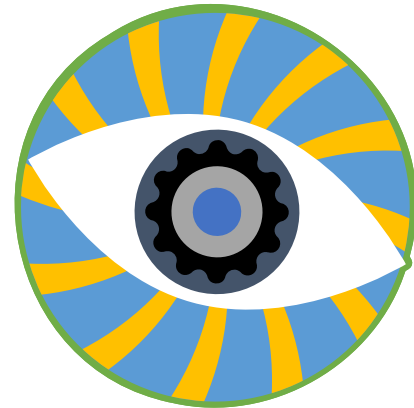
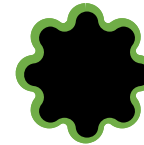


LyriGuess

Druhý ročník
APiN



Ľubo – Back-End

```
/**
 * Logout akcia - odhlásenie používateľa
 *
 * @return \Cake\Http\Response|null
 */
0 references|0 overrides
public function logout(): Response|null
{
    $this->Authentication->logout();

    // KRITICKÁ OPRAVA: úplne zničí a obnoví reláciu, aby zabránila fixácii relácie
    // a zaisťila úplné odhlásenie (opravuje problém, kde používatelia zostávajú prihlásení)
    $this->request->getSession()->destroy();
    $this->request->getSession()->renew();

    $this->Flash->success(message: __(singular: 'Boli ste úspešne odhlásení.'));
    return $this->redirect(url: ['controller' => 'Users', 'action' => 'login']);
}

/**
 * Edit metóda - úprava používateľa
 *
 * @param string|null $id ID používateľa.
 * @return \Cake\Http\Response|null|void Presmeruje pri úspešnej úprave, inak vykresľuje pohľad.
 * @throws \Cake\Datasource\Exception\RecordNotFoundException keď záznam nebol nájdený.
 */
0 references|0 overrides
public function edit($id = null): Response|null
{
    $user = $this->Users->get(primaryKey: $id, contain: []);
    if ($this->request->is(type: ['patch', 'post', 'put'])) {
        $user = $this->Users->patchEntity(entity: $user, data: $this->request->getData());
        if ($this->Users->save(entity: $user)) {
            $this->Flash->success(message: __(singular: 'Používateľ bol úspešne aktualizovaný.'));
            return $this->redirect(url: ['action' => 'index']);
        }
        $this->Flash->error(message: __(singular: 'Používateľa sa nepodarilo aktualizovať. Prosím, skúste znova.'));
    }
    $this->set(name: compact(var_name: 'user'));
}

/**
 * Delete metóda - zmazanie používateľa
 *
 * @param string|null $id ID používateľa.
 * @return \Cake\Http\Response|null|void Presmeruje na index.
 * @throws \Cake\Datasource\Exception\RecordNotFoundException keď záznam nebol nájdený.
 */
0 references|0 overrides
public function delete($id = null): Response|null
{
    $this->request->allowMethod(methods: ['post', 'delete']);
    $user = $this->Users->get(primaryKey: $id);
    // Zabráni zmazaniu admin účtov
    if (empty($user->is_admin)) {
        $this->Flash->error(message: __(singular: 'Admin účty sa nemôžu mazať.'));
        return $this->redirect(url: ['action' => 'index']);
    }

    if ($this->Users->delete(entity: $user)) {
        $this->Flash->success(message: __(singular: 'Používateľ bol úspešne zmazaný.'));
    } else {
        $this->Flash->error(message: __(singular: 'Používateľa sa nepodarilo zmazať. Prosím, skúste znova.'));
    }

    return $this->redirect(url: ['action' => 'index']);
}
```

```
<?php
declare(strict_types=1);

namespace App\Controller;

use Cake\Event\EventInterface;
use Cake\Http\Exception\ForbiddenException;

You, 3 minutes ago | 2 authors (You and others)
/**
 * Kontrolér používateľov
 *
 * @property \App\Model\Table\UsersTable $Users
 */
0 references|0 implementations|You, 3 minutes ago | 2 authors (You and others)
class UsersController extends AppController
{
    /**
     * Index metóda - zoznam používateľov
     *
     * @return \Cake\Http\Response|null|void Vykresľuje pohľad
     */
    0 references|0 overrides
    public function index(): void
    {
        $identity = $this->request->getAttribute(name: 'identity');
        if (!$identity || !$identity->get('is_admin')) {
            throw new ForbiddenException(message: 'You are not authorized to view this page.');
```

```

/**
 * Add metóda - registrácia nového používateľa
 *
 * @return \Cake\Http\Response|null|void Presmeruje pri úspešnom pridaní, inak vykresľuje pohľad.
 */
0 references | 0 overrides
public function add(): Response|null
{
    $user = $this->Users->newEmptyEntity();
    if ($this->request->is('post')) {
        $user = $this->Users->patchEntity(entity: $user, data: $this->request->getData());
        if ($this->Users->save(entity: $user)) {
            $this->Flash->success(message: __('singular: Používateľ bol úspešne vykonaný a prihlásený.'));

            // Ihneď prihlási nového používateľa (ak je podporované) a presmeruje na dennú hru
            if (method_exists(object_or_class: $this->Authentication, method: 'setIdentity')) {
                $this->Authentication->setIdentity($user);
            }

            return $this->redirect(url: ['controller' => 'Play', 'action' => 'daily']);
        }
        $this->Flash->error(message: __('singular: Používateľa sa nepodarilo uložiť. Prosím, skúste znova.'));
    }
    $this->set(compact('user'));
}

