SŠPU STŘEDNÍ ŠKOLA PRŮMYSLOVÁ A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Revas - Aplikace pro jednoduché udělání testů

Autor: Patrik Tomaštík
Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na programování
Třída: IT4
Školní rok: 2024/25

Obsah

Poděkování

Děkuji panu učiteli Mgr. Markovi Lucnemu a Ing. Petru Grussmanovi za rady při vytváření tohoto projektu.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 6. 1. 2025

. . . . . .

Podpis autora

Abstrakt

Revas je moderní webová aplikace navržená pro efektivní vytváření, správu a sdílení testů. Aplikace umožňuje uživatelům snadno vytvářet testy v různých formátech, které mohou zahrnovat různé typy otázek, jako jsou multiple choice, krátké odpovědi nebo otevřené otázky. Revas nabízí uživatelům flexibilitu ve volbě mezi veřejnými a soukromými testy, čímž poskytuje ideální nástroj pro různé scénáře – od vzdělávacích institucí po profesionální certifikační procesy.

Klíčová slova

vzdělávání, testy, dělání testů, herní testy, zábava

Abstract

Revas is a modern web application designed for efficient test creation, management and sharing. The application allows users to easily create tests in various formats that can include different question types such as multiple choice, short answer or open questions. Revas offers users the flexibility to choose between public and private tests, providing an ideal tool for various scenarios – from educational institutions to professional certification processes.

Keywords

education, tests, taking tests, game tests, entertainment

SEZNAM POUŽITÝCH ZKRATEK

API	Application Programming Interface,
HTTP	Hypertext Transfer Protocol,
JSON	JavaScript Object Notation,
REST	Representational State Transfer,
SQL	Structured Query Language,

ÚVOD

Představení projektu

Revas je aplikace, která umožňuje vytváření testů a kvízů, ale i zábavných herních testů, které kombinují otázky a odpovědi s herními prvky. Tento nástroj je navržen tak, aby byl uživatelsky přívětivý a snadno použitelný, a to jak pro učitele, studenty, tak pro širokou veřejnost, která chce spojit zábavu a učení.

Motivace

Revas vznikl z potřeby zjednodušit proces tvorby testů a zároveň přidat do učení prvek zábavy. Motivací byla frustrace z tradičních a nudných způsobů hodnocení a učení. Chtěl jsem vytvořit nástroj, který by učitelům a lektorům umožnil snadno vytvářet personalizované testy, které zároveň zaujmou studenty. Spojení herních prvků, jako je Pexeso, s otázkami a odpovědmi, přináší nový, interaktivní způsob učení, který je motivující a zároveň efektivní. Takový přístup pomáhá zvýšit angažovanost, což je klíčové pro lepší zapamatování a pozitivní vztah k učení.

Cíle projektu

Hlavním cílem této práce je vytvořit jednoduchou apliakci pro dělání testů, které přináši i prvky zábavy.

Struktura práce

Práce je rozdělana na první část, kde je ukázána logika pro dělání testů a v druhé části je ukázána vzhled testů.

1 TEORETICKÁ ČÁST

Testování je důležitým nástrojem nejen v oblasti vzdělávání, ale i v psychologii, medicíně a dalších oblastech, kde je třeba kvantifikovat a analyzovat různé schopnosti a stavy. Testy mohou mít formu otázek s výběrem odpovědí, esejí, úkolů nebo praktických zkoušek, přičemž volba formy závisí na cíli testování a povaze zkoumané oblasti.

1.1 Historie a vznik testování

Testování, jak ho známe dnes, má dlouhou historii, která sahá až do starověkých civilizací. Již ve starověkém Egyptě, Číně a Řecku se používaly formy testů pro výběr jednotlivců na různé pozice, například pro vojenské nebo vládní služby. V průběhu staletí se metody hodnocení vyvíjely, ale teprve na přelomu 19. a 20. století došlo k modernizaci testování díky rozvoji psychologie.

1.1.1 Počátky moderního testování

Za počátky moderního testování je považován vývoj psychometrie, tedy vědy o měření duševních schopností a charakteristik. Na konci 19. století vytvořil francouzský psycholog Alfred Binet první testy inteligence. Binetův test, známý jako Binet-Simon test, byl zaměřen na identifikaci dětí, které měly potíže se školním učením, a byl základem pro vznik dnešních IQ testů.

1.1.2 Rozvoj testování ve 20. století

V průběhu 20. století se testování stalo klíčovým nástrojem nejen ve vzdělávání, ale také v pracovních, vojenských a psychologických výzkumech. V roce 1916 byl ve Spojených státech vyvinut Stanford-Binet test, který rozšířil původní Binetův test a poskytl lepší nástroje pro měření inteligence. Tento test byl použit pro širokou škálu účelů, od školních zkoušek po výběr do armády.

