Vizsgaprojekt Dokumentáció

Szerk.: Kérészi Martin, Drávucz Márton
2025 November-December

Készült 2025/2026-ös tanév szakmai vizsgájára. HMSZC ReZsi Technikum
https://github.com/DamaszkusziRavaszbuzi/vizsgaprojekt

Tartalomjegyzék

Bevezető

Projektünk egy modern, interaktív szókincsfejlesztő webes alkalmazás oktatási célokra.
Célja a hagyományos papír alapú szókártyák kiváltása egy környezetbarátabb, digitális megoldással, amely intuitív funkcióival hatékonyabbá teszi a nyelvtanulást.

A programot készítették:
Drávucz Márton: https://github.com/DamaszkusziRavaszbuzi
- -Backend
- -Adatbázis struktúra
- -Dokumentáció

Kérészi Martin: https://github.com/kenyesz2
- -Frontend és Design
- -Dokumentáció
- -UX
- -UAT

A csapatmunkát a Discord felületén, privát csevegésben valósítottuk meg. A program tervezését, technikai megvalósításainak egyeztetését ott végeztük, illetve a megírt kódokat, kódrészleteket ott osztottuk meg egymással.
GitHub-ot csak a fejlesztés visszakövethetősége érdekében, illetve a régi verziók archiválására használtuk. Csapatmunkára az iskolában szerzett gyakorlat hiányában, illetve személyes preferenciából nem használtuk.

Projektünk során használtunk AI-t. Mivel az informatikának, legfőképp a szoftverfejlesztésnek ma már szerves része, ezért nem tartottuk érdemesnek sem mellőzni, sem pedig tagadni a használatát. A projekt során előkerülő monoton problémák kiküszöbölésére használtuk. Ilyen volt például:
- -~50 szavas angol-magyar szótár generálása demonstrációs célra (lásd:Technikai megvalósítások:Backend)
- -A backend API linkek létrehozása a HTML oldalak alapján
- -A backendben az adatbázist létrehozó kód
- -Refaktorolás

Funkciók

Programunk több, a nyelvtanulást segítő funkcióval is rendelkezik. Többek között: a felhasználó hozzáadhat szavakat és azok fordítását, ezeket virtuális szókártyák formályában megtekintheti, illetve a megszerzett nyelvi tudását ellenőrizheti.
Minden felhasználó rendelkezik egy saját “szótárral”, melyet egy adatbázisabn tárol a program.

A funkciók csoportosíthatóak a CRUD-műveletek szerint:

-Create (Léterehozás):
-Szavak hozzáadása manuálisan
-Algoritmus által véletlenszerűen választott szavak hozzáadása (részleges megvalósítás [1])
-Algoritmus által ajánlott szavak hozzáadása (részleges megvalósítás [1])

-Read (Olvasás):
-Szópárok megjelenítése szókártyák formájában
-Szavak véletlenszerű gyakorlása
-Az algoritmus által gykorlásra szorulónak ítélt szavak gyakorlása

-Update & Delete (Szerkesztés és Törlés):
-A felhasználói élény szempontjából minimális jelentőséggel rendelkeznek, ezért egy külön meüpontban kaptak helyet

Funkciók leírása részletesen:

Regisztráció/bejelentkezés:
A felhasználók egy fh.név+jelszó kombinációjával tudnak regisztrálni, illetve bejelentkezni. A jelszó módosítására bejelentkezés után lehetőség van a beállításokban. A gecis

titkoítás nem akart működni, ezért a jelszavak egyszerű szövegként vannak tárolva.

Beállítások:
A beállítások menüpontban módosítható a jelszó, illetve a UI téma.

Új szó hozzáadása:
A felhasználó manuálisan hozzáadhat a szótárához egy szót és annak fordítását.

Új random szó:
A program egy véletlenzerűen kiválasztott szó+fordítás párost ajánl fel a felhasználónak, amit ő elfogadhat/elutasíthat. Technikai okokból ez csak részben, demonstráció képpen valósult meg.[1]

Új ajánlott szó:
A program a felhasználó jelenlegi szavait figyelembe véve ajánl új szavakat a felhasználónak, amit ő elfogadhat/elutasíthat.[1] Technikai okokból ez csak részben, demonstráció képpen valósult meg.[1]

Random:
A program a felhasználó szótárából véletlenszerűen választ ki szavakat, melynek a fordítását a felhasználónak kell kitalálnia, ezzel fejlesztve a szókincsét.
Ebben a módban a felhasználónak lehetősge van:
- -Visszatérni a főmenübe
- -Segítséget kérni:
  - -Feleletválasztós
  - -Magánhangzók megjelenítése
- -Feladni
- -Megfordítani a fordítás irányát
- -Ellenőriznie a megoldást

Gyakorlás:
A program nyomon követi a felhasználó tudását, hogy melyik szavakat ismeri [1]. Az algoritmus a gyakorlásra szoruló

szavakat részesíti előnyben, működése megegyezik a “Random” funkcióval.