```



```

/**
 * beforeFilter
 *
 * Povoľuje neautentifikovaný prístup k akciám login a add.
 *
 * @param \Cake\Event\EventInterface $event Udalosť.
 * @return void
 */
2 references | 0 overrides
public function beforeFilter(EventInterface $event): void You, a month ago
{
    parent::beforeFilter(event: $event);
    $this->Authentication->allowUnauthenticated(['login', 'add']);
}

```





```
/**
 * Login akcia - prihlásenie používateľa
 *
 * @return \Cake\Http\Response|null
 */
0 references [0 overrides]
public function login(): Response|null
{
    $result = $this->Authentication->getResult();

    // Presmeruje na dennú hru ihneď po úspešnom POST prihlásení.
    // Toto umožňuje koreňovej stránke (ktorá smeruje na túto akciu) zobraziť prihlasovací/registračný formulár aj keď používateľ už má platnú reláciu.
    if ($result && $result->isValid() && $this->request->is(type: 'post')) {
        // BEZPEČNOSTNÁ OPRAVA: Obnoví ID relácie po úspešnom prihlásení, aby zabránila fixácii relácie
        // Toto vytvorí nové ID relácie pri zachovaní dát relácie
        $this->request->getSession()->renew();

        // Najprv sa pokúsi získať identitu z Authentication komponentu,
        // v prípade potreby sa vráti k atribútu požiadavky.
        $identity = null;
        if (method_exists(object_or_class: $this->Authentication, method: 'getIdentity')) {
            $identity = $this->Authentication->getIdentity();
        }
        if (!$identity) {
            $identity = $this->request->getAttribute(name: 'identity');
        }

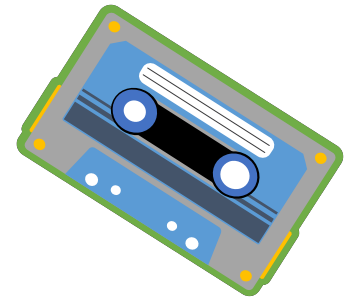
        // Robustná detekcia admina: identita môže byť Entity, ArrayAccess alebo pole.
        $isAdmin = false;
        if ($identity) {
            // Vyskúša bežné vzory prístupu
            if (is_object(value: $identity)) {
                if (method_exists(object_or_class: $identity, method: 'get')) {
                    $val = $identity->get('is_admin');
                } elseif (property_exists(object_or_class: $identity, property: 'is_admin')) {
                    $val = $identity->is_admin;
                } else {
                    $val = $identity->is_admin ?? null;
                }
            } elseif (is_array(value: $identity) || $identity instanceof \ArrayAccess) {
                $val = $identity['is_admin'] ?? $identity['role'] ?? null;
            } else {
                $val = null;
            }

            // normalizácia
            if ($val === true || $val === 1 || $val === '1' || $val === 'admin' || $val === 'yes') {
                $isAdmin = true;
            }
        }

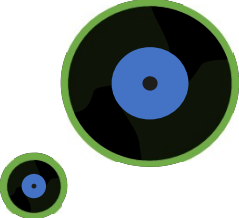
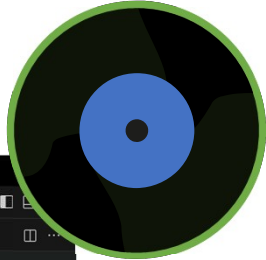
        if ($isAdmin) {
            $target = ['controller' => 'Users', 'action' => 'index'];
        } else {
            $target = $this->Authentication->getLoginRedirect() ?? ['controller' => 'Play', 'action' => 'daily'];
        }

        return $this->redirect(url: $target);
    }

    // Zobrazí chybu pri POST, keď prihlásenie zlyhalo
    if ($this->request->is(type: 'post') && !($result && $result->isValid())) {
        $this->Flash->error(message: __($singular: 'Nesprávne prihlasovacie údaje.'));
    }
}
```