Ve 20. letech 20. století začaly vznikat také první standardizované testy pro hodnocení výkonu studentů v širším měřítku. Testy byly zaměřeny na akademické dovednosti a vědomosti, například matematiku, čtení nebo gramatiku. V roce 1926 byl vyvinut test Scholastic Aptitude Test (SAT), který se stal jedním z nejznámějších a nejpoužívanějších nástrojů pro hodnocení výkonu studentů v USA.

1.2 Druhy testů

V současnosti existuje mnoho různých typů testů, které se liší jak formou, tak účelem. Některé z nejběžnějších typů testů zahrnují:

- **Testy s výběrem odpovědí**: Nejrozšířenější formou testování. Studenti vybírají jednu nebo více odpovědí z předem daných možností.
- **Esejové testy**: Testy, které vyžadují, aby studenti napsali odpověď v textové formě. Tyto testy hodnotí schopnost kritického myšlení a organizování informací.
- **Praktické testy**: Používají se k hodnocení praktických dovedností, například při výuce technických nebo uměleckých oborů.
- **Standardizované testy**: Tyto testy mají pevně stanovený formát a jsou používány k měření výkonu studentů ve srovnání s širokou populací. Příkladem jsou SAT, ACT nebo GRE.
- **Formativní testy**: Cílem těchto testů není pouze hodnocení, ale i podpora učení. Jsou používány v průběhu vzdělávacího procesu, aby učitelé zjistili, jaký pokrok studenti dělají a kde je potřeba zlepšení.

1.3 Význam testování ve vzdělávání

Testování má v současném vzdělávacím systému mnoho funkcí. Mezi hlavní patří:

- **Hodnocení pokroku studentů**: Testy umožňují měřit, jak dobře studenti zvládli učivo a jaký pokrok udělali během studia.
- **Identifikace potřeb studentů**: Testy mohou pomoci identifikovat studenty, kteří potřebují další podporu nebo kteří se nacházejí nad průměrem.
- **Motivace k učení**: Testování může sloužit jako motivace pro studenty, kteří se chtějí zlepšit nebo dokázat své schopnosti.
- **Standardizace hodnocení**: Standardizované testy poskytují objektivní způsob hodnocení, který umožňuje porovnávat výkon studentů na širší úrovni.
- **Zpětná vazba pro učitele**: Testy poskytují učitelům důležité informace o tom, jak efektivně vyučují a které oblasti je třeba zlepšit.

1.4 Současné trendy v testování

V současnosti se testování stále vyvíjí a přizpůsobuje novým technologiím a pedagogickým metodám. Mezi současné trendy patří:

- **Online testování**: S nástupem digitálních technologií se testování přesunulo do online prostředí, což umožňuje flexibilitu a dostupnost testů pro širokou veřejnost.

- **Gamifikace testů**: V některých případech je testování spojeno s herními prvky, což má za cíl zvýšit motivaci a angažovanost studentů.
- **Testování na bázi analýzy dat**: Využití pokročilé analytiky a algoritmů k hodnocení výkonu studentů a přizpůsobení testů na míru.
- **Automatizované hodnocení**: Pokroky v umělé inteligenci umožnily automatizaci hodnocení testů, což zrychluje proces a zajišťuje větší objektivitu.

Z tabulky je patrné, že každá aplikace má své silné a slabé stránky. Zatímco Anki vyniká v pokročilých funkcích a statistikách, postrádá moderní uživatelské rozhraní a gamifikační prvky. Quizlet nabízí dobrou rovnováhu mezi funkcionalitou a použitelností, ale mnoho pokročilých funkcí je dostupných pouze v placené verzi. Memrise má propracovaný systém gamifikace, ale je omezen především na jazykové učení a vyžaduje stálé připojení k internetu.

2 POUŽITÉ TECHNOLOGIE

2.1 H5P

H5P je open-source knihovna, která umožňuje vytváření interaktivních a multimediálních obsahů, jako jsou testy, kvízy, prezentace, videa a další typy výukových modulů. Je zaměřena na tvorbu bohatého, interaktivního obsahu, který může být snadno integrován do různých webových aplikací, jako jsou vzdělávací platformy nebo systémy pro tvorbu testů. H5P poskytuje nástroje pro snadné vytváření a správu různých typů interaktivního obsahu bez nutnosti hlubokých znalostí programování.

