**EÖTVÖS LORÁND TUDOMÁNYEGYETEM**



Informatikai Kar

Média- és Oktatásinformatika Tanszék

ConjúgaMe - spanyol

nyelvfejlesztő kvízprogram

|  |  |
| --- | --- |
| **Témavezető:** | **Szerző:** |
| Turcsányi-Szabó Márta | Tóth Márton Antal |
| egyetemi docens | programtervező informatikus BSc. |

Budapest, 2022

# Kivonat

A szakdolgozat tématerülete a spanyol nyelvtanulás fejlesztése kvízjátékok segítségével. Az elkészült programban kétféle kvíz található: az egyik az igeragozás gyakorlását segíti elő, míg a másik a főnevek, illetve melléknevek magyarról spanyolra fordításával foglalkozik. A felhasználó mindkét játékmódot a saját igénye szerint testre szabhatja. Normál játékmód esetén meg kell adnia, hogy összesen hány szót szeretne majd kidolgozni a kvíz során, míg időzített esetén a megadott idő lejártáig tart a játék.

Több gyakorlást segítő paraméter beállítása mellet szükséges legalább egy szótár kiválasztása is a kvízek elkezdéséhez, mivel a program a bennük található szavakból generálja a feladatsorokat. A szótárak egy lokálisan elérhető adatbázis tábláiból származnak.

A kvízjátékok végén a felhasználó megtekinthet egy részletes kiértékelő táblázatot az elért pontszámmal, a hibásan megadott szavak helyes változataival, valamint egyéb, teljesítménnyel kapcsolatos statisztikákkal. A pontszámok a játékos fejleszteni való területeivel együtt mentésre kerülnek és később visszanézhetők. A program az első kitöltött kvíztől kezdve a játékos hibái alapján állítja össze az újabb feladatsorokat, előnyben részesítve azokat a szavakat, amelyekre nem, vagy kevesebb helyes válasz érkezett a korábbiakban.

Tartalomjegyzék

[Kivonat 2](#_Toc122349903)

[Bevezetés 6](#_Toc122349904)

[Felhasználói dokumentáció 12](#_Toc122349905)

[Rendszerkövetelmények 12](#_Toc122349906)

[Hardveres követelmények 12](#_Toc122349907)

[Szoftveres követelmények 12](#_Toc122349908)

[JDK (és JRE) telepítése 13](#_Toc122349909)

[Telepítés 13](#_Toc122349910)

[Első lépések és a főmenü 15](#_Toc122349911)

[Kvíz preferenciák 16](#_Toc122349912)

[Igeragozás kvíz preferenciái 17](#_Toc122349913)

[Szófordítás kvíz preferenciái 19](#_Toc122349914)

[Preferenciák mentése 21](#_Toc122349915)

[Igeragozás kvíz 21](#_Toc122349916)

[Szófordítás kvíz 23](#_Toc122349917)

[Kitöltési konvenciók 24](#_Toc122349918)

[Kvíz kiértékelő panel 24](#_Toc122349919)

[Eredmények 26](#_Toc122349920)

[Pontszám táblázat 26](#_Toc122349921)

[Átnézésre váró igék és fordítások 27](#_Toc122349922)

[Bélyegek 29](#_Toc122349923)

[Beállítások 30](#_Toc122349924)

[Az alkalmazás beállításai 31](#_Toc122349925)

[A kvízjáték beállításai 32](#_Toc122349926)

[Ismert hibák és kezelésük 33](#_Toc122349927)

[Egyszerűbb hibák 34](#_Toc122349928)

[Súlyos hibák 35](#_Toc122349929)

[A program eltávolítása 35](#_Toc122349930)

[Fejlesztői dokumentáció 37](#_Toc122349931)

[Az implementáció megtervezése 37](#_Toc122349932)

[Dependenciák 38](#_Toc122349933)

[Fejlesztői környezet 39](#_Toc122349934)

[A konfigurációs fájlok 39](#_Toc122349935)

[Külső adatszerkezetek (lokális táblák) 40](#_Toc122349936)

[Igeragozás kvízhez kapcsolódó táblák 41](#_Toc122349937)

[Szófordítás kvízhez kapcsolódó táblák 42](#_Toc122349938)

[Eredményeket eltároló táblák 43](#_Toc122349939)

[Belső adatszerkezetek (modellek) 44](#_Toc122349940)

[Ige és szó kvíz komponensek 46](#_Toc122349941)

[Igék (Verb és VerbBasic osztályok) 49](#_Toc122349942)

[Szavak/fordítások (Word osztály) 50](#_Toc122349943)

[Az alkalmazás kerete és a statikus osztályok 52](#_Toc122349944)

[MainWindow 52](#_Toc122349945)

[Input/Output config 53](#_Toc122349946)

[Database absztrakt osztály 54](#_Toc122349947)

[Vezérlők és nézetek 55](#_Toc122349948)

[Dashboard és MenuButton 56](#_Toc122349949)

[QuizSetup, QuizPreferences 58](#_Toc122349950)

[A kvízpanelek és vezérlőik 60](#_Toc122349951)

[EndQuiz és QuizResults 63](#_Toc122349952)

[Eredmények és vezérlőik 65](#_Toc122349953)

[Beállítások és preferenciáik 66](#_Toc122349954)

[JUnit tesztek 67](#_Toc122349955)

[Böngészőbe integrált változat (WebSwing) 68](#_Toc122349956)

[Executable és telepítő 69](#_Toc122349957)

[Összefoglalás 70](#_Toc122349958)

[További fejlesztési lehetőségek 71](#_Toc122349959)

# Bevezetés

Az idegen nyelvek tanulásával kapcsolatos módszertanok között mára már rendszeresen megjelenik a technológiai eszközök alkalmazása. Könnyen és gyorsan fellelhető több népszerű, valamint – bár gyakran csak részben, de - ingyenes weboldal és alkalmazás, amely az ezen a területen való készségfejlesztést garantálja. A legtöbb viszont egy általam hátrányosnak ítélt tényező miatt nem feltétlenül tudja teljesen kimeríteni a szerepét; ez a tényező pedig az egy felületen több idegen nyelv gyakorlásának az egységesítése.

Lassan másfél éve, hogy egy hallgatói mobilitási programnak köszönhetően az Ausztriában töltött félévemből hazaérkezve beiratkoztam egy spanyol nyelviskola kezdő tanfolyamára. Az új, külföldi baráti kapcsolatok fenntartásáért és egy megmagyarázhatatlan, hirtelen jött érdeklődés miatt elmélyedtem az alapvető szókincs, illetve nyelvtani ismeretek gyarapításában. Ezt nemsokára már előszeretettel folytattam tanórán kívül is, kikapcsolódás gyanánt (bármennyire is megrémisztett, hogy fiatalabb korom egyik legkellemetlenebb kényszerű elfoglaltsága befurakodott a szabadidős tevékenységeim közé), végül ennek okán kezdtem el kipróbálni az elismertebb nyelvtanító alkalmazásokat.

Ezt követően az életemben egy olyan többhónapos időszak vette kezdetét, ahol a korábbi hobbijaimat felváltották ezek az ígéretes alkalmazások spanyol felületei, és ahol a gyakori beszédtémáim egyike lett az ezekről való eszmecsere. Igyekeztem hűséget fogadni egy oldal mellett, de mindegyikkel kapcsolatban elhangzottak erős ellenérvek: voltak könnyed megoldási módszerekkel operálók, prémium verzió nélkül bántóan limitált kapacitásúak, valamint túlságosan agresszívan kampányoló applikációk is, amennyiben a felhasználója elhanyagolta őket egy időre. Mégis valahogy mindegyik el tudta adni magát kellően megbízhatónak és hitelesnek a szememben arra a pár hónapra, amíg még nem sikerült felmérnem, hogy szinte semmit nem tettek hozzá a nyelvi szintfejlődésemhez, „csupán” szórakoztattak (ami pedig továbbra is csak rosszul esett).

