KettoLingo

Jakub Chrappa, Dávid Kurek, Boglárka Farkas, Marek Mikula, Miloš Ilovský

December 5, 2024

Contents

1 Úvod

Projekt **KettőLingo** je interaktívna aplikácia určená na efektívnu výučbu cudzích jazykov. Cieľom aplikácie je poskytovať používateľom rôzne výučbové aktivity, ktoré sú navrhnuté tak, aby im pomohli zlepšiť ich jazykové zručnosti a rozšíriť slovnú zásobu v cudzích jazykoch. Aplikácia ponúka bohatý obsah vrátane rôznych jazykových kategórií, kvízov a rôznych metód učenia, ktoré sa prispôsobujú potrebám a úrovniám používateľov.

Kettőlingo je navrhnutý tak, aby bol vhodný pre širokú škálu používateľov. Môže slúžiť ako nástroj pre študentov, ktorí sa pripravujú na jazykové skúšky, ale aj pre dospelých, ktorí sa chcú zdokonaliť v cudzích jazykoch. Aplikácia podporuje učenie sa cez rôzne metódy, ako sú kvízy, testy, a sledovanie pokroku, čo umožňuje každému používateľovi prispôsobiť si štúdium podľa svojich individuálnych potrieb.

Nezáleží na tom, či je používateľ začiatočník alebo pokročilý, Kettőlingo ponúka nástroje, ktoré zaručujú efektívne učenie a zábavu v jednom. Tento projekt sa zameriava na podporu výučby jazykov prostredníctvom moderných technológii, ktoré uľahčujú proces učenia a motivujú používateľov k dosahovaniu svojich jazykových cieľov.

2 Funkcie

2.1 Široká ponuka jazykov a kategórií

Aplikácia podporuje výuku cudzích slov v piatich jazykoch:
- Maďarčina
- Slovenčina
- Čeština
- Taliančina
- Angličtina

Slová sú rozdelené do rôznych tematických kategórií, ako napríklad cestovanie, jedlo, povolania, príroda a ďalšie.

2.2 Interaktívna výuka

Používatelia si môžu prezerať, opakovať a zapamätať slová pomocou vizuálnych alebo textových pomôcok.

Funkcie na zvýšenie efektivity učenia, napríklad:
- Flashcards na zapamätanie slov
- Kvíz

2.3 Testovanie znalostí

Po ukončení štúdia kategórie môžu používatelia absolvovať test, ktorý overí ich nadobudnuté znalosti.

Testy sú prispôsobené na mieru konkrétnej kategórii a jazyku.

2.4 Vyhodnotenie testov

Po dokončení testu aplikácia okamžite zobrazí výsledky:
- Celková úspešnosť (percentuálne skóre).
- Počet správnych a nesprávnych odpovedí.
- Zoznam chýb spolu s vysvetlením správnych odpovedí.

2.5 Profil používateľa

Profil používateľa obsahuje viaceré zaujímavé informácie:
- História testov: Každý test, ktorý používateľ absolvoval, sa uloží do jeho profilu.
- Detailné výsledky: Používateľ má prístup k podrobným analýzam svojich predchádzajúcich testov, vrátane chybovosti a správnych odpovedí.
- Sledovanie pokroku: Aplikácia umožňuje sledovať zlepšovanie sa v jednotlivých jazykoch a kategóriách na základe absolvovaných testov.

3 Front-End

Frontend aplikácie je implementovaný pomocou **React.js**, moderného JavaScriptového frameworku, ktorý umožňuje efektívne a modulárne budovanie užívateľského rozhrania. Na správu balíčkov a vývoj sa používa **Vite**, rýchly a moderný nástroj pre vývoj webových aplikácií.

3.1 Štruktúra Aplikácie

Hlavná štruktúra aplikácie je definovaná v súbore `App.jsx`, ktorý obsahuje definície trás pomocou `react-router-dom`. Tieto trasy zodpovedajú jednotlivým stránkam aplikácie.