Hlavní vlastnosti:

- Snadná tvorba interaktivního a multimediálního obsahu
- Podpora různých typů interaktivních aktivit (kvízy, prezentace, hry, videa)
- Flexibilita pro integraci s webovými aplikacemi a platformami (např. Moodle)
- Podpora pro exportování obsahu ve standardizovaných formátech
- Možnost integrace s analytickými nástroji pro sledování výkonnosti uživatelů

2.2 FileSaver.js (Pro ukládání testů)

Pro ukládání testů a jejich export do různých formátů byla využita knihovna FileSaver.js. Tato knihovna umožňuje uživatelům jednoduše stáhnout soubory (např. testy ve formátu JSON nebo textovém formátu) přímo na jejich zařízení, což usnadňuje práci s uloženými testy a zajišťuje jejich bezpečné uložení bez potřeby serveru pro zpracování.

Výhody:

- Možnost ukládání souborů na uživatelském zařízení bez potřeby serveru
- Snadné exportování testů a dalších dat do souborů
- Kompatibilita s různými formáty souborů (např. .txt, .json, .csv)

2.3 Moodle XML Generator

Pro převod textových formátů do formátu Moodle XML byla použita specializovaná technologie, která umožňuje automatizované generování testů, otázek a odpovědí pro platformu Moodle. Tento proces je nezbytný pro snadné publikování a správu online testů a zajišťuje efektivní exportování testového obsahu pro využití na Moodle platformě.

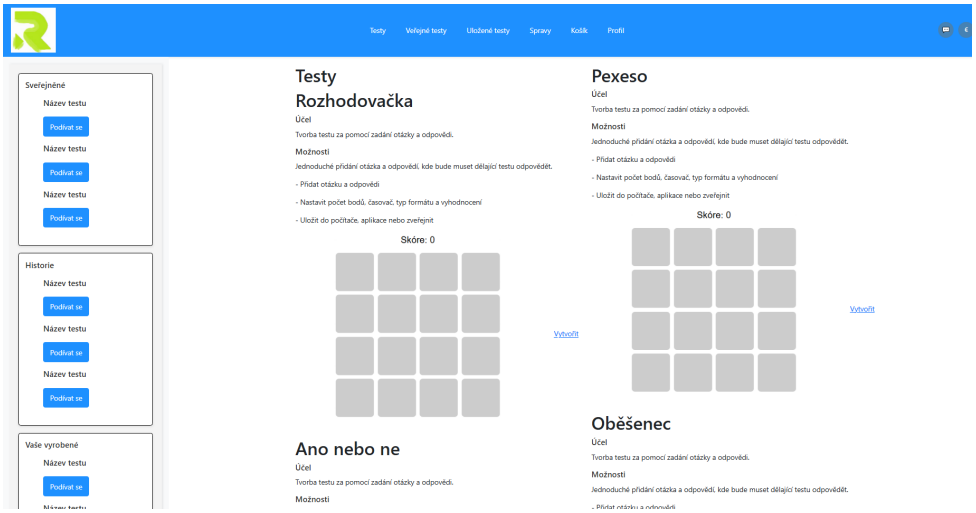
Klíčové výhody:

- Automatizované generování testových otázek a odpovědí
- Podpora pro různé typy otázek (výběr z více možností, textové odpovědi, přiřazování)
- Efektivní správa exportovaných testů pro platformu Moodle

3 ARCHITEKTURA APLIKACE

3.1 Celková architektura

Aplikaci jsem postavil na Django frontend, i když není primárně frontendový framework. Django umožňuje: templates, static files, forms.



Obrázek 3.1: Architektura aplikace

Hlavní architektonické principy:

- Lepší testovatelnost jednotlivých komponent
- Snadnější údržba a rozšiřitelnost kódu
- Jasná separace zodpovědností
- Konzistentní terminologie v celém projektu
- Škálovatelnost aplikace

3.2 Datový model

Pro reprezentaci dat v aplikaci jsem použil Django backend.

```
# models.py
from django.db import models
from django.contrib.auth.models import User

# Model pro definici typu testu
class TestType(models.Model):
# Název typu testu (např. "Matematika", "Historie"), musí být jedinečný
```

```
    name = models.CharField(max_length=100, unique=True)

    description = models.TextField()

    def __str__(self):
        return self.name


# Model pro konkrétní test
class Test(models.Model):
    # Název testu
    name = models.CharField(max_length=255)
    # Popis testu
    description = models.TextField()
    image = models.ImageField(upload_to='tests/')
    created_by = models.ForeignKey(User, on_delete=models.CASCADE)
    is_public = models.BooleanField(default=False)

    # Metoda pro zobrazení názvu testu
    def __str__(self):
        return self.name

    # Přepsaná metoda save, která se spustí při každém uložení testu
    def save(self, *args, **kwargs):
        # Nejprve zavoláme standardní save metodu pro uložení objektu
        super().save(*args, **kwargs)
        self.test_type.update_statistics()


# Model pro uchování statistik o uživatelských testech
class UserStatistics(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    total_tests_created = models.PositiveIntegerField(default=0)
    test_statistics = models.JSONField(default=dict)

    # Metoda pro zobrazení statistik uživatele
    def __str__(self):
        return f"Statistiky pro {self.user.username}"