Mire elért ez a felismerés, már valamennyire tájékozott voltam a spanyol nyelv felépítésével. Tisztában voltam vele, hogy - akár csak az anyanyelvemen -hétköznapi beszélt, vagy írott környezetben nem futok össze gyakran az igék ragozatlan alakjaival. Így bár sokat segített a szótári alakok betanulásában mind a nyelviskola, mind a különféle kvízoldalakra feltöltött spanyol szókártyás játékok kitöltése, a nyelvtudásom alkalmazásakor ritkán tudtam ezt a tudást ténylegesen felhasználni. Különösen a reméltnél gyakrabban felbukkanó szabálytalan ragozásnak köszönhetően. Ez vezetett rá arra, hogy nem előnyös már a nyelv ezen alapvető sajátossága miatt sem egy felületen (vagyis többé-kevésbé egy teljesen homogén környezetben) vezetni például az angol nyelvtanítást a spanyol mellett. Ennek alátámasztására további szemléletek is születtek bennem, amelyekkel kapcsolatban ez a szakdolgozat sajnos a már említettnél is jóval kevesebbet kell, hogy foglalkozzon. (Egy másik, hogy az olvasott spanyol szöveg adekvát kiejtési készlete miatt a beszédfejlesztés intenzitása jelentős mértékben elmaradhat az angol, vagy a franciáéhoz képest. A népszerűbb, több különböző nyelvet oktató platformok tehát kevésbé foglalkoznak azzal, hogy sokkal gyorsabban redundánssá válik az „olvasd fel a következő párbeszédet” feladatok javarésze a spanyol esetében.)

A korábban említett szabálytalan igeragozásból adódó nyelvi gát jelentette a legkiemelkedőbb problémát számomra a fejlődésben. Ezt mindenképp egy erre kifejlesztett gyakorló alkalmazás beszerzésével szerettem volna orvosolni, de a főbb app store-ok kínálatai csak fizetős szolgáltatásokat soroltak fel. A megoldás hiánya végül elkísért az alapszakom utolsó félévéig is, egészen pontosan egy unalmasabb délutánig, amikor elkezdett nyomasztani egy gondolat. Nem lehettem egyedül ezzel, de nem tartottam már számon, hogy hány programot készítettem el beadandónak, viszont pontosan tudtam, hogy abból hánynak volt utóélete, azaz, hogy tényleg szüksége lett volna rá bárkinek is a későbbiekben. Ez adta meg az elhatározást, hogy elkészítsem a megoldást a magam bajára, és egy ingyenes, platformfüggetlen program keretein belül gyakorolhassam az igeragozást.

Közben újabb fejlesztésre váró területek bukkantak fel a nyelvtanulásom során, amelyekhez szintén igyekeztem feladatokat implementálni a *¡ConjúgaMe!* névre keresztelt első komolyabb hangvételű projektemben. Kevésbé jelentős, de korai tanulási fázisokban előforduló probléma tud még lenni a nem triviális végződésű szavak nemének eltalálása, ami a melléknevek egyeztetése miatt kulcsfontosságúvá is válhat a megértésben. Ennek fejlesztésére született meg a másik játékmódja a programnak, amely igék helyett magyarról spanyolra fordítandó szavakkal dolgozik.

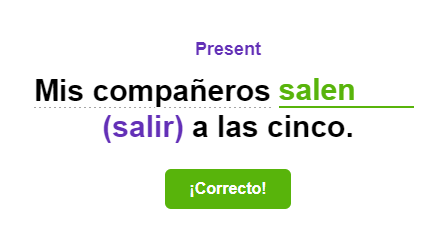
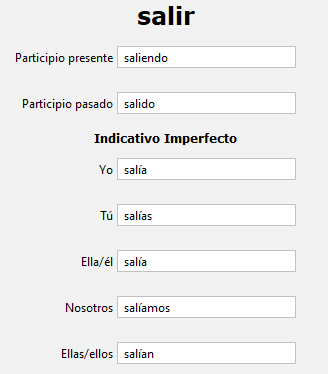
Igyekeztem a pontos időt elfelejteni, de emlékeim szerint pár nap telt el a szakdolgozati témabejelentőm leadása után, amikor felfedeztem a SpanishDict névre hallgató angol nyelvű weboldalt. Az egyik legfontosabb tulajdonságát az oldalnak már a neve is sugallja: kizárólag a spanyol nyelv fejlesztésével foglalkozik, a lehető legsokrétűbb módszereket és feladatötleteket bevetve. A mai napig lelkesen használom, bármilyen kétélű is a kapcsolatom vele. Az oldal „*Conjugation Drill*” menüpontjában részletes paraméterezést követően egy igeragozásra fókuszáló kvíz indul el, amelyben példamondatok állítmányait kell a megfelelő alannyal és igeidővel egyeztetve megformálni. Ennek az oldalnak a felfedezése után gyakorlatilag megsemmisültnek éreztem az addig innovatívnak vélt ötletemet. A csalódottság persze idővel alábbhagyott, szerencsére sikerült több érvet is találnom, hogy megtarthassam az elköteleződést a projektem megvalósításában.

A spanyol-angol és spanyol-magyar szókincsfejlesztő kvízek használatából eredő tapasztalataim alapján a nyelvtanulás folyamata jóval kevésbé rögös, ha az anyanyelvünkre fordítjuk le az új kifejezéseket, még akár egy problémamentesen és napi szinten használt, de másodlagos nyelv helyett is. Továbbá országunk sajnos gyakran az elsők között mutatkozik be az idegen nyelveket egyáltalán nem beszélők népesség szerinti arányát felsoroló diagramokon.

Ezek alapján megmaradt az elképzelés, hogy rajtam kívül még felbukkanhat valaki Magyarországon, aki tényleges szükséget tanúsíthat a program felé. Lehet akár a kényelmesebb nyelvtanulási folyamat miatt, de a spanyol első idegen nyelvként való felvétele miatt is (személy szerint ez egy különösen praktikus döntés lehet, de tényleg túlzás lenne erről is írnom egy oldalt).

A SpanishDict, bár feltételezem, hogy globálisan az egyik legátfogóbb ingyenes nyelvtanuló felület a spanyol nyelv minden egyes szegletét érintve, tudott mutatni számomra pár olyan gyermekbetegséget, amelyeket figyelembe véve jobban kiélezhettem a kvízjátékomat az igeragozás gyakoroltatására, mint azt eredetileg elképzeltem. A legelső ilyen szembejövő jelenség a „felesleges” módok kiválasztásának lehetősége volt (ezek a módok természetesen csak a jelenlegi kontextusban vehetőek annak, a nyelvben annál inkább használatban vannak).

A spanyol nyelven belül fellelhető igei módok számosságára több válasz is létezik, nem csak a nyelvjárásokból adódó sajátosságok miatt, hanem mert különböző nézőpontok alapján ebbe a kategóriába sorolhatóak az ún. *perifrázis igék* is. Míg az angolban a folyamatos múlt/jelen/jövőnek nevezett nyelvtani formákat igeidőknek tanultam, addig a spanyolban – ahol ugyanezek szinte teljesen azonos formában fellelhetők -, mindig vitatottnak éreztem, hogy hova szabad őket besorolni. Leginkább azért, mert a folyamatos módot jelző segédige és további, az állítmány körülményeinek pontosításához alkalmazott praktikák aligha bírnak a „tényleges” igeidők és -módok egy gyakori tulajdonságával; azaz kevésbé körültekerten, a verbális perifrázisok nincsenek tele irreguláris alakokkal. Ezen igemódok használatának elnyújtott gyakoroltatása tehát könnyen válhat szárazzá, szükségtelenné. Ennélfogva több módot is nélkülözhetőnek ítéltem a kvíz testreszabhatóságának kialakítása közben. Ez természetesen nem jelenti azt, hogy teljesen elhanyagoltam őket az alkalmazásból, csupán kevesebb (szerintem elegendő, de egyelőre nyelvészi jóváhagyásra vár) hangsúly marad rajtuk.

Egy további egyértelmű különbség is felbukkan a két felület összehasonlításakor. Míg a SpanishDict egyszerre egy megoldást vár, azaz összesen egy alanyt és módot ad fel a megadott ige konjugálásához, addig a ConjugaMe-ben a feladatonként elvárt megoldások száma bővíthető akár az összes, a nyelvben megtalálható alanyhoz egyeztetett alakra is.