Jakub – Back-end



```
daily.php PlayController.php
LyriGuess > src > Controller > PlayController.php
10 {
115 {
118     $users = $Users->find()
120     //...
124
125     $this->set(compact('users'));
126 }
127
128 /**
129  * Porovná odhad s názvom alebo autorom.
130  * Ignoruje diakritiku, medzery, interpunkciu a veľké/malé písmená.
131  * Povolí aj varianty "autor - názov" a "názov - autor".
132  */
133 private function isCorrectGuess(string $guess, string $title, string $artist): bool
134 {
135     $n = function (string $s): string {
136         // transliterácia diakritiky + zníženie + odfiltrovanie ne-alfa-num
137         $$slug = Text::slug(mb_strtolower($s), ['replacement' => '-']);
138         return preg_replace('/[^a-z0-9]+/', '', (string)$slug) ?? '';
139     };
140
141     $g = $n($guess);
142     $t = $n($title);
143     $a = $n($artist);
144
145     if ($g === $t || $g === $a) {
146         return true;
147     }
148
149     $both1 = $n($artist . ' - ' . $title);
150     $both2 = $n($title . ' - ' . $artist);
151     return $g === $both1 || $g === $both2;
152 }
153 }
154
```



```
div>

php if ($locked && !$solved): ?>
<div style="background:#fff4e6; padding:.75rem; border-radius:.25rem;">
  <strong>Vyčerpal si všetky pokusy (5).</strong>
  <div>Správna odpoveď: <strong><?= h($song->artist) ?>  <?= h($song->song_name) ?></strong></div>
  <div style="margin-top:.5rem; color:#666;">Skús znovu zajtra.</div>
</div>
php else: ?>
<?php if (!$solved): ?>
  <p><em>Nápoveda (krok <?= (int)$hintStep ?> z 5):</em></p>
  <pre style="white-space: pre-wrap; background:#f6f6f6; padding:.75rem;"><?= h($hint) ?></pre>
<?php else: ?>
  <div style="background:#e6ffe6; padding:.5rem; border-radius:.25rem;">
    <input checked="" type="checkbox"/> <strong>Správne!</strong> <?= h($song->artist) ?>  <?= h($song->song_name) ?>
  </div>
<?php endif; ?>

<?= $this->Form->create(null, ['url' => ['action' => 'daily'], 'type' => 'post']) ?>
  <?= $this->Form->control('guess', ['label' => 'Tip (názov alebo autor)', 'required' => true]) ?>
  <?= $this->Form->button('Odoslať') ?>
<?= $this->Form->end() ?>
php endif; ?>
```



Discord

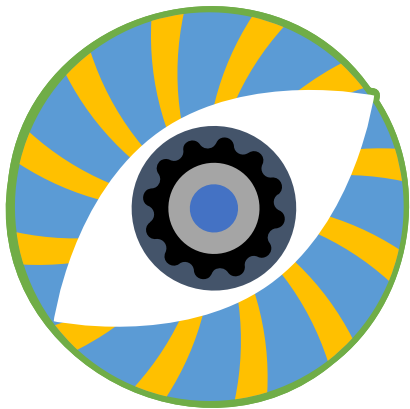
YourWorstNightmare (4 hours ago) Ln 27, Col 4 Spaces: 2 UTF-8 LF {} PHP