```

```
47      # Metoda pro aktualizaci statistik o počtu testů podle typu testu
48      def update_statistics(self, test_type_name):
49          # Pokud již existuje záznam pro tento typ testu, zvětšíme počet
50          if test_type_name in self.test_statistics:
51              self.test_statistics[test_type_name] += 1
52          # Pokud tento typ testu ještě nemáme, přidáme nový typ s počtem 1
53          else:
54              self.test_statistics[test_type_name] = 1
55          self.total_tests_created += 1
56          self.save()
57
58
```

Kód 3.1: Definice datových modelů

4 IMPLEMENTACE

4.1 Frontend implementace

4.1.1 Uživatelské rozhraní

Vzhled uživatelského rozhraní je velmi důležitý, protože dává první dojmy z aplikace:

Podoba herních testů, Stránky aplikace

```
<div class="container">
    <h1>Vytvořit test</h1>
    <form id="testForm" method="post"
    enctype="multipart/form-data" action="{% url 'verejne_testy' %}">
        {% csrf_token %}

        <!-- Popisek testu -->
        <h2>Popisek k testu</h2>
        <div class="test-name">
            <label for="testName">Název testu:</label>
            <input type="text" id="testName" name="testName"
            placeholder="Zadejte název testu" required>
        </div>
        <label for="testImage">Obrázek testu (nahrát soubor):</label>
        <input type="file" id="testImage" name="testImage"
        accept="image/*">
        <label for="testDescription">Popis testu:</label>
        <textarea id="testDescription" name="testDescription"
        placeholder="Zadejte popis testu"></textarea>

        <!-- Nastavení časovače -->
        <h2>Nastavení časovače</h2>
        <label for="timer">asovač (v minutách):</label>
        <input type="number" id="timer" name="timer" min="1"
        placeholder="Nastavit časovač">
        <label for="enableTimer">Povolit časovač:</label>
        <input type="checkbox" id="enableTimer" name="enableTimer">

        <!-- Nastavení bodování -->
        <h2>Bodování</h2>
        <label for="grade1">Známka 1 (minimální body):</label>
```

```
        <input type="number" id="grade1" name="grades[1]" min="1"
        value="90">
        <label for="grade2">Známka 2 (minimální body):</label>
        <input type="number" id="grade2" name="grades[2]" min="1"
        value="70">
        <label for="grade3">Známka 3 (minimální body):</label>
        <input type="number" id="grade3" name="grades[3]" min="1"
        value="50">
        <label for="grade4">Známka 4 (minimální body):</label>
        <input type="number" id="grade4" name="grades[4]" min="1"
        value="30">
        <label for="grade5">Známka 5 (minimální body):</label>
        <input type="number" id="grade5" name="grades[5]" min="1"
        value="0">

        <!-- Otázky a odpovědi -->
        <h2>Otázky</h2>
        <div id="questionsContainer"></div>
        <button type="button" class="add-question-btn"
        onclick="addQuestion()">Přidat otázku</button>

        <!-- Ukázka testu -->
        <h2>Ukázka testu</h2>
        <button type="button" class="preview-btn"
        onclick="previewTest()">Ukázat test</button>

        <!-- Typ uložení -->
        <h2>Typ uložení</h2>
        <button type="submit"
        class="submit-btn">Uložit v aplikaci</button>
        <button type="button"
        class="submit-btn"
        onclick="saveTest('profil')">Uložit v aplikaci</button>
        <button type="button"
        class="submit-btn"
        onclick="saveTest('verejne_testy')">
        Uložit jako veřejný test</button>
        <button type="button"
        onclick="saveAsHtml()">Uložit jako HTML</button>
```

```
71          <button type="button"
72          onclick="saveAsMoodleXML()">Uložit jako Moodle XML
73          </button>
74          <button type="button"
75          onclick="saveAsJson()">Uložit jako JSON</button>
76      </form>
77
78      <!-- Náhled testu -->
79      <div id="previewTest" class="test-section">
80          <h2>Náhled testu</h2>
81          <div id="testContent"></div>
82          <div class="timer-display" id="timerDisplay" style="display:none;">
83              <p>as zbývající: <span id="timerCount"></span></p>
84          </div>
85      </div>
86
87      <!-- Výsledky testu -->
88      <div class="results" id="resultsSection" style="display:none;">
89          <h2>Výsledky testu</h2>
90          <p id="scoreDisplay"></p>
91          <p id="gradeDisplay"></p>
92          <p id="percentageDisplay"></p>
93          <p id="correctAnswersDisplay"></p>
94          <p id="partiallyCorrectDisplay"></p>
95          <p id="wrongAnswersDisplay"></p>
96      </div>
97  </div>
98
```

4.2 Backend implementace

Backend aplikace jsem postavil na Djangu v aplikaci Pycharm.