. ábra: egy példa a 'salir' ragozatlan ige várt konjugált alakjaira a két alkalmazásban

Ez elsőre a túlzott morfémaismétlés és felesleges billentyűleütések érzetét is keltheti a felhasználóban. Sajnos erre a problémára nincs még kiforrott ellenérvem, vagy state-of-the-art megoldási javaslatom (viszont elkerülhetővé tettem, a felhasználó bármikor a SpanishDict felületéhez hasonlóra módosíthatja a kvízjátékot).

Ellenben úgy érzem, kifizetődő befektetéssé is válhat a sok billentyűleütés, amiért a több alannyal való gyakorlás során kirajzolódhatnak kisebb-nagyobb összefüggések a különböző konjugációk között. A programom tesztelése során fedeztem fel például, hogy tőhangváltós igék esetén a többes szám első és második személyében is hasonló „dupla rendellenesség” zajlik le szimultán a legtöbb igemódban. Az ezen összefüggésekre való rámutatás és/vagy a megmagyarázásuk nem hiszem, hogy gyakori jelenségnek számítanak az általános nyelvoktatási környezetben, leginkább azért, mert kellő nyelvhasználat után tudat alatt is felbukkanhatnak. Az előbb említett módon való igeragozás-gyakorlás az alkalmazásomban pedig előidézheti korábbra ezt a felbukkanást.

Az édesanyám előtti tiszteletből megírt lingvisztikai erőfeszítéseimet szerettem volna a bevezetésbe sűríteni, hogy a szakdolgozat további része kizárólag az programmal foglalkozzon. A következő fejezetben a szoftver telepítési útmutatója után igyekszem konyhanyelven taglalni a felhasználói felület minden egyes szegletét, valamint a program lehetséges felhasználási módjait. Ezt követően a GUI mögött rejlő rozsdamentes acélból fabrikált mérnöki csodát, avagy a programom szerény kis felépítését szeretném elemezni az összes tátongó kátyújával és azok befoltozásaival egységben. Ez magába foglalja majd a telepítőcsomag tartalmazta executable alatti osztályokat és dependenciákat, az adatbázisban tárolt táblákat és lekérdezéseiket, valamint a konfigurációs fájlokat. Ha szakmailag titulálnom kellene azt a szekciót, kellő önreflexióval az „izgalmas erőlködés” szópárt választanám.

# Felhasználói dokumentáció

A képen szöveg, edények, clipart, tányér látható

Automatikusan generált leírás

. ábra: az alkalmazás logója

A ConjúgaMe egy kisméretű, egyszerű, letisztult felületű asztali alkalmazás. A weboldalam (*taughtmarci.web.elte.hu*) meglátogatásával szerezhető be a legkönnyebben, ahonnan letölthető a hordozható (*portable*), illetve Windows operációs rendszer használata esetén a telepíthető változata is. A Java alkalmazások futtatásához szükséges komponenseket leszámítva más programoktól és interneteléréstől is teljesen függetlenül használható, éppen ezért igen sok környezetben futtatható.

## Rendszerkövetelmények

### Hardveres követelmények

* CPU: legalább 1.0 GhZ, X86-os család
* RAM: legalább 256 MB
* Internetelérés: nem szükséges
  + Az adatbázis frissítések és egyéb online információk eléréséhez viszont ajánlott a programot internetcsatlakozás mellett használni. Amennyiben erre nincs lehetőség, a program indításkor felajánlja az offline üzemmódot.
* Minimum 650x650-es felbontású monitor
* Billentyűzet (magyar, vagy spanyol kiosztással), egér

### Szoftveres követelmények

* Operációs rendszer:
  + Microsoft Windows 10, 11
  + MacOS X 19.8.3 és felette
  + MacOS 11, 12
  + Oracle Linux 7, 8
  + Ubuntu Linux 17.04 és felette
  + Bármely, az alább jegyzett verziószámú JDK és JRE környezetekben futtatni képes operációs rendszer
* Java Runtime Environment (*JRE*) 17.0.3, vagy újabb verzió
* Java Development Kit (*JDK*) 17.0.3.1, vagy újabb verzió

## JDK (és JRE) telepítése

Amennyiben az alkalmazást futtató számítógépen még nincs, vagy egy korábbi verziója található meg a Java-nak, internetkapcsolattal gyorsan beszerezhető egyszerre mind a két szükséges programcsomag. Ehhez elég a JDK-t a Java hivatalos weboldaláról (oracle.com/java) letölteni.

A lap alján található *Java products and technologies* szekcióhoz görgetve, majd a *Java SE* fülön belül a kiemelt *Download Java now* gombra kattintva egy új oldalon megjelenik az összes támogatott operációs rendszerre elérhető letöltési link. A megfelelő operációs rendszer fül kiválasztása után érdemes egy *installer*-re végződő elemet keresni a listában, amely letöltés után feltelepíti az összes szükséges összetevőt a számítógépre.

*Megjegyzés: 2022 november végére megjelent egy nagyobb Java frissítés (19), amely a leírások alapján nem tartalmaz semmilyen összeférhetetlenséget a korábbi verziókkal, így az alkalmazás is hibátlanul kell, hogy fusson rajta. Az ideális környezet, amelyben az ConjúgaMe meg lett írva, természetesen továbbra is a 17-es verzióra érvényes, de a későbbi újdonságok és ráncfelvarrások miatt vélhetőleg az alkalmazás számára is bekövetkezik a verzióváltás, így mindenképp érdemes a frissített Java-t választani.*

## Telepítés

A *portable* változat beszerzése esetén ez az alfejezet hanyagolható. A tömörített fájl kicsomagolása után elég megnyitni a célterületen található mappát és alkalmazás azonnal elindítható.

*Fontos: Kicsomagolás után a mappában található összes összetevőre (almappák, ismeretlen kiterjesztésű fájlok) szükség van! Az alkalmazás nem fog elindulni, ha az exe fájl, vagy bármelyik másik elem részlegesen át lett helyezve. Másolás/áthelyezés esetén az összes fájlt tartalmazó mappát érdemes mozgatni, így biztosan nem válnak szét a benne található elemek.*

A képen szöveg látható

Automatikusan generált leírásHosszútávú használat esetén a Windows felhasználóknak érdemesebb a telepítőt választani. Telepítés után könnyebb és egyértelműbb a program indítása, valamint kisebb az esélye, hogy elkeverednek a fájlok. Ez a verzió természetesen rendelkezik *uninstaller*-el is, azaz telepítés után is bármikor törölhető az alkalmazás az összes összetevőjével együtt.

3. ábra: a ConjúgaMe telepítője

A *ConjugaMe\_setup.exe* elindítása és a telepítő nyelvének kiválasztása után (ahol a spanyol is megjelenik, mint opció) el kell fogadni a licenszszerződést. A következő oldalon (3. ábra: a ConjúgaMe telepítője*3. ábra*) lehet megadni a telepítés helyét (módosítás nélkül a telepítő alapértelmezetten a *Program Files* mappába készíti el a program könyvtárát), majd pedig engedélyt adni a parancsikonok elhelyezésére a Start menüben és az asztalon.

Ezután a telepítő kicsomagolja a programot és az összetevőit a megadott mappába, végül pedig kiírja az első használat előtti ajánlott tudnivalókat (*README* *szövegfájl*).

## Első lépések és a főmenü

A képen szöveg látható

Automatikusan generált leírásAmennyiben a telepítés sikeres volt, az asztalra helyezett parancsikonra, vagy a kicsomagoláskor/telepítéskor megadott elérési útvonalon található *ConjugaMe.exe* fájlra kattintva el is indítható az alkalmazás. Rövid időn (legkésőbb pár másodpercen) belül egy új ablakban megjelenik a program főmenüje.