Kártyák:

A felhasználó szótárából véletlenszerűen kiválasztott szavak virtuális kártyák formályában kerülnek megjelenítésre. Ezeket a felhasználónak lehetősége van mozgati, illetve felfordítani, ezzel mimikálva a valódi, fizikai szókártyákat. A felhasználó törölhet, illetve hozzáadhat a megjelnített kártyákhoz.

Szavak kezelése:

A felhasználó örölheti, illetve módiíthatja a szótárában szereplő szavakat.

Statisztika:

A felhasználó megtekintheti a program által alkotott statisztikát a felhasználó szókincsérél.

[1] Lásd: Technikai megvalósítások fejezet

Technikai megvalósítások: Frontend

A frontend HTML, CSS, illetve JavaScript-ben készült.
A program kizárólag webes felülettel rendelkezik. Elsősorban asztali felhasználásra készült, de rendelkezik mobilon is használható, egy natív alkalmazással megegyező élményt nyújtó, reszponzív felülettel, ezzel eleget téve a projektvizsgára vonatkozó követelményeknek.

Több okból döntöttünk a kizárólag webes felület mellett:

- -Platformfüggetlen
- -Széles körben elterjedt
- -Könnyen létrehozható dinamikus felület
- -A programnak nincs szüksége a hardveres erőforrások közvetlen elérésére
- -Nincs szükség külön alkalmazás letöltésére
- -Bárhonnal elérhető
- -5 tanév során nem tanultunk se mobilalkalmazás, se asztali GUI alkalmazás fejlesztést ¯_(ツ)_/¯

A frontend fejlesztés, design során fontos szempont volt a UX (felhasználói élmény). Több design döntés is segíti a program intuitív használatát:

- -Kontrasztos színek
- -Nagyméretű gombok
- -Ikonok helyett egyértelmű szövegek

A UI tervezése közben nagy hangsújt fektettünk a fizikai szókártyák egyszerűségének megőrzésére. Fölösleges “gimmick” funkciók helyett egy egyszerű, letisztult felhasználói felület.

A frontend JavaScript restAPI-on keresztül kommunikál a backend-del, használva a HTTP metódusokat (GET, POST).

Technikai megvalósítások: Backend

A backend Pythonban íródott, a Flask keretrendszert használva. A backendhez egy SQL adatbázis van csatolva.

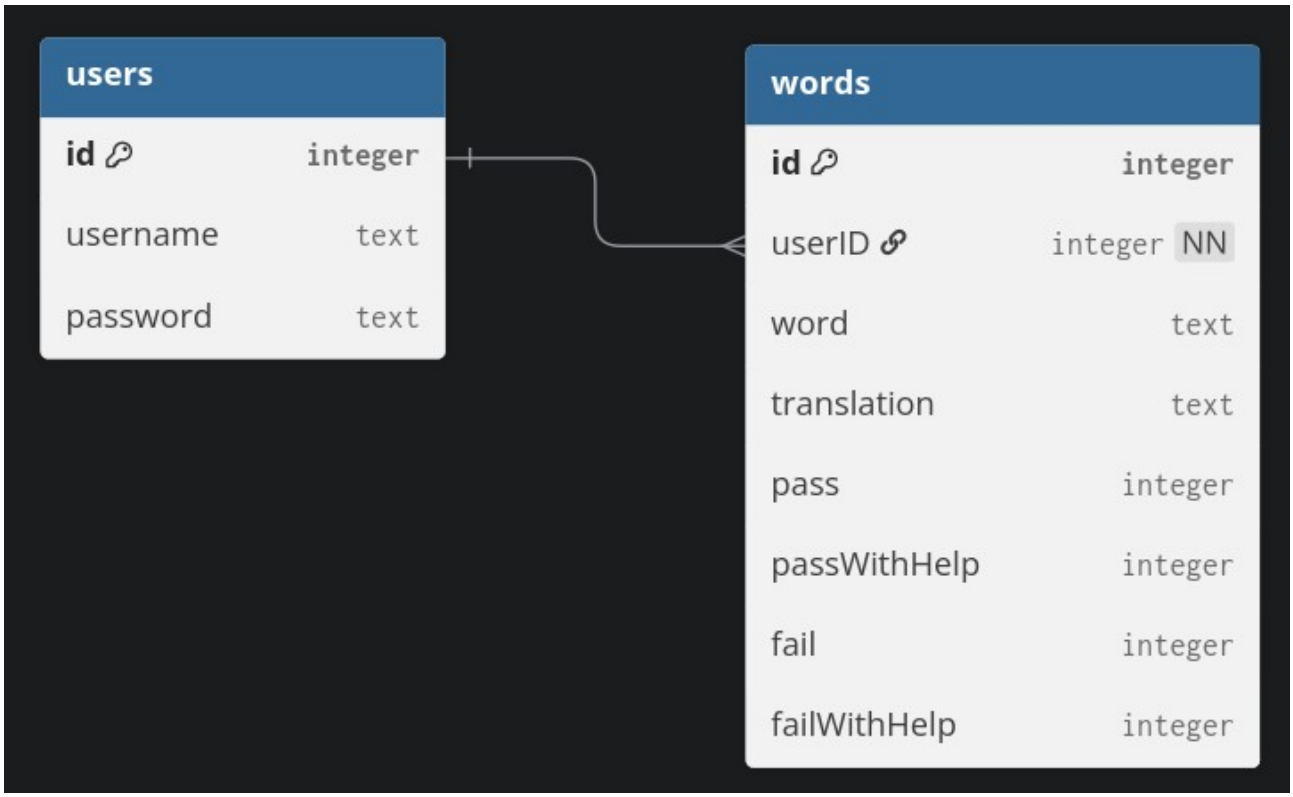
A felépítése javarészt követi a szabványos Flask backendek struktúráját - pl.:

```
///
@app.route('/login', methods=['POST'])
     def login():
          …
///
```

-így ezt nem érdemes részletezni.