- Databáze SQL (Ukládání testů a uživatelů)
- Moodle XML Generator
- Realtime chat

4.3 Implementace Testů

Implementace přidání otázky

```
function addQuestion() {
    questionCount++;
    const questionHTML = `
        <div class="question-wrapper" id="question${questionCount}">
            <label>Otázka ${questionCount}:</label>
            <input type="text" name="question${questionCount}_text"
            placeholder="Zadejte otázku" required>
            <div class="error-message" id="errorQuestion${questionCount}"
            style="display: none;">Tato otázka musí mít text.</div>

            <label>Body:</label>
            <input type="number" name="question${questionCount}_points"
            min="1" value="1">

            <div class="answers" id="answers${questionCount}">
                <label>
                    <input type="checkbox"
                    name="question${questionCount}_correct_answer"
                    value="1">
                    <input type="text" placeholder="Odpověď 1" required>
                    <button type="button"
                    onclick="removeAnswer(${questionCount}, 1)">
                    Smazat odpověď</button>
                </label>
            </div>

            <div class="error-message" id="errorAnswer${questionCount}"
            style="display: none;">Musíte označit správnou odpověď.</div>

            <button type="button" onclick="addAnswer(${questionCount})">
            Přidat odpověď</button>
            <button type="button" onclick="deleteQuestion(${questionCount})">
            Smazat otázku</button>
        </div>
    `;
    document.getElementById('questionsContainer').
```

```
    insertAdjacentHTML('beforeend', questionHTML);
}

```

4.4 Pro ukládání otázek

Implementace pro uložení otázky.

```
// Uložení otázky
document.getElementById("saveQuestionBtn").addEventListener("click",
function() {
            const questionText =
            document.getElementById("questionInput").value;
            const answers = [
                { text: document.getElementById("answer1").value, correct:
                document.getElementById("correct1").checked },
                { text: document.getElementById("answer2").value, correct:
                document.getElementById("correct2").checked },
                { text: document.getElementById("answer3").value, correct:
                document.getElementById("correct3").checked },
                { text: document.getElementById("answer4").value, correct:
                document.getElementById("correct4").checked }
    ];

```

4.5 Náhled testů

Implementace offline režimu je založena na lokální SQLite databázi.

Klíčové aspekty offline funkcionality:

Implementace pro náhled testů.

```
// Funkce pro náhled testu
    function previewTest() {
        const testContent = document.getElementById('testContent');
        const testName = document.getElementById('testName').value;
        testContent.innerHTML = `<h2>${testName}</h2>`;
```

```
6
7           const questions = document.querySelectorAll('.question-wrapper');
8           questions.forEach(question => {
9           const questionText =
10          question.querySelector('input[name^="question"]
11          [name$="_text"]').value;
12          const options = [];
13          question.querySelectorAll('input[name$="_option1"],
14          input[name$="_option2"], input[name$="_option3"],
15          input[name$="_option4"]').forEach(option => {
16              options.push(option.value);
17              });
18              testContent.innerHTML += `
19                  <div class="test-question">
20                      <p>${questionText}</p>
21                      <ul>
22                          <li>${options[0]}</li>
23                          <li>${options[1]}</li>
24                          <li>${options[2]}</li>
25                          <li>${options[3]}</li>
26                      </ul>
27                  </div>
28              `;
29          });
30
31          document.getElementById('previewTest').style.display = 'block';
32      }
33
```