4. ábra: a program főmenüje alapértelmezett módban

*Megjegyzés: internetkapcsolat hiánya esetén egy párbeszédpanel fogad, amely felkínálja az offline mód lehetőségét. Ennek elfogadásával ugyanúgy elindul a program, és semmilyen hátránnyal nem jár a használat során! Kizárólag a későbbiekben elérhető frissítések automatikus letöltése miatt van az alkalmazásnak szüksége internetre.*

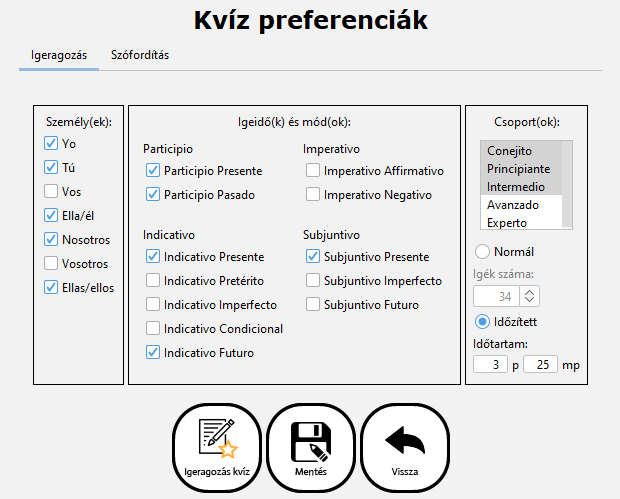
*Fontos! Amennyiben bármilyen oknál kifolyólag sikertelen volt az indítás, egy hibaüzenet jelenik meg róla. Ebben az esetben az általános javaslat az alkalmazás újratelepítése, de előfordulhat, hogy más megoldásra van szükség. A konkrét hibaüzenetekről, valamint az okokról és megoldásokról a dokumentáción belüli utolsó szekcióban, az „Ismert hibák és kezelésük”-ben (32. oldal) található több információ.*

A főmenüben, ahogy azt a *4. ábra* is mutatja, hat menüpont található: az első kettőre, azaz az igeragozás/szófordítás kvízek egyikére kattintva rögtön ki is próbálható a játék az alapértelmezett preferenciákkal beállítva.

Az alsó sorban az *Eredmények* menüponton belül van lehetőség az átnézésre váró (azaz kvízjátékok során elhibázott) szavak, valamint a korábban befejezett kvízek kiértékeléseinek megtekintésére. A *Beállítások*on belül az alkalmazással, valamint a kvízekkel kapcsolatos átfogóbb módosítások (pl. megjelenési mód megváltoztatása, feladatsorokon való továbblépés billentyűje, eredmények visszaállítása stb.) találhatóak. A *Kilépés* opció homologikus.

Az első használat alkalmával érdemes az eddig még nem említett *Kvíz testreszabása* menüpontot választani. A kattintás után megjelenő oldalon körülnézve tanulmányozható, hogyan is épül fel pontosan a két kiválasztható kvízjáték. Az itt található elemeket a következő alfejezet taglalja.

## Kvíz preferenciák

Ezen az oldalon lehet egyénileg módosítani, hogy milyen preferenciák alapján generáljon feladatsorokat a program. Rögtön a felirat alatt található a játékban fellelhető két kvíztípus neve, ezekre kattintva választható ki az azok felépítését tartalmazó panel.

. ábra: az igeragozás kvíz testreszabása

### Igeragozás kvíz preferenciái

Az igeragozással kapcsolatos kvíz egyedül igékkel foglalkozik, azoknak is kizárólag a ragozatlan (avagy szótári; pl. *ir, venir*) alakjait írja ki a játék alatt. A játékosnak ezen alakok konjugáltjait kell helyesen megadnia, viszont a nyelv sajátosságából adódóan egy szónak több, mint száz konjugáltja is lehet. A preferenciák panelen sokat lehet ebből szűrni, hogy ténylegesen az éppen gyakorolni kívánt alakokra és igékre kérdezzen rá a kvíz.

Érdemes a középső *(Igeidő(k) és mód(ok))* oszloppal kezdeni a testreszabást, amiért itt lehet kiválasztani, hogy pontosan melyik mód(ok) gyakorlására van szükség. Mivel nincs hivatalos magyar fordítása a legtöbb igeidőnek, az eredeti nevükön jelennek meg a listában. Ezek közül a bal oldalon az első két opció egy alanytól független ragozást követel meg:

A *Participio Presente* a hivatalos neve a *gerundio* alaknak; ebbe az alakba kell helyezni az igét pl. a folyamatosság jelzése esetén (*estoy yendo, lleva trabajando).* Amennyiben kijelölésre kerül, egy plusz szövegmezőt ad hozzá minden egyes feladathoz: ide csak az ige *gerundio* alakját kell majd beírni (segédigék nélkül).

A *Participio Pasado* szintén egy széleskörűen alkalmazott ragozás. Az összes *befejezett* igeidő ebben a formájában követeli meg az állítmányt *(hemos hecho, habíais llamado)*. Az előzővel egyetemben egy további szövegmezőt kap minden egyes feladott ige ezen elem kijelölésekor, és itt is csak a megfelelő alakba tett igére van szükség a feladatsor kitöltése során.

A többi csoport (*Indicativo, Imperativo* és *Subjuntivo*) a kijelentő, felszólító, illetve a kötőmódot és annak igeidőit sorolja fel. A kvíz az ezeken a csoportokon belüli kijelölések összessége alapján rendel majd hozzá minden egyes igéhez egy igeidőt. Ezek esetében már mindenképp szükséges az alanyt is egyeztetni.

*Megjegyzés: az utolsó mondatból adódóan, amennyiben csak a Participo Presento vagy a Participio Pasado van kijelölve, az első (Személyek) oszlop akár teljesen üresen is hagyható. Amint a három fő igemód csoportjai közül is kijelölésre kerül legalább egy elem, ahhoz minimum egy személyt is társítani kell.*

6. ábra: példa a csak Participio-t választott játékmódra

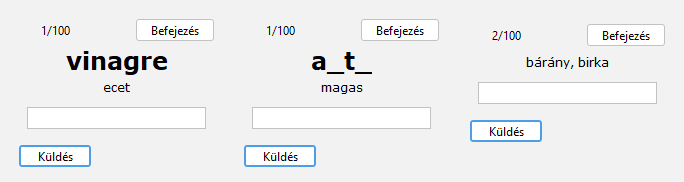
A kiválasztott igeidők után a személyek kiválasztása lehet a következő lépés. Itt megtalálható a nyelv összes lehetséges személyes névmása, beleértve a leginkább csak spanyol nyelvterületen használt *Vosotros*, illetve Latin-Amerika több területén is előforduló *Vos* formákat is. Fontos kitérni arra, hogy az itt kijelölt elemek egyenként plusz egy szövegmezőt adnak hozzá minden egyes kiosztott igeragozáshoz, így minél több személyes névmás kerül kiválasztásra, annál hosszabb lesz a kvíz kitöltésének ideje (valamint annál több pont is szerezhető).

Az utolsó oszlopban választható(k) ki a gyakorolni kívánt igecsoport(ok), azaz a szótár(ak), ahonnan a program kiválasztja majd a kvíz során megjelenő igéket. A lista alatt a játékmódot lehet módosítani: *Normál* mód esetén számszerűen megadható, hogy hány ige konjugálására kérdezzen rá a játék (az itt megadott számnál a kitölteni való szövegmezők száma sokkal több is lehet majd, a kiválasztott személyek és *Participio*-k számának függvényében). Az *időzített* mód ennél valamennyivel kompetitívebbé teszi a játékot. Ebben a módban a program a megadott perc és másodperc leteltéig engedi a kvízkitöltést, ilyenkor csak a játékostól függ, hogy hány igeragozást fog tudni elvégezni. *Normál* módban az igék száma 1 és 100 között kell, hogy legyen, míg *időzített*ben az időtartam 1 és 5 perc 59 másodperc között mozoghat.

