load from the config.env file the secret used to encode/decode JWTs.
$jwt_secret = JWTManager::getSecretKey();
$app->add(new JWTAuthMiddleware());
```

B) Interacting with your now-secure Web service:

Once you got a token, now you need to include it in every HTTP request's *Authorization* header. The token must be sent embedded in the request as bearer token (see below).



C) Recording interaction/access/etc., information into the database:

The following code shows you: **1)** how to retrieve the decoded JWT's payload that contains the authenticated & authorized account's information from the request object; and **2)** how to log access information into the database using the provided `WSLoggingModel`

```
// Route: /films
1 reference | 0 overrides
public function getAllFilms(Request $request, Response $response)
{
    //-- Logging info into the database.
    // Step 1) Retrieve the parsed JWT from the request object.
    $token_payload = $request->getAttribute(APP_JWT_TOKEN_KEY);
    //var_dump($token_payload);exit;
    // Step 2) Log request info into the ws_log table.
    $logging_model = new WSLoggingModel();
    $request_info = $_SERVER["REMOTE_ADDR"].' '.$request->getUri()->getPath();
    $logging_model->logUserAction($token_payload, $request_info);
    //-----------------------------------
```