4.6 Ukládání do souborů

Implementace pro Ukládání testů do souborů.

```
1 // Funkce pro uložení jako Moodle XML
2     function saveTestToXml() {
3             const testName = document.getElementById('testName').value;
4             const testDescription =
5             document.getElementById('testDescription').value;
```

```
            const questions = gatherQuestions();

            let xmlContent = `<?xml version="1.0" encoding="UTF-8"?>
<quiz>
    <name>${testName}</name>
    <description>${testDescription}</description>`;

            questions.forEach((question, index) => {
                xmlContent += `
    <question type="multichoice">
        <name>
            <text>Otázka ${index + 1}</text>
        </name>
        <questiontext format="html">
            <text>${question.text}</text>
        </questiontext>
        <answer>
            <text>${question.correctAnswer}</text>
            <feedback>
                <text>Správná odpověď!</text>
            </feedback>
        </answer>
    </question>`;
            });

            xmlContent += `
</quiz>`;

            const blob = new Blob([xmlContent],
            { type: 'application/xml' });
            saveAs(blob, `${testName}.xml`);
        }


// Funkce pro uložení jako JSON
 function saveTestToJson() {
            const testName =
            document.getElementById('testName').value;
            const testDescription =
```

```
            document.getElementById('testDescription').value;
            const questions = gatherQuestions();

            const testData = {
                name: testName,
                description: testDescription,
                questions: questions
            };

            const blob = new
            Blob([JSON.stringify(testData, null, 2)],
            { type: 'application/json' });
            saveAs(blob, `${testName}.json`);
        }



function saveTestToHtml() {
    const testName =
    document.getElementById('testName').value;
    const testDescription =
    document.getElementById('testDescription').value;
    const questions =
    gatherQuestions(); // Funkce pro získání všech otázek

    let htmlContent = `
        <html>
            <head>
                <title>${testName}</title>
                <style>
                    body {
                        font-family: Arial, sans-serif;
                    }
                    .question {
                        margin-bottom: 20px;
                    }
                    .question label {
                        font-weight: bold;
```

```
                    }
                    .result {
                        margin-top: 20px;
                    }
                </style>
            </head>
            <body>
                <h1>${testName}</h1>
                <p>${testDescription}</p>
                <form id="testForm">
    `;

    // Generování otázek
    questions.forEach((question, index) => {
        htmlContent += `
            <div class="question">
                <p><strong>Otázka ${index + 1}:</strong>
                ${question.text}</p>
                <label for="answer_${index}">Odpověď:</label>
                <input type="text" name="answer_${index}"
                placeholder="Zadejte odpověď">
            </div>
        `;
    });


    htmlContent += `
            <button type="button" onclick="evaluateTest()">Vyhodnotit test</butto
            <div id="resultsContainer"></div> <!-- Kontejner pro výsledky -->
        </form>
        </body>
    </html>
    `;

    const blob = new Blob([htmlContent], { type: 'text/html' });
    saveAs(blob, `${testName}.html`);
}

```

4.7 Ukládání do aplikace

Implementace pro Ukládání testů do aplikace.

```
function saveTest(type) {
    const testName =
    document.getElementById('testName').value;
    const testDescription =
    document.getElementById('testDescription').value;
    const testImage =
    document.getElementById('testImage').files[0];
    // získání obrázku
    const isPublic =
    (type === 'verejne_testy') ? true : false;
    // Pokud je 'verejne_testy', test bude veřejný

    const formData = new FormData();
    formData.append('name', testName);
    formData.append('description', testDescription);
    formData.append('image', testImage);
    formData.append('is_public', isPublic);

    // Odeslání dat na server pomocí AJAX
    fetch('/api/save_test/', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        const messageElement =
        document.getElementById('responseMessage');
        // Element pro zpětnou vazbu na stránce
        if (data.success) {
            messageElement.textContent = 'Test byl úspěšně uložen.';
            messageElement.style.color = 'green'; // Zelená pro úspěch
        } else {
```

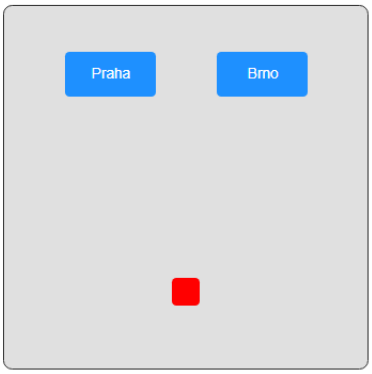
```
            messageElement.textContent = 'Nastala chyba při ukládání testu.';
            messageElement.style.color = 'red'; // ervená pro chybu
        }
    })
    .catch(error => {
        const messageElement =
        document.getElementById('responseMessage');
        // Element pro zpětnou vazbu na stránce
        messageElement.textContent =
        'Došlo k chybě při komunikaci se serverem.';
        messageElement.style.color = 'red'; // ervená pro chybu
    });
}


```

4.8 Ukazování výsledků

Implementace pro ukazování výsledků.

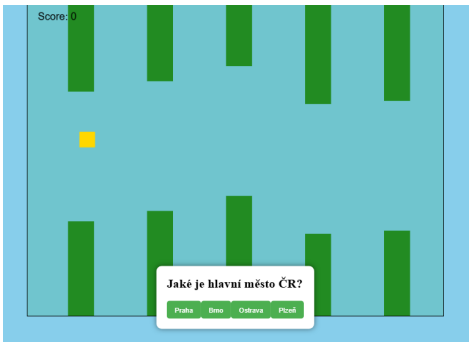
```
// Funkce pro výpočet skóre
    function calculateScore(answers) {
        totalPoints = 0;
        correctAnswers = 0;

        answers.forEach(answer => {
            if (answer === 'Ano') {
                totalPoints += 1;
                correctAnswers += 1;
            }
        });

        return correctAnswers;
    }

        document.getElementById('enableTimer').addEventListener('change', (e) =>
            if (e.target.checked) {
                totalTime = parseInt(document.getElementById('timer').value) * 60
```