### Szófordítás kvíz preferenciái

. ábra: a szófordítás kvíz testreszabhatósága

Az oldal második panelén, a szófordítás kvíz terén kevesebb módosítható összetevő van. A jobb oldali oszlop teljesen megegyezik az igeragozás kvízben látottakkal, annyi különbséggel, hogy itt főneveket és mellékneveket tartalmazó szótárakat sorol fel a lista. Bár az eredeti csoportok, amelyeket az alkalmazás beépítetten tartalmaz, azonos nevekkel rendelkeznek mindkét kvíz típus esetén, ez nem feltétlenül kell, hogy igaz maradjon későbbi szótárak hozzáadása esetén. A két játékmód is teljesen azonos, viszont egy említést érdemlő aspektus, hogy a szófordító kvíz esetén egyenként kizárólag egy fordításra váró szó, azaz egy szövegmező kerül terítékre. Így *normál* mód esetén a megadott szavak számát érdemesebb magasabbra (az igeragozás kvíztől itt eltérően, akár maximum 250-re is lehet), míg *időzített* módban az időtartamot ajánlott rövidebbre állítani az igeragozás kvízéhez képest, ha elvárás a hasonló mennyiségben történő szókidolgozás.

A bal oldali oszlopban található *nehézségi szint* szekció a fordításhoz adott segítség mennyiségét, illetve milyenségét adja meg *(8. ábra)*. Tanuló módban a magyarul megadott szavak és szinonimák mellett azok spanyol megfelelőjét is kiírja az alkalmazás, így ebben a módban elég a képernyőről lemásolva begépelni ugyanazt a szót. Ez a teljesen idegen szavak megismerését segíti elő. A három ezt követő fokozat már nem írja ki a megoldást, de segít megtippelni a hosszát alsóvonásokkal, illetve a nehézségi szinttől függően valamennyi betűt is megmutat a szóból. A *nincs segítség* kiválasztása esetén a magyar szavakon kívül egyáltalán nem jelenik meg segítőkész információ a várt spanyol szóval kapcsolatban.

8. ábra: szófordítás kvíz (balról jobbra) tanuló, közepes és segítség nélküli nehézségi fokozatokon

A nehézségi fokozat átállítása nem befolyásolja a pontszerzést semmilyen téren, viszont a kvíz után a jegyzett eredmények között a pontszám és egyéb statisztikák mellett látható lesz a beállított fokozat szintje is.

Egy másik beállítható paramétere a szófordítás kvíznek a névelők kihelyezésének szükségessé tétele. A szoftver kezdetleges állapotában ez a lehetőség ki van kapcsolva, de a szavak nemhez párosításához, illetve a melléknevek felismeréséhez egy különösen hasznos kellékké válhat. Amennyiben a *Névelők: szükségesek* közti jelölőnégyzet ki van pipálva, a kvíz során minden egyes főnév elé be kell gépelni az *el/la* névelők megfelelő változatát. Ha a főnév mindkét nem mellett használható (pl. *el/la estudiante*), akkor a program feltételezi, hogy a játékos ezzel tisztában van, és mindkét lehetséges opciót elfogadja. Amennyiben ilyen esetben még egyeztetni kell a szóvégi magánhangzót, akkor csak a jól párosított megoldások lesznek elfogadva (pl. *la hermana, el hermano – elfogadottak; el hermana, la hermano – hibásak)*. A melléknevek elé helyezett névelők hibásnak lesznek megjelölve, hiszen csak nagyon specifikus kontextusban fordul elő, hogy ezek nyelvtanilag nem helytelen szerkezetet eredményeznének. A névelők megfelelő helyeken és formákban történő kitétele természetesen ezen opció kikapcsolt állapotában sem okoznak hibás kiértékelést, ahogy a hiányuk sem. A pontszámítást ennek a paraméternek a ki/bekapcsolt állapota sem befolyásolja.

### Preferenciák mentése

A két panel állapota megmarad a kettő közötti váltások során is, de egy panel összetevőinek testreszabása után **a módosítások kizárólag akkor lépnek érvénybe**, ha az alsó sorban található három gomb közül az első kettő valamelyikére érkezik kattintás.

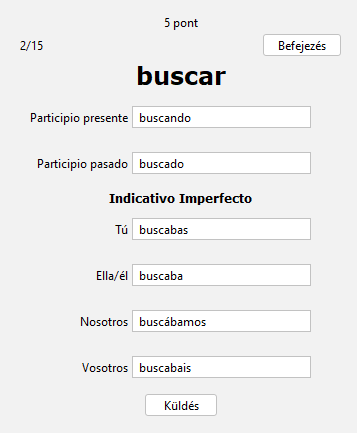
. ábra: a szófordítás preferenciái panel alsó sora

A megfelelő típusú kvíz gombját választva a módosított preferenciákkal azonnal el is kezdhető a játék. A *Mentés* gomb menti a konfigurációk közé a módosításokat, de nem lép tovább a jelenlegi oldalról. A *Vissza* gomb megnyomására a program visszalép a főmenübe és mindkét kvíz preferenciái visszaállnak a jelenlegi módosítások nélküli, korábbi állapotukba.

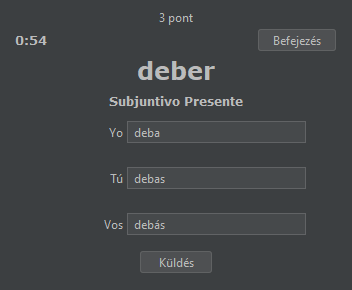
*Fontos: a felület nem engedi elindítani a kvízt, amennyiben a preferenciák beállítása valamilyen oknál fogva nem volt sikeres. Ilyenkor a mentési próbálkozás után a felület felső részén megjelenő hibaüzenettel figyelmeztet a mentés sikertelenségére, valamint kiírja a problémát, ami ezt okozza (pl. nincs egy szótár sem kiválasztva, vagy túl nagy a megadott időintervallum az időzített módhoz). Sikeres újrapróbálás után ez is jelzésre kerül, zölden szedett „Sikeres mentés!” üzenettel.*

## Igeragozás kvíz

A kvíz elindításakor a legutóbbi mentett preferenciák alapján generál feladatsorokat a program, majd az így készült listából egy véletlenszerűen kiválasztott igével kezdve várja a felhasználó megoldásait. A megoldást váró bemeneti mezők egymás alatt helyezkednek el (amennyiben több is van). A várt megoldás egyértelműsítésére helyezkednek el mellettük az alakok vagy személyes névmások *(10. ábra)*.

 *Normál* módban a bal felső sarokban található a jelenleg feladott ige sorszáma az összesből. *Időzített* mód esetén itt a hátralévő idő látható *(****Hiba! A hivatkozási forrás nem található.****)*. Felül középen, amennyiben az azonnali visszajelzések engedélyezve vannak (erről több információ a *Beállítások* alfejezetben olvasható a 29. oldalon), megjelenik az eddig elért pontszám, jobb oldalt pedig, amennyiben a játékos korábban befejezné a kvízt, egy erre szolgáló gomb helyezkedik el. (A játék egyébként korai befejezés esetén is kiértékelésre kerül, ha az addig megszerzett pontszám nagyobb, mint 0.)

10. ábra: példa az igeragozás kvíz kitöltéséhez

 Ha kiválasztásra került legalább egy *Participio* alak, akkor az közvetlenül a ragozatlan ige alatt helyezkedik el. Ezt követően kerül kijelzésre a csak megadott alanyokra megoldható aktuális igeidő (tehát a *Participio*-kon kívül az összes többi opció, ha lett kiválasztva legalább egy), majd soronként minden egyes bejelölt személyenként egy-egy szövegmező.

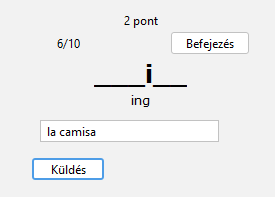
. ábra: egy másik példa, a kvíz időzített módja három megadott személlyel, valamint sötét megjelenéssel

Több szövegmező esetén a kitöltés sorrendje a felhasználó igénye alapján szabadon variálható. Alapértelmezés szerint az enter gomb megnyomására a program a következő szövegmezőbe mozgatja a kurzort. A feladatsor elküldéséhez az utolsó szövegmezőből kell kétszer entert nyomni, ilyenkor a kvíz át is ugrik a következő feladatsorra.

*Normál* módban az utolsó lap befejezése jelöli a kvíz végét. Ilyenkor elég a megszokott módon elküldeni a kvízt, majd az alkalmazás felülete automatikusan átirányításra kerül a kiértékelő táblázatokhoz. *Időzített* mód esetén ez a bal felső sarokban olvasható idő lejártakor következik be.