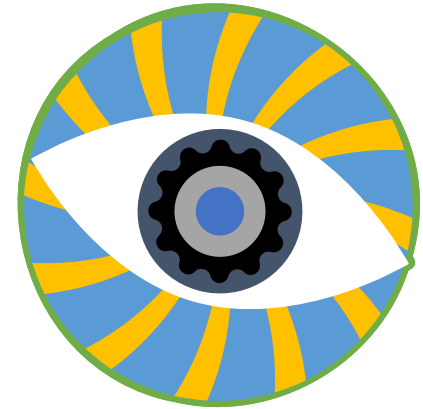
```
LyriGuess > src > Controller > PlayController.php
1 <?php
2 declare(strict_types=1);
3
4 namespace App\Controller;
5
6 use Cake\I18n\FrozenDate;
7 use Cake\Utility\Text;
8
9 class PlayController extends AppController
10 {
11     public function daily()
12     {
13         $Songs = $this->fetchTable('Songs');
14
15         // 1) Ak by DB bola prázdna, zobraz hlášku a skonči
16         $count = (int)$Songs->find()->count();
17         if ($count === 0) {
18             $this->set([
19                 'song' => null,
20                 'attempts' => 0,
21                 'solved' => false,
22                 'hint' => '',
23                 'hintStep' => 1,
24             ]);
25             $this->Flash->warning('V databáze nie sú žiadne piesne. Pridaj aspoň jednu do tabuľky songs.');
```

```
LyriGuess > src > Controller > PlayController.php
10 {
12     {
13         $keyBase = "game.daily.$dateKey.song_{$song->id}";
14
15         // 4) RESET stavu (GET ?reset=1 alebo POST hidden reset=1)
16         if ($this->request->getQuery('reset')) {
17             $session->delete($keyBase);
18             $this->Flash->success('Resetnuté. Môžeš testovať od začiatku.');
```

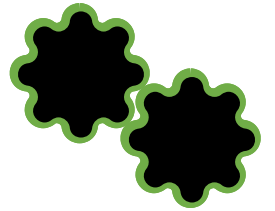




```
LyriGuess > src > Controller > PlayController.php
10 {
59 $locked = (bool)($session->read("keyBase.locked") ?? false);
64 if ($this->request->is('post') && !$solved && $locked && !$this->request->getData('reset')) {
68 } else {
84 } else {
85     $attempts++;
86     $session->write("keyBase.attempts", $attempts);
87     if ($attempts >= 5) {
88         $locked = true;
89         $session->write("keyBase.locked", true);
90     } else {
91         $this->Flash->error('Nesprávne. Skús znova.');
```



Erik – Back-End



```
0 references | 0 implementations
class CreateUsers extends BaseMigration
{
    0 references | 0 overrides
    public function change(): void
    {
        $table = $this->table(tableName: 'users');
        $table->addColumn(columnName: 'username', type: 'string', options: [
            'default' => null,
            'limit' => 255,
            'null' => false,
        ]);
        $table->addColumn(columnName: 'email', type: 'string', options: [
            'default' => null,
            'limit' => 255,
            'null' => false,
        ]);
        $table->addColumn(columnName: 'password', type: 'string', options: [
            'default' => null,
            'limit' => 255,
            'null' => false,
        ]);
        $table->addColumn(columnName: 'created', type: 'datetime', options: [
            'default' => null,
            'null' => false,
        ]);
        $table->addColumn(columnName: 'modified', type: 'datetime', options: [
            'default' => null,
            'null' => false,
        ]);
        $table->create();
    }
}
```

```
class CreateSongs extends BaseMigration
{
    0 references | 0 overrides
    public function change(): void
    {
        $table = $this->table(tableName: 'songs');
        $table->addColumn(columnName: 'artist', type: 'string', options: ['limit' => 255, 'null' => false])
            ->addColumn(columnName: 'song_name', type: 'string', options: ['limit' => 255, 'null' => false])
            ->addColumn(columnName: 'lyrics', type: 'text', options: ['null' => false])
            ->addPrimaryKey(columns: 'id')
            ->create();
    }
}
```

```

<?php if ($users->isEmpty()): ?>
    <p>No users available.</p>
<?php else: ?>

    <table>
        <thead>
            <tr>
                <th></th>
                <th>Meno</th>
                <th>Skóre</th>
            </tr>
        </thead>
        <tbody>
            <?php $i = 1; foreach ($users as $user): ?>
                <tr>
                    <td><?=$i ?></td>
                    <td><?=$this->Html->link(title: h(text: $user->username), url: ['controller' => 'Users', 'action' => 'view', $user->id]) ?></td>
                    <td><?=$user->score ?? 0 ?></td>
                </tr>
            <?php $i++; endforeach; ?>
        </tbody>
    </table>

<?php endif; ?>