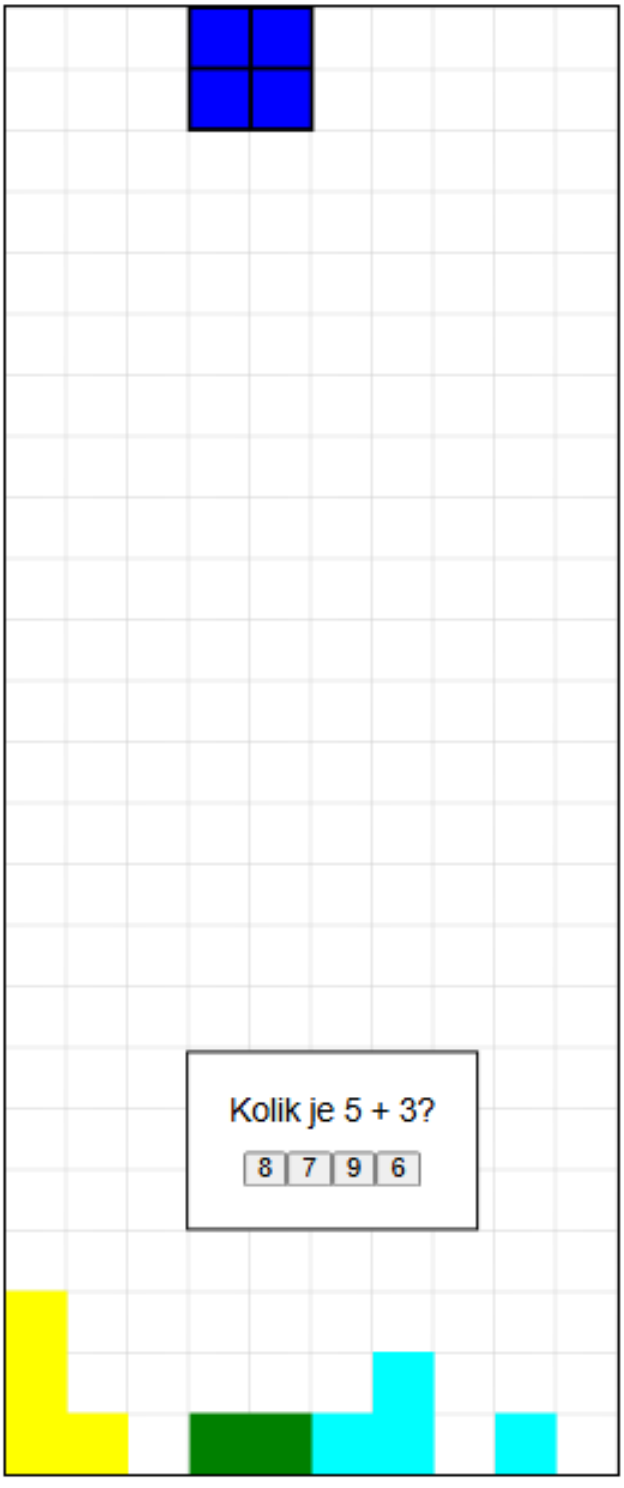
```
                document.getElementById('timerDisplay').style.display = 'block';
                startTimer();
            } else {
                clearInterval(timerInterval);
                document.getElementById('timerDisplay').style.display = 'none';
            }
        });

          function displayResults(correctAnswers, userAnswers) {
    const correctDisplay = document.getElementById('correctAnswersDisplay');
    const wrongDisplay = document.getElementById('wrongAnswersDisplay');
    let correctHTML = '';
    let wrongHTML = '';
    let correctCount = 0;
    let wrongCount = 0;

    correctAnswers.forEach((answer, index) => {
        const userAnswer = userAnswers[index];
        if (userAnswer.toLowerCase() === answer.toLowerCase()) {
            correctHTML += `<span class="correct">Otázka
            ${index + 1}: Správně - Odpověď: ${userAnswer}</span><br>`;
            correctCount++;
        } else {
            wrongHTML +=
            `<span class="wrong">Otázka ${index + 1}: patně -
            Odpověď: ${userAnswer} (Správně: ${answer})</span><br>`;
            wrongCount++;
        }
    });

    correctDisplay.innerHTML = `plně správně: ${correctCount} otázek<br>${correctHTML}`;
    wrongDisplay.innerHTML = `plně špatně: ${wrongCount} otázek<br>${wrongHTML}`;
}

```



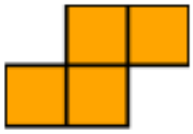
Hlavní město ČR?

