



Temat: Projekt Inżynierski – Dokumentacja kodu projektu 'Anonimowa Ankieta'

Autorzy: Mateusz Nosek, Miłosz Kamiński

1. Opis projektu

Projektem jest system webowy, który umożliwił będzie przeprowadzanie ankiet w sposób gwarantujący anonimowość użytkowników. Aplikacja realizuje następujące funkcje:

- a. Informacje przechowywane są w odpowiedni sposób w bazie danych
- b. Reprezentacja danych nie umożliwia na powiązanie użytkownika z konkretnymi danymi
- c. Istnieje możliwość, aby osoba z wypełnionymi przez siebie ankietami mogły zobaczyć swe odpowiedzi.
- d. Wykładowcy mają możliwość sprawdzenia odpowiedzi anonimowych i niepowiązanych ankiet.

2. Omówienie kodu aplikacji

```
protected function validator(array $data)
{
    return Validator::make($data, [
        'name' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255',
'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
    ]);
}
```

Rys 1. Funkcja tworząca walidację pól.

Fragment kodu przedstawiony na Rys 1 przedstawia walidację wszystkich pól, które mamy w zakładce rejestracji. Wszystkie pola są obowiązkowe, pola 'name' oraz 'email' mają ograniczenie znaków - maksymalnie 255 znaków. Hasło użytkownika musi posiadać minimum 8 znaków. Funkcja jako parametr przyjmuje tablicę z danymi.

```
protected function create(array $data)
{
    return User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => Hash::make($data['password']),
    ]);
}
```

Rys 2. Funkcja tworząca konta użytkownika

Fragment kodu z Rys 2. pokazuje stworzenie użytkownika ze wszystkimi danymi podanymi podczas procesu rejestracji. Warunkiem koniecznym przy tworzeniu konta jest poprawne przejście procesu walidacji.

```
final class Cryption
{
    // Store the cipher method
    // https://www.php.net/manual/en/function.openssl-get-cipher-
    methods.php
    private $ciphering = "AES-128-CTR";

    private $options;
    // Use OpenSSL Encryption method
    private $iv_length ;

    // Non-NULL Initialization Vector for encryption
    private $encryption_iv;

    public function __construct()
    {
        $this->iv_length = openssl_cipher_iv_length($this->ciphering);
        $this->encryption_iv = '1234567891011121';
        $this->options = 0;
    }

    // Use openssl_encrypt() function to encrypt the data
    public function encryption(string $data, string $encryptionKey)
    {
        return openssl_encrypt($data, $this->ciphering, $encryptionKey,
        $this->options, $this->encryption_iv);
    }
}
```

Rys 3. Funkcja szyfrująca odpowiedzi użytkownika

Rys 3. przedstawia funkcję odpowiedzialną za szyfrowanie odpowiedzi użytkownika. Ankieta składa się ze standardowych pytań, na które student udziela odpowiedzi. Pole 'Klucz szyfrujący odpowiedzi' to dodatkowe pole obowiązkowe które ma na celu szyfrować odpowiedzi w zależności od klucza podanego przez użytkownika.

```
$encryption = new Crypton();
$user = Auth::user();

$string = json_encode($answers);
$str = $encryption->encryption($string, $encryptionKey);

DB::table("user_answers")->insert([
    'user_id' => $user->getAuthIdentifier(),
    'questionnaire_id' => $questionnaireId,
    'answers' => $str,
    'created_at' => date('Y-m-d H:i:s')
]);

DB::table("questionnaire_answers")->insert([
    'questionnaire_id' => $questionnaireId,
    'answers' => $string,
    'created_at' => date('Y-m-d H:i:s'),
    'hash' => hash('sha512', $string.$user->email),
    'hash_own' => hash('sha512', $string)
]);

return redirect()->route('questionnaire.answers');
```

Rys 4. Funkcja szyfrująca oraz dodawanie odpowiedzi do bazy danych

Kod przedstawiony na Rys 4. pokazuje jak działa funkcja szyfrująca oraz wykorzystanie metody szyfrującej w momencie dodawania odpowiedzi do bazy danych.