```



```

public function leaderboard(): void
{
    $Users = $this->fetchTable(alias: 'Users');

    $users = $Users->find()
        ->select(fields: ['id', 'username', 'score', 'created'])
        ->where(conditions: ['is_admin' => false])
        ->order(fields: ['score' => 'DESC', 'username' => 'ASC'])
        ->limit(limit: 50)
        ->all();

    $this->set(name: compact(var_name: 'users'));
}

```

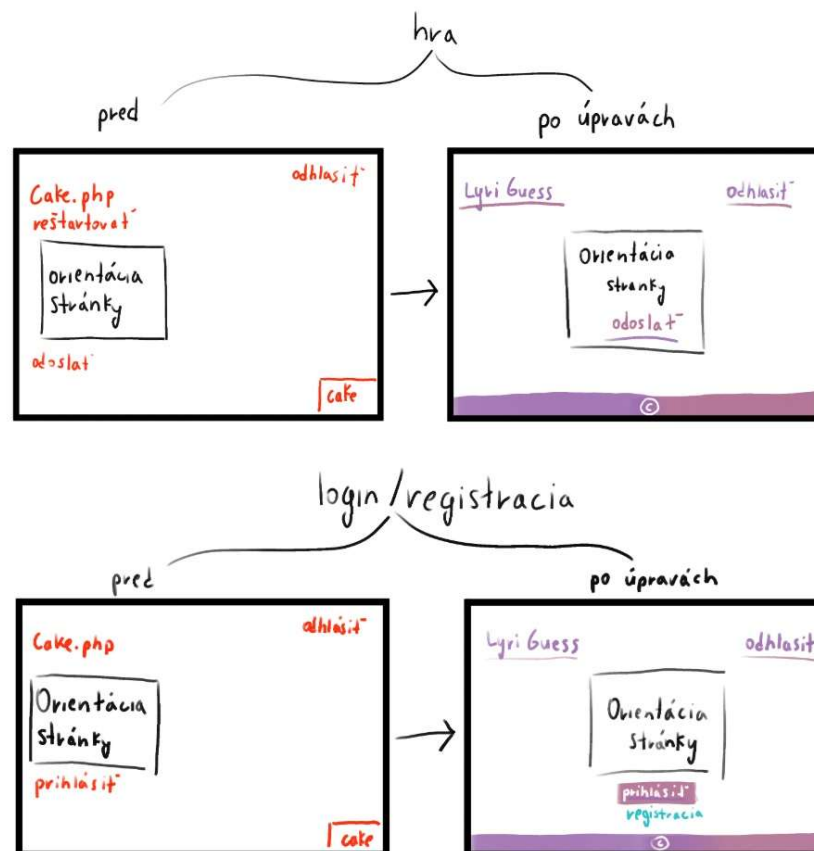


```
$points = max(value: 1, values: 6 - $attempts);  
$identity = $this->request->getAttribute(name: 'identity');  
if ($identity) {  
    $Users = $this->fetchTable(alias: 'Users');  
    $user = $Users->get(primaryKey: $identity->getIdentifier());  
    $user->score = ($user->score ?? 0) + $points;  
    $Users->save(entity: $user);  
}
```

```
if ($this->isCorrectGuess(guess: $guess, title: (string)$song->get('song_name'), artist: (string)$song->get('artist'))) {  
    $solved = true;  
    $session->write(name: "$keyBase.solved", value: true);  
  
    $points = max(value: 1, values: 6 - $attempts);  
    $identity = $this->request->getAttribute(name: 'identity');  
    if ($identity) {  
        $Users = $this->fetchTable(alias: 'Users');  
        $user = $Users->get(primaryKey: $identity->getIdentifier());  
        $user->score = ($user->score ?? 0) + $points;  
        $Users->save(entity: $user);  
    }  
  
    $this->Flash->success(message: "Správne! 🎉 Uhádol si názov alebo autora. +{$points} bodov");  
} else {  
    $attempts++;  
    $session->write(name: "$keyBase.attempts", value: $attempts);  
    if ($attempts >= 5) {  
        $locked = true;  
        $session->write(name: "$keyBase.locked", value: true);  
    } else {  
        $this->Flash->error(message: 'Nesprávne. Skús znova.');    }  
}
```