*Megjegyzés: ha az idő lejártáig nem sikerül teljesen befejezni az aktuális feladatsort, a program nem veszi figyelembe a kitöltés nélkül maradt szövegmezőket, illetve az esetlegesen félbemaradt kitöltés miatt csak a helyes válaszokat adja hozzá a végeredményhez. Ez viszont csak az időzített mód utolsó feladott lapján fordul elő!*

## Szófordítás kvíz

A másik típushoz képest ez a kvíz mindenképp egy egyszerűbb felülettel bír. A felső panel teljesen megegyezik az igeragozáséval. A bal oldal *normál* módban itt a feladatsorok számai helyett az elvégzett/összes szót tartalmazza.

. ábra: szófordítás kvíz nehéz fokozaton

Ha a nehézségi fokozat nem a *nincs segítség*re lett állítva, középen vastagon szedve olvasható pár betű a megoldásból. Közvetlen alatta található a magyarra fordított jelentés, vagy jelentések sorozata vesszővel elválasztva.

Mivel itt sosem fordulhat elő több szövegmező egyszerre, ezért az enter gomb leütése konfigurációtól függetlenül a következő szóra lépést (tehát a *Küldés* gombra kattintás helyettesítését) jelenti.

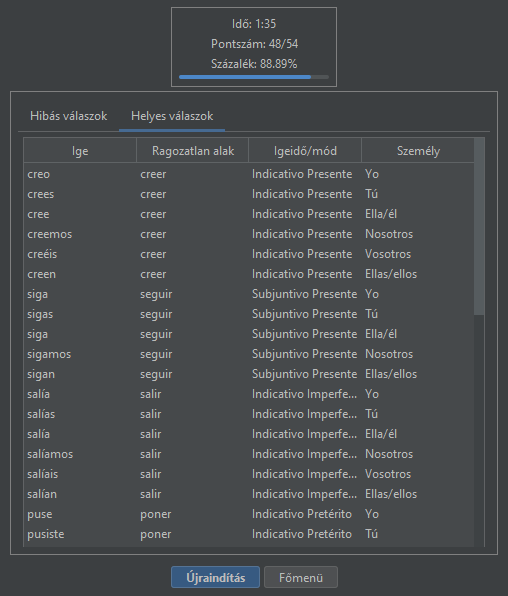
*Fontos: amennyiben a preferenciák között a névelők kötelező kiírása meg lett jelölve, azt a segítségként kiírt szó(töredék) feliraton nem kerül kijelzésre!*

## Kitöltési konvenciók

Mindkét kvízjáték során a szövegmezők azonos egyezmények alapján működnek. Tekintettel a magyar nyelvű felületre, használat közben elvárt, hogy a billentyűzet kiosztása magyar legyen. Ennek köszönhetően a spanyolban használt összes ékezetes magánhangzó begépelhető, így a helyesírási szabályok alapján a hangsúlyt jelző magánhangzók helyére ékezetes betűt kell írni, hogy a program helyes kiértékelést adjon (pl. *canción* – elfogadott; *cancion* – hibás).

Ebből kiindulva a nyelv közismert speciális karakterének használata is minden szükséges esetben elvárt. Tekintettel arra, hogy ez a karakter ritkábban jelenik meg a magyar billentyűzetkiosztások terén, **a program az alapértelmezett beállításai alapján az *ny* betűpárt a begépelése után azonnal** (a spanyolban ugyanazt a hangot jelölő) ***ñ*-re cseréli.**

## Kvíz kiértékelő panel

Egy kvíz befejezését követően a kiértékelő panelen rögtön megtekinthető az elért eredmény, valamint listába felsorolva az összes hiba és azok helyes formája *(****Hiba! A hivatkozási forrás nem található.****)*. Egy másik alpanelen az elfogadott válaszok is felsorolásra kerülnek. Az erről az oldalról való továbblépés esetén nincs lehetőség visszatérni, viszont a fontosabb itt felsorolt információk mentésre kerülnek. A felső sorban látható pontszám, a szavak száma (vagy az időtartam) és a százalékos eredmény mind visszanézhető a főmenüből elérhető *Eredmények* menüpontban. A hibák is ugyanoda kerülnek, pontosabban az átnézésre váró listába, valamint a következő kvízjátékok során gyakrabban visszaköszönnek majd, lehetőséget adva a felhasználónak a helyes változatuk gyakorlásához.

. ábra: példa egy szófordítás kvíz kiértékelő oldaláról

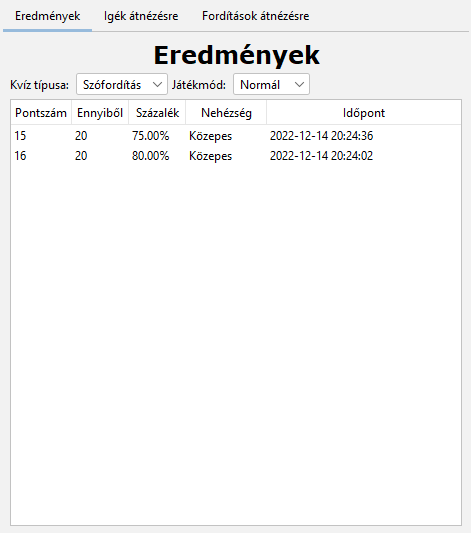
. ábra: egy igeragozás kvíz statisztikái és a helyes válaszok listája

A listák kvíztípustól függően más oszlopokat tartalmaznak (A főnevek a nő-, illetve hímnem mellett a magyar jelentést mutatják, míg az igék esetében a konjugált, illetve ragozatlan alak mellett az igeidő és a személyes névmás is kiírásra kerül). Az oszlopok sorrendje drag and drop módszerrel szabadon variálható az átláthatóság érdekében.

Az *Újraindítás* gombra kattintva az előzőleg lejátszott kvízt próbálhatja meg újra kitölteni a játékos, ugyanazokkal a preferenciákkal. Bár a szavak sorrendje megváltozik az előzőéhez képest, a kvíz továbbra is csak a korábbi kvízben felbukkanó szavakat fogja újra megjeleníteni. Több elkövetett hiba kijavítása miatt nem feltétlen szükséges az újraindítást választani, tekintve, hogy a program a kvízek befejezése után figyeli és tárolja az elkövetett hibákat. Az akár helyes válaszként megadott szavakkal kapcsolatos általános bizonytalanságok miatt ajánlott ez az opció választása, amiért ugyanazt a szavakból álló listát használja fel még egyszer a program.

## Eredmények

### Pontszám táblázat

A korábban befejezett kvízek kiértékeléseinek megtekintésére a főmenüben található *Eredmények* menüpont kiválasztásával van lehetőség. Itt a legelső panelen található listába olvassa be az alkalmazás az elmentett eredményeket. A két legördülő menü segítségével lehet kiválasztani, hogy pontosan melyik típusú kvíz melyik játékmódjában kitöltött eredmények lekérdezésére van-e igény. A lista alapból az igeragozás kvíz *normál* módját mutatja, de bármelyik legördülő menüben történő változás esetén frissül a lekérdezés alapján.

. ábra: a szófordítás kvíz normál módjának eredményei

A kiválasztott eredmények típusától függően más oszlopok jelennek meg. Az oszlopok szélessége itt is átméretezhető, és a sorrendjük is módosítható. A listák alapból a legutóbbi játékok időpontjának sorrendjében töltik be az elemeket, de az oszlopok feliratára kattintva ez sorrend megváltoztatható (pl. a *Pontszám* oszlopra kattintva a lista a legnagyobb/legkisebb elért pontszám sorrendjébe rendezi az elemeket).

*Fontos: a lista maximum 100 elemet tud megjeleníteni. A 101. elemtől kezdve a korábbi mentett eredmények így elvesznek.*

Lehetőség van az összes tárolt eredmény törlésére is, ez a *Beállítások* menüponton belül választható ki (29. oldal).

### Átnézésre váró igék és fordítások

A képen szöveg látható

Automatikusan generált leírásA pontszámok és egyéb statisztikák mellett a program eltárolja azon igék és fordítani való szavak listáját is, amelyeket a felhasználó a korábban kitöltött kvízei során hibásan küldött be. A főmenüben az *Eredmények* felületre navigálva található meg a két idetartozó panel.