Odpowiedzi znajdują się w tabeli 'user_answers'. Dzięki takiemu rozwiązaniu użytkownik, który wypełnił ankietę ma szansę na sprawdzenie swoich odpowiedzi. Jednocześnie nikt nie jest w stanie zidentyfikować osoby która ankietę wypełniła.

```

public function available()
{
    $user = Auth::user();
    $questionnaire = [];
    $availableQuestionnaire = DB::select(DB::raw(
        "SELECT a.answers, q.id, q.title, q.data FROM questionnaires q
LEFT JOIN user_answers a ON q.id = a.questionnaire_id WHERE a.answers IS
NULL AND q.group_id = ".$user->group_id
    ));

    foreach ($availableQuestionnaire as $item) {
        $questionnaire[] = [
            'id' => $item->id,
            'title' => $item->title,
            'questions' => json_decode($item->data)
        ];
    }

    return view('questionnaire/list', ['questions' => $questionnaire]);
}

```

Rys 5. Funkcja wyświetlająca dostępne ankiety do wypełnienia

Za wyświetlanie dostępnych ankiet do wypełniania odpowiada kod przedstawiony na Rys 5.

```

if (Auth::user()->is_student) {
    $res = DB::select(DB::raw("
        SELECT
            ua.answers as ans,
            q.title
        FROM user_answers ua
        JOIN questionnaires q ON q.id = ua.questionnaire_id
        WHERE ua.user_id = " . $user->getAuthIdentifier() . "
        AND q.id = " . $request->input("question_id")
    ));

    $answersJSON = (new Crypton())->decryption($res[0]-
>ans,$request->input("hash"));
    $answerHASH = hash('sha512', $answersJSON.$user->email);
    $res2 = DB::select(DB::raw(
        "SELECT
            hash, hash_own, answers
        FROM questionnaire_answers
        WHERE hash = '$answerHASH'"
    ));

    $isCorrect = isset($res2[0]->hash) && hash("sha512",$res2[0]-
>answers) == $res2[0] -> hash_own;
    $answersList = json_decode($answersJSON, true);

```

Rys 6. Funkcja odpowiedzialna za odszyfrowanie danych.

Rys 6 przedstawia fragment kodu opowiadający o odszyfrowaniu danych, który sprawdza czy konto jest kontem studenta, jeśli tak, sprawdza poprawność podanego klucza i na jego podstawie przechodzi do odszyfrowania odpowiedzi. Odpowiedzi zapisuje w tabeli z odpowiedziami studenckimi jak i w miejscu odpowiedzialnym za wyświetlanie odpowiedzi osobom uprawnionym do ich odczytywania.

```

$res = DB::select(DB::raw(
    "SELECT
    ua.answers as ans,
    q.title,
    ua.hash_own
    FROM questionnaire_answers ua
    JOIN questionnaires q ON q.id = ua.questionnaire_id
    WHERE ua.questionnaire_id = " . $request->input("question_id")
));
$isCorrect = hash("sha512",$res[0]->ans) == $res[0] -> hash_own;
$answersList = json_decode($res[0]->ans, true);
}

$data['questionnaire'] = [
    'ans' => $res[0]->title,
    'is_correct' => $isCorrect,
    'data' => $answersList
];

```

Rys 7. Funkcja odpowiedzialna za wyświetlanie odpowiedzi.

Wyświetlanie odpowiedzi, które w żaden sposób nie zostały zmanipulowane odpowiada kod przedstawiony na Rys 7. Kod ten sprawdza shas stworzony przy zapisywaniu odpowiedzi studenta jak i sprawdza hash który został wygenerowany po zapisaniu odpowiedzi w tabeli odpowiadającej za wyświetlanie odpowiedzi osobom uprawnionym.

```

@if($quest == false && $questionnaire != false)
    <h3>Ankieta: {{ $questionnaire['ans'] }} -
    <b>Funkcja skrótu @if($questionnaire['is_correct']) poprawna @else
niepoprawna @endif </b>
    </h3>
<ul>
    @if(!$questionnaire['is_correct'])
        <b>Była przeprowadzana ingerencja w Twoje odpowiedzi.</b>
    @endif

    @if($questionnaire['data'] && $questionnaire['is_correct'] )
    @foreach($questionnaire['data'] as $q)
        <li>{{ $q['question'] }}
            <ul>
                <li>{{ $q['answer'] }}</li>
            </ul>
        </li>
    @endforeach
    @endif
</ul>
@endif

```

Rys 8. Fragment kodu przedstawiający widok odpowiedzialny za wyświetlenie odpowiedzi lub powiadomienie o ingerencji osób trzecich..

Za wyświetlanie odpowiedzi odpowiada widok (Rys 8), który po sprawdzeniu odpowiedzi oraz tego, czy użytkownik jest studentem czy też prowadzącym wyświetla odpowiedzi lub komunikat mówiący o manipulowaniu wyników. Komunikat o manipulacji odpowiedziami wyświetla się studentom oraz prowadzącym.