Roberta – front-end



Denný song

Pokusy: 0

Nápoveda (krok 1 z 5):

Sme už starí na rozprávky,
ale zase mladý na prehry.

Tip (názov alebo autor)

ODOSLAŤ

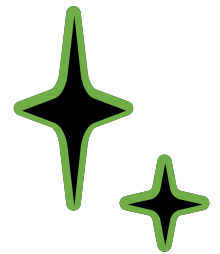


```
<nav class="top-nav">
<div class="top-nav-title">
  <div style="margin-bottom: 0.5rem;">
    <!--<?= $this->Html->image('logo.png', ['alt' => $appName]) ?> pridanie loga-->
  </div>
```

```

webroot > css > # hra.css > .hra-wrapper .button
1  /* Skryť linky / elementy s textom alebo odkazom na CakePHP */
2  .powered-by,
3  .cake-powered,
4  .cakephp,
5  footer .powered-by,
6  footer .cake-powered {
7      display: none !important;
8  }
9
10 /* Skryť odkazy vedúce na cakephp.org alebo obsahujúce "cakephp" v href */
11 a[href*="cakephp.org"],
12 a[href*="cakephp"] {
13     display: none !important;
14 }
15
16 /* Skryť DebugKit toolbar (skúsi rôzne bežné selektory) */
17 #debug-kit-toolbar,
18 #DebugKitToolbar,
19 .debug_kit_toolbar,
20 .debug-kit,
21 .cake-debug-toolbar,
22 #debug_kit,
23 .DebugKit,
24 .debugkit,
25 .debug-kit-toolbar {
26     display: none !important;
27 }
28
29 /* Zakryť DebugKit iframe alebo elementy s debug atribútmi */
30 iframe[id*="debug"],
31 iframe[src*="debug"],
32 iframe[class*="debug"],
33 [id*="DebugKit"],
34 [class*="DebugKit"]