16. ábra: példa egy átnézésre váró igecsoportra

A megfelelő panelre kattintás után a közvetlenül a felirat alatt található legördülő menüben választható ki a szótár, vagy szócsoport, amelynek az átnézésre váró tartalmát a felhasználó meg kívánja tekinteni. A szótár kiválasztása után a lista automatikusan frissül.

*Megjegyzés: az átnézésre váró listák nem tartalmazzák a korábbi kvízek során még nem megjelent szavakat. Ahhoz, hogy a listában egy szó megjelenjen, legalább egy hibát kell véteni a megfelelő kvízjáték során.*

Az alkalmazásban megjelenő többi listához hasonlóan, az átnézésre váró táblák oszlopai is szabadon variálhatóak sorrendileg. Előfordulhat, hogy néhány felirat (leginkább a *Definíció(k)* táblában) túl hosszú ahhoz, hogy alapértelmezetten elférjen a táblában. Ilyenkor az oszlop feliratának egyik oldalán található függőleges vonalat megragadva átméretezhető a teljes oszlop szélessége *(16. ábra)*.

A képen szöveg látható

Automatikusan generált leírásAhhoz, hogy egy, a listában megjelenő ige vagy fordítás eltűnjön, az **elkövetett hibák száma \* 3** alkalommal kell helyesen megadni a további kvízek során megjelenő ige konjugáltjait, vagy a szó fordítását. Amennyiben a megoldások begyakorlása sikeres, ez nem feltétlenül igényel túl sok időt. Az alkalmazás a minél többet elhibázott szavakra fókuszál a legtöbbet, azaz ezeket helyezi fontossági sorrendben a többi elé, majd jeleníti meg az újabb kvízek során. Például egy szófordítás kvíz esetén az első hiba után három alkalommal kell játékot indítani ugyanazon szótár kiválasztásával, amelyik a hibás szót tartalmazza. Ha a játékos szeretné minél előbb eltüntetni egy csoport listájából az összes átnézésre váró szót, megfelelő mennyiségű gyakorlás után elég elindítania egy kvízt az összes átnézésre váró szó számának megadásával a megfelelő csoportra.

17. ábra: egy több kvíz lejátszása után preferált lista

Amint az összes korábban elkövetett hiba elegendő alkalommal javításra kerül, a lista visszakerül abba az állapotába, amelyben a legelső kvíz indítása előtt is található volt *(17. ábra)*.

### Bélyegek

A kvízkitöltésekkel történő hosszútávú gyakorlás során lehetőség van egy-egy elért teljesítmény jutalmául szolgáló bélyegek gyűjtésére. A program első verziójában kizárólag az igeragozás kvízen belül van lehetőség az összesen maximum 15 darab bélyeg megszerzésére.

Ehhez nem kell mást tenni, mint minél több igét megfelelően konjugálni. A medálok egy-egy szótárban teljesített összes sikeres konjugálás százalékáért kerülnek kiosztásra.

18. ábra: három példa a bélyegekre. Az első a Conejito szótár felének teljesítése után, a második az Intermedio 80%-a után, míg a harmadik az Experto szótár összes szavának sikeres ragozásai után jár

A képen szöveg látható

Automatikusan generált leírásA program minden egyes kvíz végén ellenőrzi, hogy teljesült-e bármelyik bélyeg megszerzéséhez szükséges feltétel. Amennyiben igen, erről egy párbeszédpanelben is értesíti a játékost *(19. ábra)*.

19. ábra: új bélyeg szerzésekor megjelenő párbeszédpanel

Ahogy az előző ábrán is olvasható, a bélyegek az *Eredmények* menüben érhetőek el. Itt a felső sorban a balról a negyedik panelre kattintva *(20. ábra)* lehet megtekinteni az eddig begyűjtött bélyegeket (amennyiben már begyűjtésre került egy).

20. ábra: egy példa a bélyegek panelre, az első két szótárban található összes ige ragozása után

A begyűjtött bélyegek az összes, *Eredmények* oldalon található panel további tartalmaival egyetemben az alkalmazás kilépése után is megmaradnak.

## Beállítások

Az program és a kvízjátékok felületének, valamint egyéb összetevőinek testreszabására a főmenüből elérhető *Beállítások* oldalon lehet szert tenni. Itt találhatóak meg továbbá a felhasználó által mentésre került különböző adatok (eredmények, átnézésre váró listák, bélyegek) visszaállításához használható gombok is. Az itt meghozott módosítások mentés után a kvíz preferenciákhoz hasonlóan egy konfigurációs fájlban tárolva, az alkalmazás bezárása után is érvényben maradnak.

### Az alkalmazás beállításai

Egy kiemelt (bár talán kissé szubjektív) fontossággal bíró funkciója az újabb alkalmazásoknak a világosról sötét megjelenési módra váltás. Ez a lehetőség természetesen a ConjúgaMe beállításai között is megtalálható. A bal oldali szekció legfelső soraiban a megfelelő rádiógomb kijelölésével, majd az oldal alján található *Mentés* ikonra kattintás után az alkalmazás megjelenése meg is változik a kiválasztott üzemmódra *(21. ábra)*.

21. ábra: a beállítások oldal a sötét módra váltás mentése után

A következő, *Offline mód* feliratú jelölőnégyzet kijelölésével a program kikapcsolja az interneten keresztül történő kommunikációt. Ez egyben azzal jár, hogy nem fognak érkezni értesítések az adatbázist érintő frissítésekről, valamint további új információkról sem érkezik értesítés, amíg az offline mód bekapcsolva marad. Célszerű viszont kijelölni, ha a futtatási környezetben nem lehetséges internetkapcsolatot létrehozni, így az alkalmazás indításakor nem ugrik fel a csatlakozási hibából eredő, offline mód bekapcsolását kérő párbeszédablak.

Az alaphelyzetbe állítás részleg gombjaira kattintva teljesen törölhetőek az eddig elért eredmények, valamint játékos teljesítményéről szóló adatok. Az *Eredmények* gombra kattintva a **pontszámokat és egyéb statisztikákat tartalmazó táblázat mellett az eddig elért medálok is törlésre kerülnek**! A hibás igék és hibás fordítások gombok alaphelyzetbe állítják az összes szótárban található igének a felbukkanási esélyeit, valamint kiürítik az átnézésre váró igék és fordítások listák tartalmait is.

*Fontos: mindegyik gomb megnyomása után egy párbeszédablak figyelmeztet, hogy a visszaállítással teljesen elvész a korábbi adatok visszanyerésének lehetősége. Ennek tudomásul vételeként szolgál a párbeszédablak „igen” opciója, amely a mentés gombra kattintás nélkül, azonnal törli a kiválasztott adatokat!*

### A kvízjáték beállításai

22. ábra ny cseréje ñ-re opció működésben

A spanyol szavak bevitelével kapcsolatos legáltalánosabb felmerülő probléma az *ñ* karakter megtalálása az ASCII-kódok között (amely nemrégiben gyakorlatilag lehetetlenné is vált a magyar billentyűzeteken), vagy beillesztése más kontextusból. Ennek leegyszerűsítése érdekében, ha a felhasználónak nincs gyors opciója a karakter beszúrásához, hasznos lehet az *ny cseréje ñ-re* funkció aktiválása. Bekapcsolt állapotban az alkalmazás figyeli az összes szövegmezőt, ahol a felhasználó beütheti az *ny* karaktereket, és azonnal átfordítja azokat a spanyol alakra, ahogy ezt a 22. ábra is bemutatja.

A soron következő beállítás a kvízzel kapcsolatos oszlopon belül az *azonnali visszajelzés* mód. Ilyenkor mindkét típusú kvíz során láthatóvá válik a játékos részeredménye a felső sorban, illetve a feladatsor elküldése után egy zöld pipa, vagy egy piros X jelzi a szövegmezők mellett a megoldások sikerességét. A szavak gyakorlása esetén érdemes bekapcsolva hagyni a funkciót, viszont kikapcsolt állapotban így egy vizsga szimulálására is használhatóvá válhatnak a kvízek.