Adatbázis:

Az adatbázis struktúráját ez a fasza relációs ábra jól szemlélteit:



A felhasználók és a szavak külön-külön táblákban vannak tárolva. Mindekettő esetében egy egyedi ID a PK, a userID alapján vannak összekapcsolva.

Lényegesebb algoritmusok:

“Magabiztosság” (confidenceIndex) kiszámítása:

A program nyomon követi a felhasználó válaszainak helyességét a Random és Gyakorlás módban. A felhasználó válaszait 4 értékelésbe lehet besorolni:

- **Pass**: A felhasználó jó választ adott meg
- **PassWithHelp**: A felhasználó jó fálaszt adott meg segítséggel
- **Fail**: A felhasználó helytelen választ adott meg vagy feladta
- **FailWithHelp** A felhasználó helytelen választ adott meg vagy feladta segítséggel

Ezeket a backend számon artja az adatázisban (lást: fentebbi ábra), és ezek alapján számol ki egy “magabiztossági indexet” (a programban: confidenceIndex, továbbiakban: **cI)**, mely megközelítőleg megmutatja, hogy a felhasználó mennyire ismeri az adott szót, mennyire biztos a tudásában.
A következő képlettel lehet kiszámítani:
cI = pass*2+passWithHelp-fail-failWithHelp*2
Egy helyes válasz 2, a segítséggel megadott helyes válasz 1, a hibás megoldás -1, illetve a hibás megoldás segítséggel -2 “pontot” ér.
Ez a változó a Gyakorlás módban rendelkezik jelentős szereppel.

Gyakorlás:

A negatív cI-vel rendelkező szavakat kiválogatja a backend. Ha 5-nél több ilyen szó van, azokat átadja a frontendnek, ami azokat váletlenszerű sorrendben “gyakoroltatja” a felhasználóval. Minden szó csak 1x kerül kiválasztása a frontenden. Ha elfogytak a szavak, a backend újra kiszámítja a cI-ket, majd a ciklus kezdődik előről.

Amennyiben 5-nél kevesebb negatív cI-vel rendelkező szó van a negatív cI-vel rendelkező szavak kiválasztásra kerülnek, a kimaradt szavak pedig a cI-jük alapján növekvő sorrendben rendeződnek. A lista elejéről (alacsony cI) addig kerülnek szavak kiválasztásra, amíg a kiválasztott szavak száma el nem éri a 10-et. Ekkor a szavak ismét átadásra kerülnek a frontendnek, …

Új random szó:

Eredeti tervünk az volt, hogy egy lokális adatstruktúrából, mely tartalmazza (közel) az összes angol szót, egy szó véletenszerűen kiválasztásra kerül, egy külső API azt lefordítja, majd a szó-fordítás pár ajánlásra kerül a felhasználónak. A felhasználó elfogadhatja/visszautasíthatja a szó hozzáadását a saját szótrába.

A terv gyakorlati kiviteleése sajnos megbukott, mivel nem találtunk érdemlegesen használható, ingyenes, kulcs nélküli API szolgáltatást, mely eleget tett volna a feltételeknek. Illetve egy ilyen fordító API szolgáltatás “self-hosting”-ja pedig további problémákat jelentett volna.

Szóba került még egy lokális szótár használata erre a célra, azomban érdemleges, használhat adatstruktúrával rendelkező angol-magyar szótárat szintén nem találtunk

Demonstrációs célból a program ezen része egy lokális, ~50 szóból álló szótárral valósult meg.

Új ajánlott szó:

Ez a programrész a felhasználó meglévő szavai alapján tud új szavakat ajánlani. Ehhez a meglévő szavak hosszát veszi figyelembe; a hossz alapján különíti el az egyszerűbb szavakat (hi, and, if, car,…) a bonyolultabb szavaktól (inconspicuous, detrimental, revelation,…), habár ez egy meglehetősen felületes módszer, hiszen nagy az átfedés, egy szó “bonyolultságát” nem feltétlen tükrözi a hossza, illetve a felhasználó szókincse sem mindig becsülhető meg ezzel a módszerrel.

Az algoritmus kiválaszt 10 véletlenszerű szót, majd amelynek a hossza a legközelebb esik a meglévő szavak átlaghozzához leforításra kerül.

Az előző részben részletezett technikai okok miatt ez is csak demonstráció képp került megvalósításra.