```



Lyri Guess

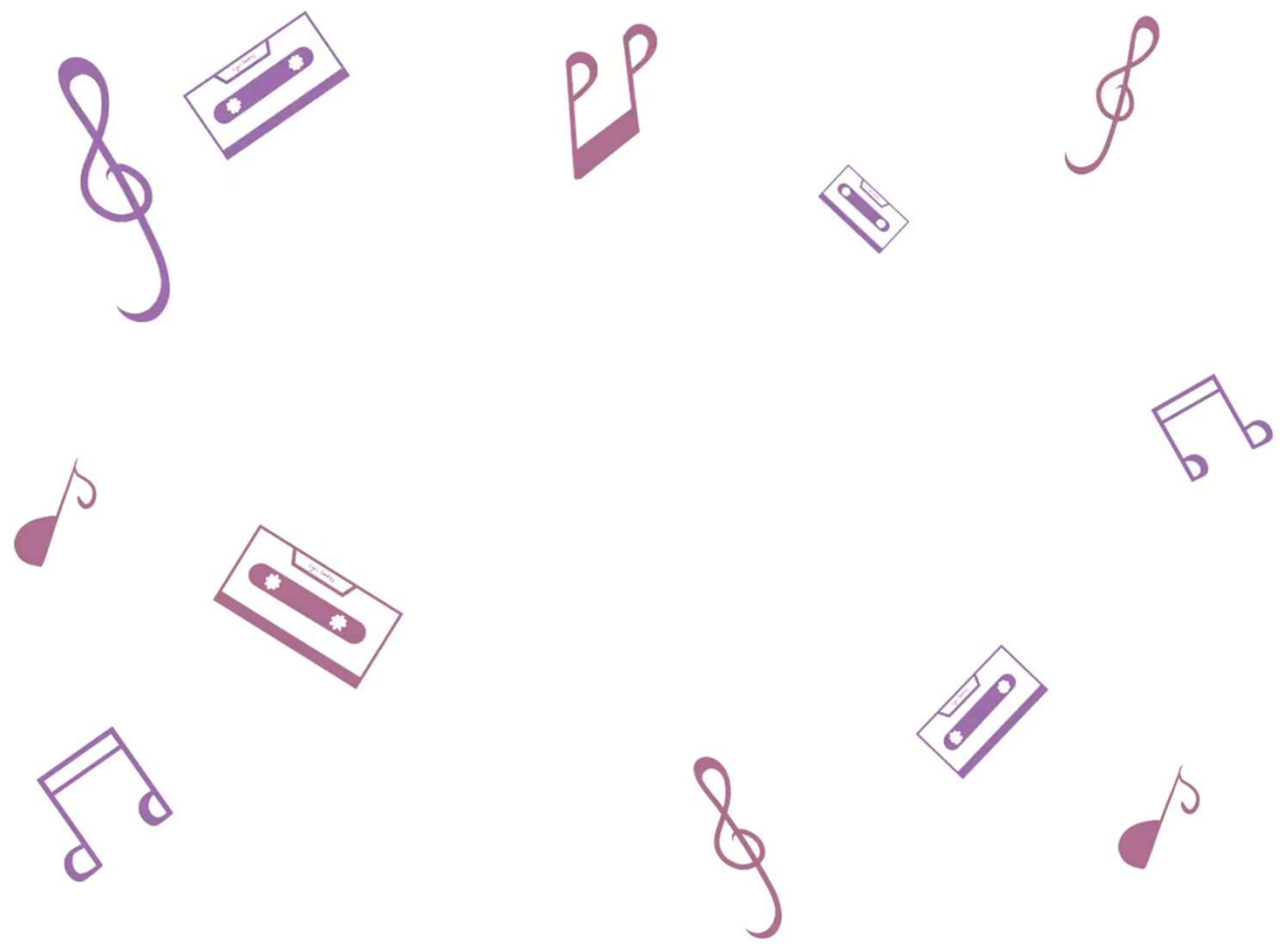
Orientácia

tlacidlá

Logo

Orientácia

tlacidlá

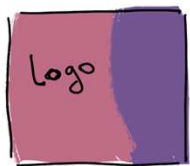


Katka – front-End



Lyriguesz?
? LYRICGUESS?
Lyri²guess
Lyric Guess

LYRI
GUES?



Vitaj v LyriGuess!
Nejaký popis... Možno Pravidlá...