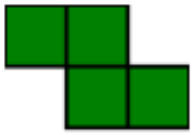
Čas: 27s





Next:





Score: 0

Test: Spojte správné páry

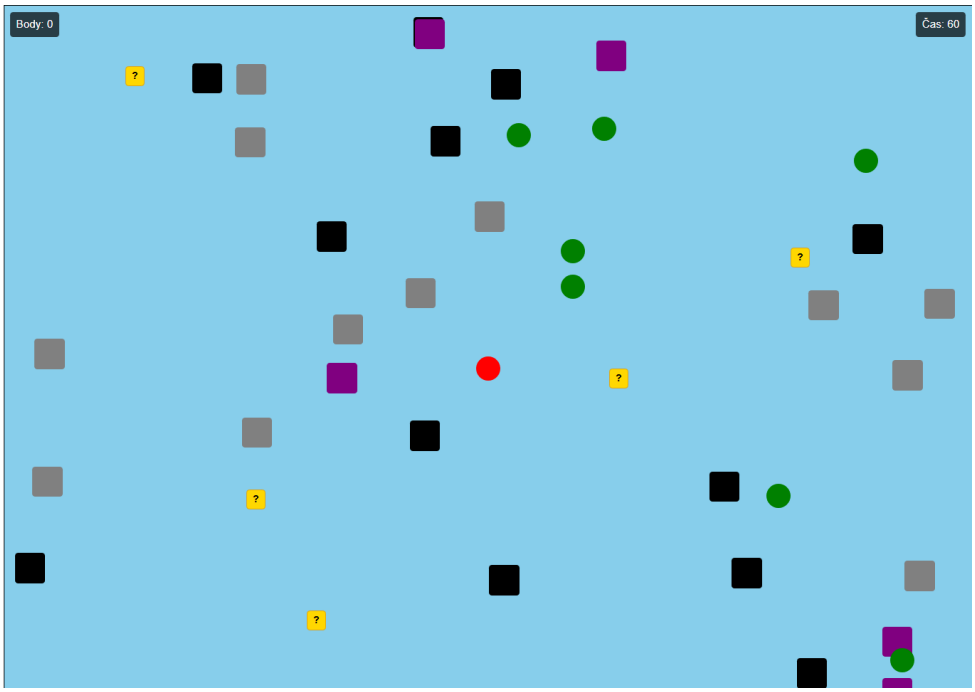
Hlavní město Německa?	Berlín
Hlavní město ČR?	Praha
Hlavní město Slovenska?	Bratislava

Odeslat odpovědi

Jaké je nebe?

Modré | Žluté | Černé | Jak to mám vědě, nahoru nekoukám

Další otázka



5 VÝSLEDKY

5.1 Splněné cíle

V rámci projektu se mi podařilo úspěšně vytvořit aplikaci pro dělání testů a k tomu udělat live chat s možností nákupu vymožeností.

5.2 Nedostatky a budoucí zlepšení

- Přidání více testů
- Přidání větší volnosti při vytváření testů
- Lépe si zorganizovat času
- Přidání dalších testů v podobě kahootu a nebo denních výzev

5.3 Přínosy

Projekt Revas úspěšně dosáhl svého cíle. Je možné vytvořit jednoduché a hravé testy.

Hlavní přínosy projektu:

- Efektivnější dělání testů.
- Zábavu
- Zkušenosti jak zacházet s časem a plánování
- Zkušenosti do budoucna

6 ZÁVĚR

Aplikace Revas představuje moderní nástroj, který výrazně usnadňuje tvorbu a správu testů v různých formách. Díky flexibilnímu přístupu a intuitivnímu uživatelskému rozhraní nabízí uživatelům širokou škálu možností pro tvorbu jak statických, tak dynamických testů, ať už ve formě otázek a odpovědí nebo gamifikovaných testů, které mohou být zábavným a efektivním způsobem učení.

Zdrojový kód projektu je na GitHubu (`https://github.com/Patrik1T/Maturitniprojekt?tab=readme-ov-file#classquiz`).

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Quiz Flask App by simboli: Open-source webová aplikace postavená na Pythonu a Flasku s MySQL databází, umožňuje vytváření a správu kvízů. Dostupné z: https://github.com/thepasterover/flask-quiz-app

[2] Quiz Flask App Open-source webová aplikace postavená na Pythonu a Flasku s MySQL databází, umožňuje vytváření a správu kvízů. Dostupné z: https://github.com/simboli/quiz-flask-app

[3] ClassQuiz: Open-source aplikace podobná Kahoot!, umožňuje učitelům vytvářet kvízy, které mohou studenti hrát na dálku. Dostupné z: https://github.com/mawoka-myblock/ClassQuiz

[4]Moosh: Nástroj pro správu Moodle z příkazové řádky, který může pomoci při exportu testů do Moodle. Dostupné z: https://github.com/tmuras/moosh