Authentication

I’ve chosen Token-based authentication. Due to scalability and independency.

Implementation: “src/Auth.php”

```php
namespace ErfanGooneh\T1;

use ErfanGooneh\T1\Models\User;

class Auth
{
    private const SECRET_KEY = '9afza5vpuemopj83ffu65gt9sh9v8n9s';
    static private function getSign($header, $payload)
    {
        $sign = hash_hmac('sha256', $header . '.' . $payload, self::SECRET_KEY);
        return $sign;
    }
```

Due to JWT standards we have a constant SECRET_KEY (I’ve didn’t include it via environmental vars) to sign the data and then hash it (HMAC).

```php
static public function generateToken($payload)
{
    $payload['issued'] = time();
    $payload = base64_encode(json_encode($payload));
    $header = base64_encode(json_encode([
        'alg' => 'HS256',
        'typ' => 'JWT'
    ]));
    $sign = base64_encode(self::getSign($header, $payload));
    return "$header.$payload.$sign";
}
```

Also we use standard headers(it was not necessary) and adding IssuedTime to payload for hardening it. After that we will encode it to base64

Then we will get the token from HTTP[‘AUTHORIZATION_HEADER’]. Decode it – and retrieve the user.

```php
static public function getUser()
{
    if (!isset($_SERVER['HTTP_AUTHORIZATION']))
        return NULL;
    $token = $_SERVER['HTTP_AUTHORIZATION'];
    if (!self::validateToken($token)) {
        http_response_code(400);
        exit();
    }
    $payload = self::getPayload($token);
    $user = User::get($payload['username']);
    return $user;
}
```