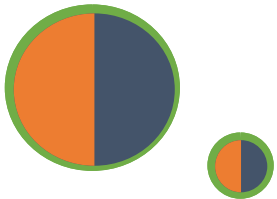
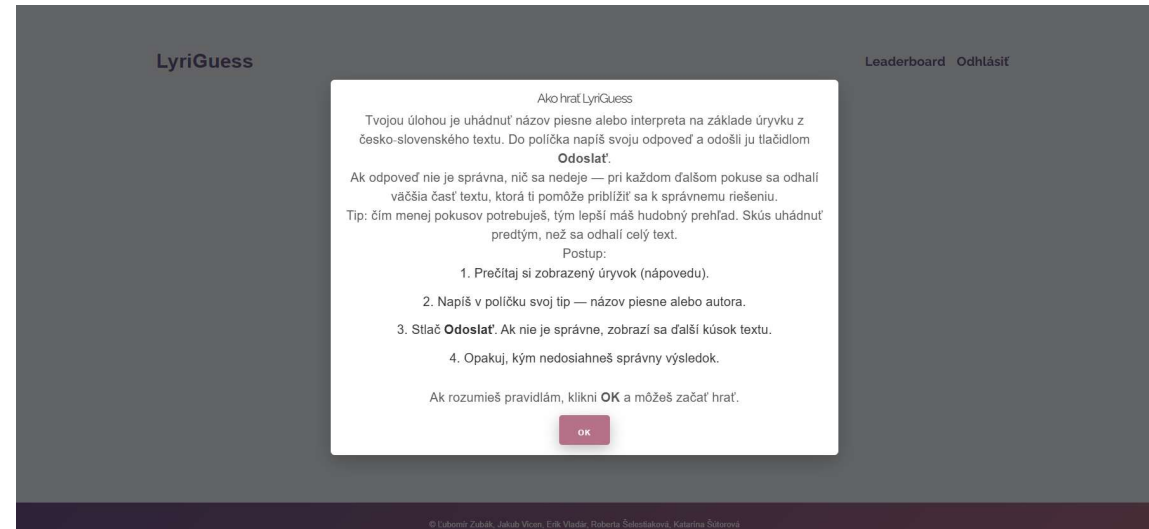
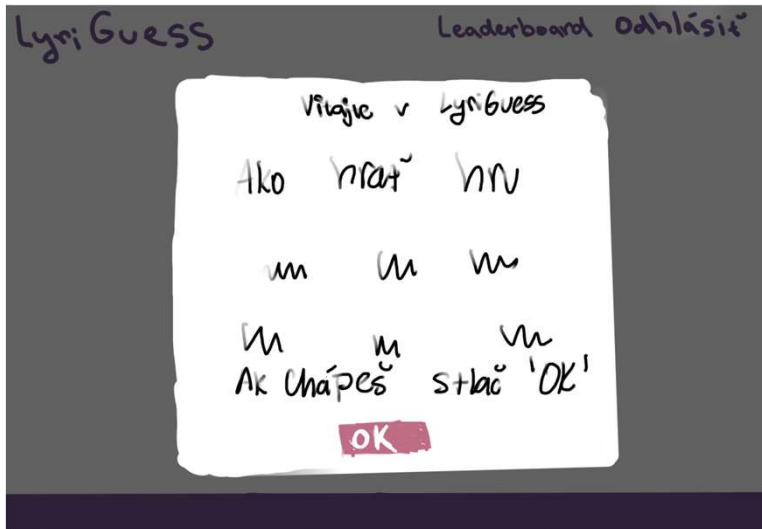
Vitaj v LyriGuess

Hádaj skladby podľa textov. Zaregistruj sa alebo sa prihlás a začni hrať.

PRIHLÁSIŤ

REGISTROVAŤ

```
<?=$this->Html->meta(['rel' => 'icon', 'type' => 'image/png', 'href' => $this->Url->build('/img/logo.png')]) ?> <!-- pridanie favicon -->
<link rel="icon" type="image/png" href="<?=$this->Url->build('/img/logo.png') ?>" /> <!-- pridanie favicon -->
```



```

< DOCTYPE html>
<html>
<head>
  <?=$this->Html->charset() ?>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>
    <?=$this->fetch('title') ?>
  </title>

  <?=$this->Html->meta(['rel' => 'icon', 'type' => 'image/png', 'href' => $this->Url->build('/img/logo.png')]) ?> <!-- pridanie favicon -->
  <link rel="icon" type="image/png" href="<?=$this->Url->build('/img/logo.png') ?>" /> <!-- pridanie favicon -->

  <?=$this->Html->css(['normalize.min', 'milligram.min', 'fonts', 'cake', 'home', 'hra']) ?>

  <?=$this->fetch('meta') ?>
  <?=$this->fetch('css') ?>
  <?=$this->fetch('script') ?>
</head>
<body>
  <div class="home-page hra-wrapper">
    <header class="hero text-center">
      <?=$this->Html->image('logo.png', ['alt' => 'LyriGuess', 'class' => 'logo', 'width' => 200]) ?>
      <h1>Vitaj v LyriGuess</h1>
      <p>Hádaj skladby podľa textov. Zaregistruj sa alebo sa prihlás a začni hrať.</p>
      <p class="actions">
        <?=$this->Html->link(__('Prihlásiť'), ['controller' => 'Users', 'action' => 'login'], ['class' => 'button outline']) ?>
        <?=$this->Html->link(__('Registrovať'), ['controller' => 'Users', 'action' => 'add'], ['class' => 'button outline']) ?>
      </p>
    </header>
  </div>
</body>
</html>

```