3.1.1 Ukážka kódu pre `App.jsx`

```
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Register from './components/Register.jsx';
import Login from './components/Login.jsx';
import Overview from './components/Overview.jsx';
import Profile from "./components/Profile.jsx";
import LearningPage from "./components/LearningPage.jsx";
import QuizPage from "./components/QuizPage.jsx";

function App() {
  return (
    <Router>
      <div>
        <Routes>
          <Route path="/register" element={<Register />} />
          <Route path="/login" element={<Login />} />
          <Route path="/overview" element={<Overview />} />
          <Route path="/profile" element={<Profile />} />
          <Route path="/learn/:foreignLanguageId/:categoryId" element={<LearningPage />} />
          <Route path="/" element={<Login />} />
          <Route path="/quiz/:foreignLanguageId/:categoryId" element={<QuizPage />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

Listing 1: Definícia trás v `App.jsx`

3.2 Konfigurácia Prostredia

Frontend aplikácia využíva **Vite** na vývoj a kompiláciu. Konfiguračný súbor `vite.config.js` definuje proxy na komunikáciu s backend serverom a pridáva podporu pre React.

3.2.1 Ukážka konfigurácie `vite.config.js`

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

```

```
export default defineConfig({
  server: {
    proxy: {
      '/api': {
        target: process.env.KETTO_BE || 'http://127.0.0.1:5000',  // Flask backend server
        changeOrigin: true,
        secure: false,
        rewrite: (path) => path.replace(/^\/api/, ''),  // Optional rewrite rule
      },
    },
  },
  plugins: [react()],
})
```

Listing 2: Konfigurácia prostredia pomocou Vite

3.3 Komponenty

Každá stránka aplikácie je implementovaná ako samostatný React komponent. Nižšie sú stručné popisy hlavných komponentov:

3.3.1 Register

Komponent `Register.jsx` poskytuje formulár na registráciu používateľa. Používa metódu `POST` na odoslanie údajov na server.

3.3.2 Login

Komponent `Login.jsx` obsahuje prihlasovací formulár a overuje údaje používateľa.

3.3.3 Overview

Komponent `Overview.jsx` slúži ako hlavná prehľadová stránka, kde používateľ vidí dostupné jazykové kategórie a možnosti výučby.

3.3.4 Profile

Komponent `Profile.jsx` zobrazuje údaje o používateľovi a umožňuje ich úpravu.

3.3.5 LearningPage

Komponent `LearningPage.jsx` zobrazuje slová na učenie na základe vybranej jazykovej kategórie.

3.3.6 QuizPage

Komponent `QuizPage.jsx` poskytuje otázky pre testovanie znalostí používateľa.

3.4 HTML Šablóna

Frontend aplikácia využíva základnú HTML šablónu definovanú v súbore `index.html`.

3.4.1 Ukážka HTML Šablóny

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale
        =1.0" />
    <title>Vite + React</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com"
        crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Oi&display=
        swap" rel="stylesheet">
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Listing 3: HTML Šablóna pre aplikáciu

3.4.2 Zhodnotenie

Táto frontend aplikácia je efektívne navrhnutá s použitím React.js a Vite, čo umožňuje rýchlu a flexibilnú vývojovú skúsenosť. Integrácia s backendom cez proxy server zjednodušuje komunikáciu medzi frontend a backend časťami aplikácie.

4 Back-End

Táto časť popisuje implementáciu backendu pre aplikáciu KettoLingo. Backend je postavený na Flask frameworku a poskytuje API pre správu používateľských údajov, jazykov, kategórií, testov a ich výsledkov.

4.1 Konfigurácia

4.1.1 Prostredie

Backend využíva Flask a je nakonfigurovaný na PostgreSQL databázu:

```
app.config['SQLALCHEMY_DATABASE_URI'] = os.getenv('KETTO_DB',
  default='postgresql://postgres:postgres@localhost/KettoLingo')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

JWT autentifikácia je zabezpečená pomocou tajného kľúča:

```
app.config['JWT_SECRET_KEY'] = 'your_jwt_secret_key'
```

4.1.2 Rozšírenia

Použité rozšírenia:

- **Flask-Bcrypt** pre hashovanie hesiel.
- **Flask-JWT-Extended** pre správu JWT tokenov.
- **Flask-CORS** pre podporu CORS požiadaviek.
- **Flask-Migrate** pre migráciu databázy.

4.2 API Endpoints

4.2.1 Registrácia a prihlásenie

POST `/api/register` Registrácia nového používateľa.

POST `/api/login` Prihlásenie používateľa, vracia `access_token`.

POST `/api/logout` Odhlásenie aktuálneho používateľa, token je pridaný do zoznamu zablokovaných.

4.2.2 Správa profilu

GET `/api/profile` Získanie údajov o profile aktuálne prihláseného používateľa.

PUT `/api/profile` Aktualizácia používateľského profilu.

4.2.3 Jazyky a kategórie

GET `/api/languages` Získanie zoznamu podporovaných jazykov.

GET `/api/languages_dropdown` Získanie jazykov okrem natívneho jazyka používateľa.

GET `/api/categories` Získanie zoznamu kategórií.

4.2.4 Slová a učenie

GET `/api/learn/<foreign_language_id>/<category_id>` Získanie slov na učenie pre daný jazyk a kategóriu.

4.2.5 Testy a výsledky

GET `/api/quiz/<foreign_language_id>/<category_id>` Získanie otázok na test pre daný jazyk a kategóriu.

POST `/api/quiz_result` Uloženie výsledku testu spolu s detailmi.

GET `/api/quizzes` Získanie výsledkov testov pre používateľa, kategóriu a jazyk.

GET `/api/quiz-details/<quiz_id>` Získanie detailov výsledku testu.

Progres a známe slová

GET `/api/user_progress` Získanie progresu používateľa.

GET `/api/known_words/<user_id>/<category_id>` Získanie známych slov pre daného používateľa a kategóriu.

4.3 Bezpečnosť

JWT tokeny sú kontrolované na blokovanie pomocou nasledovného mechanizmu:

```
@jwt.token_in_blocklist_loader
def check_if_token_in_blocklist(jwt_header, jwt_payload):
    jti = jwt_payload['jti']
    return TokenBlocklistRepository.is_token_blacklisted(jti)
```

4.4 Použité služby a repozitáre

Modulárna štruktúra backendu využíva služby a repozitáre na implementáciu logiky:

- **Služby:** `AuthService`, `ProfileService`, `QuizService`, atď.
- **Repozitáre:** `UserRepository`, `QuizResultRepository`, `WordsRepository`, atď.

5 Databáza

5.1 PostgreSQL

PostgreSQL je výkonný a otvorený relačný databázový systém (RDBMS), ktorý je známy svojou stabilitou, výkonnosťou a podporou pokročilých funkcií. Je vhodný pre aplikácie rôznych veľkostí, od malých projektov až po rozsiahle systémy s veľkým objemom dát.

Medzi hlavné vlastnosti PostgreSQL patria:

- **Podpora štandardu SQL**: PostgreSQL plne podporuje SQL a poskytuje množstvo rozšírení pre komplexnejšie operácie.
- **Rozšíriteľnosť**: Používateľ môže pridávať vlastné typy dát, funkcie, operátory a indexy.
- **Integrita dát**: PostgreSQL poskytuje transakcie na úrovni ACID (Atomicita, Konzistencia, Izolácia, Trvanlivosť), čo zaručuje bezpečné ukladanie a manipuláciu s dátami.
- **Podpora pokročilých typov dát**: Databáza umožňuje prácu s JSON, XML, geografickými dátami (GIS) a ďalšími špeciálnymi typmi.
- **Podpora cudzieho kľúča a vzťahov**: PostgreSQL efektívne spravuje vzťahy medzi tabuľkami prostredníctvom cudzích kľúčov.
- **Otvorený zdrojový kód**: Komunita aktívne prispieva k rozvoju PostgreSQL, čo zaručuje pravidelné aktualizácie a širokú podporu.

V projekte KettoLingo slúži PostgreSQL ako hlavný databázový systém. Poskytuje robustnú platformu na uchovávanie a spracovanie údajov, vrátane používateľských účtov, jazykových kategórií, slov a pokroku používateľov. Vďaka svojej spoľahlivosti a výkonu je PostgreSQL ideálnou voľbou pre túto aplikáciu.

5.2 Implementácia DB

Databázová štruktúra aplikácie KettoLingo je navrhnutá na efektívne spravovanie dát súvisiacich s výučbou jazykov. Využíva PostgreSQL ako relačný databázový systém. Databáza obsahuje nasledujúce entity:

- **User (users)**: Tabuľka uchováva informácie o používateľoch, vrátane jedinečného mena (`username`), e-mailovej adresy (`email`) a hesla (`password`).
- **Category (categories)**: Tabuľka obsahuje kategórie slov, ako napríklad "základné slová" alebo "pokročilá slovná zásoba".
- **Language (languages)**: Ukladá podporované jazyky ako angličtina, maďarčina, taliančina a ďalšie.
- **Word (words)**: Tabuľka spravuje slovnú zásobu, vrátane slov v rôznych jazykoch a ich priradenie ku kategóriám.
- **UserProgress (user_progress)**: Táto tabuľka sleduje pokrok používateľov, ako sú naučené slová (`learned_words`) a skóre z kvízov (`quiz_score`) v konkrétnych kategóriách.
- **UserKnownWord (user_known_words)**: Tabuľka mapuje slová, ktoré používateľ pozná, na základe jeho aktivity v aplikácii.
- **QuizResult (quiz_results)**: Obsahuje celkové výsledky kvízov, vrátane skóre (`score`), dátumu (`date`), používateľa a jazykovej kategórie.
- **QuizResultDetail (quiz_results_detailed)**: Poskytuje detailnú analýzu výsledkov kvízov, ako sú správne a nesprávne odpovede na jednotlivé slová.

Všetky tabuľky sú navzájom prepojené cez cudzie kľúče, čo umožňuje konzistentné a efektívne spravovanie vzťahov medzi entitami. Napríklad tabuľka `user_progress` je prepojená s tabuľkami `users` a `categories`, čo umožňuje sledovať pokrok jednotlivých používateľov v rámci rôznych kategórií.

Táto databáza podporuje hlavné funkcie aplikácie, ako je registrácia používateľov, učenie slov, kvízy a sledovanie pokroku.

6 Príprava testov

6.1 Pridanie aplikácie do systémovej cesty

Pred testovaním je potrebné pridať nadradený adresár aplikácie do systémovej cesty, aby bolo možné importovať Flask aplikáciu (`app.py`) a databázové modely:

```
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(
    __file__))))
```

Fixture: `test_client`

Fixture `test_client` pripravuje testovacie prostredie pre Flask aplikáciu:

- Aktivuje režim `TESTING`.
- Nastavuje testovaciu databázu `test_db`.
- Vytvára klienta na simuláciu HTTP požiadaviek.

Kód:

```
@pytest.fixture(scope='module')
def test_client():
    app.config['TESTING'] = True
    app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://user:
        password@localhost/test_db'
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

    with app.test_client() as testing_client:
        with app.app_context():
            pass
        yield testing_client
```

6.2 Popis jednotlivých testov

6.2.1 Test: Registrácia používateľa

Úloha: Otestovať funkčnosť registrácie nového používateľa.
Kód:

```
def test_register_user(test_client):
    response = test_client.post('/api/register', json={
        'username': 'testuser',
        'email': 'testuser@example.com',
        'password': 'password123'
    })
    assert response.status_code == 200
    assert b"User registered successfully" in response.data
```

Očakávaný výsledok: HTTP kód 200 a textová odpoveď `User registered successfully`.

6.2.2 Test: Prihlásenie používateľa

Úloha: Overiť prihlásenie používateľa s platnými a neplatnými prihlasovacími údajmi.
Kód:

```python
def test_login_user(test_client):
    invalid_credentials = {
        "email": "nonexistent@example.com",
        "password": "wrongpassword"
    }
    response = test_client.post('/api/login', json=invalid_credentials
        )
    response_json = response.get_json()
    assert response_json[1] == 401
    assert response_json[0] == {'error': 'Invalid credentials'}

    valid_credentials = {
        "email": "testuser@example.com",
        "password": "password123"
    }
    response = test_client.post('/api/login', json=valid_credentials)
    response_json = response.get_json()
    assert response_json[1] == 200
    assert "access_token" in response_json[0]
```

Očakávaný výsledok:

- Pri neplatných údajoch: HTTP kód 401 a správa `Invalid credentials`.
- Pri platných údajoch: HTTP kód 200 a vrátenie `access_token`.

6.2.3 Test: Získanie jazykov

Úloha: Otestovať endpoint na získanie zoznamu podporovaných jazykov.
Kód:

```python
def test_get_languages(test_client):
    login_response = test_client.post('/api/login', json={
        'email': 'testuser@example.com',
        'password': 'password123'
    })
    token = login_response.get_json()[0]['access_token']
    response = test_client.get('/api/languages', headers={
        'Authorization': f'Bearer {token}'
    })
    assert response.status_code == 200
    assert isinstance(json.loads(response.data), list)
```

Očakávaný výsledok: HTTP kód 200 a JSON zoznam jazykov.

6.2.4 Test: Získanie kategórií

Úloha: Overiť, či endpoint vracia zoznam kategórií.
Kód:

```python
def test_get_categories(test_client):
    login_response = test_client.post('/api/login', json={
        'email': 'testuser@example.com',
        'password': 'password123'
    })
    token = login_response.get_json()[0]['access_token']
    response = test_client.get('/api/categories', headers={
        'Authorization': f'Bearer {token}'
    })
```

```
    assert response.status_code == 200
    assert isinstance(json.loads(response.data), list)
```

Očakávaný výsledok: HTTP kód 200 a JSON zoznam kategórií.

6.2.5 Test: Prístup k chráneným cestám

Úloha: Otestovať prístup k chránenému endpointu s platným tokenom.
Kód:

```
def test_protected_route(test_client):
    login_response = test_client.post('/api/login', json={
        'email': 'testuser@example.com',
        'password': 'password123'
    })
    token = login_response.get_json()[0]['access_token']
    response = test_client.get('/api/protected', headers={
        'Authorization': f'Bearer␣{token}'
    })
    assert response.status_code == 200
```

Očakávaný výsledok: HTTP kód 200.

7 Záver

Celý projekt predstavuje komplexný systém na výučbu cudzích jazykov, ktorý integruje frontend a backend do jednotného riešenia. Frontend je postavený na technológiách React.js a Vite, ktoré umožňujú moderný, modulárny a vysoko výkonný vývoj. Backend zabezpečuje spoľahlivú správu dát a funkcií, vrátane registrácie, prihlasovania, testovania a uchovávania histórie používateľov.

Hlavné výhody projektu zahŕňajú:

- **Používateľská prívetivosť**: Jednoduché a intuitívne rozhranie umožňuje rýchlu orientáciu používateľov a efektívne využitie funkcií aplikácie.
- **Flexibilita**: React komponenty umožňujú ľahké rozširovanie aplikácie o nové funkcie, pričom Vite poskytuje rýchlu spätnú väzbu pri vývoji.
- **Výkon**: Vďaka moderným technológiám je aplikácia optimalizovaná na rýchle načítanie a plynulý chod.
- **Integrácia s backendom**: Proxy server a jasne definované API zjednodušujú komunikáciu medzi frontend a backend časťami, čo zvyšuje spoľahlivosť systému.

Projekt poskytuje robustné riešenie pre výučbu cudzích jazykov, čím podporuje používateľov v rozširovaní svojich jazykových schopností. V budúcnosti je možné aplikáciu ďalej rozvíjať, napríklad pridaním ďalších jazykov, pokročilých testovacích metód alebo využitím umelej inteligencie na personalizáciu obsahu.

Táto aplikácia ukazuje, ako môžu moderné technológie prispieť k zlepšeniu vzdelávania a zároveň demonštruje silu React.js a Vite pri tvorbe komplexných webových aplikácií.