Az utolsó kvízzel kapcsolatos testreszabási lehetőség a szövegmezők közötti váltás beállítása. Mivel a szófordítás során az alkalmazás egyszerre csak egy szövegmezőt jelenít meg, ez a funkció **kizárólag az igeragozás kvízekben érhető el**. A *Következő mező (tab)* rádiógombot választva a billentyűzetről a *tab* billentyű mellett az *Enter* lenyomásával is lehetségessé válik a következő szövegmezőre ugrás a több személyes névmással operáló igeragozások során. A felhasználói visszajelzések alapján ez néhány játékos számára kézhezállóbb lehet a gyorsabb haladás érdekében.

*Megjegyzés: ebben a módban az utolsó szövegmező befejeztével az első leütés után megerősítésképp még egy enter megnyomása szükséges a feladatsor elküldéséhez.*

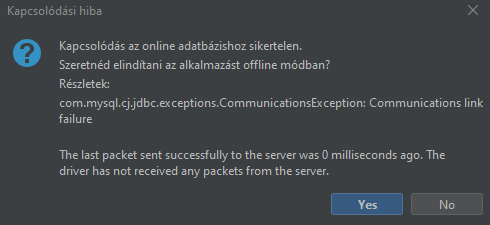
A hagyományos üzemmód kiválasztásához a *Küldés gomb (enter)* rádiót kell bejelölni. Ilyenkor kizárólag a tab-bal marad lehetséges az ugrás a sorok között, vagy a mindkét esetben elérhető átkattintással.

Amennyiben sikerült elvégezni az összes szükséges módosítást a felületen, azok érvénybe lépéséhez az alul található *Mentés* ikonra kell kattintani. A sikeres mentésről egy felirat is jelzést ad a felső sorban. A mentés nélküli *Vissza* gombra kattintással az összes végrehajtott módosítás elvész.

## Ismert hibák és kezelésük

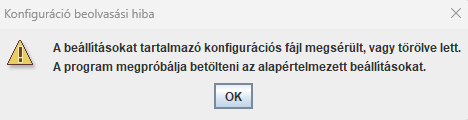
A program használata közben előfordulhatnak váratlan hibák, amelyek megakadályozzák azt az elvártak szerinti működésében. Természetesen bármilyen szokatlan viselkedés, vagy váratlan hiba fellépése esetén a program párbeszédpaneleken keresztül értesíti a felhasználót, és ha nem tudja tovább garantálni a futás hibátlan folytatását, kilép.

### Egyszerűbb hibák

Már korábban is említésre került, szituációtól függően is gyakran felbukkanhat az *Offline mód* aktiválását kérő ablak, amely a gyenge, vagy hiányzó internetkapcsolat esetén üti fel a fejét:

. ábra: sikeretlen csatlakozást jelző párbeszédpanel

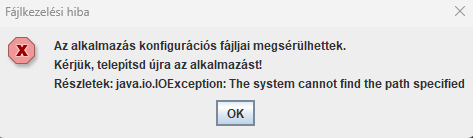
Az offline módban történő folytatáshoz ilyenkor elég a *Yes* gombra kattintani. A *No* gomb megnyomásával az alkalmazás nem indul el.

A tesztelés során a legtöbb problémát az károsult (*korrupt*), vagy hiányzó konfigurációs fájlok tudták előidézni. Az erről szóló hibaüzenet *(****Hiba! A hivatkozási forrás nem található.***22. ábra*)* leginkább a hordozható (*portable*) változat használata során bukkanhat fel, amiért annak komponensei kevésbé kötöttek. Bár ez a hiba önmagában még nem elég, hogy teljesen gátolja a programot a futásban, amennyiben a felhasználó gyakran találkozik vele, érdemessé válhat a váltás a telepíthető verzió használatára. Ilyen esetben valamelyik más program vagy az operációs rendszer váratlanul módosításokat végezhet a ConjúgaMe főkönyvtárán belül.

. ábra: egy hibaüzenet, amely a hiányzó, vagy károsult fájlok esetén lép fel

## Súlyos hibák

Előfordulhat viszont, hogy a konfigurációs fájlok túlságosan károsulnak, amelyet egy szinttől kezdve a program önállóan már nem tud kezelni (például, ha egy teljes almappa eltűnik). Ilyenkor a következő párbeszédpanel ugrik fel pár másodperccel a 23. ábrán bemutatott után:



. ábra: a súlyos hibaüzenet, amely az alkalmazás teljes működésképtelenségét jelenti

A képen szöveg látható

Automatikusan generált leírásTekintettel arra, hogy mindkét beszerezhető verzió *lightweight*, azaz nagyon kevés tárhelyet foglal el, a leggyorsabb módja ezen hibaüzenet kiváltásának az, ami olvasható is benne: a program újratelepítése, vagy a hordozható verzió újbóli letöltése (és a jelenlegi törlése).

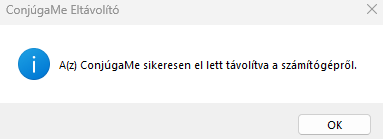
Ugyanez a szituáció áll fenn az adatbázis sérülése esetén:

. ábra: hiányzó vagy károsult adatbázisról értesítő súlyos hiba

## A program eltávolítása

Amennyiben a felhasználónak nincs szüksége a ConjúgaMe használatára, vagy szükségessé vált az újratelepítése, eltávolíthatja azt az összes komponensével együtt a számítógépről. Ehhez Windows operációs rendszer használata esetén, amennyiben telepítéskor a Start menü parancsikonok elhelyezése engedélyezve volt, elég a Start menüben a ConjugaMe alkalmazást megkeresni. Jobb kattintással kiválasztható az *Eltávolítás* (*Uninstall*) opció, majd a megjelenő ablakban egy újabb kiválasztással a listából el is indul az *uninstaller*.

A másik lehetőség az *uninstall.exe* fájl megkeresése a program telepítési könyvtárán belül. Alapértelmezetten ez a könyvtár a *C:\Program Files\* elérési útvonalon található meg.

Az eltávolító alkalmazás először megerősítést kér az eltávolításról, majd rögtön el is kezdi a program és komponenseinek törlését. Pár másodperc elteltével az eltávolítás sikerességét az alábbi ablak jelzi:

. ábra: az eltávolítás befejezésekor megjelenő ablak

Újratelepítés igénye esetén ezt követően a telepítő el is indítható.

# Fejlesztői dokumentáció

## Az implementáció megtervezése

A grafikus felhasználói felület kialakításához a ConjúgaMe a *Java Swing* toolkit elemeit használja fel, így a JVM-en fut az egyik lehető legszéleskörűbb offline platformfüggetlenség elérésének érdekében. A szoftver architektúrája az *MVC (Model-View-Controller)* programtervezési minta alapján épült fel. Ennek ellenére előfordulnak eltérések az eredeti alapvetésektől, például a boilerplate kódok és túlzott egymásra hivatkozások elkerülésének érdekében egyes nézetbeli osztályok tartalmaznak eseményvezérlőt.

A képen szöveg látható

Automatikusan generált leírásAz alkalmazás kódja a *Gradle* build automation toolt használja a dependenciák kezeléséhez, a tesztelésekhez, illetve a deployment folyamatának egyszerűsítéséhez. A verziókezeléshez a *Github* felülete vált a program hasznára az első billentyűleütéstől kezdve (28. ábra).

28. ábra: commit snippetek a Githubról

A képen szöveg látható

Automatikusan generált leírás A szótárak és a felhasználóhoz köthető adatok eltárolásáról a konfigurációs fájlok mellett egy SQLite adatbázis gondoskodik, amely lokálisan érkezik a kész programmal. Az adatbázis csatlakozását a java osztályokkal a *Gradle* által kezelt, *Maven Central Repository*ból származó *SQLite JDBC driver* garantálja.

29. ábra: néhány template sql fájl és a program használt adatbázisa

A lekérdezések és az update-k az adatbázisból külön-külön sql template-ként vannak tárolva (29. ábra), valamint egy alapszintű Java I/O-handler által kerülnek meghívásra és lekezelésre a kódban.

## Dependenciák

A képen szöveg látható

Automatikusan generált leírásAz előbb említett *SQLite connector* mellett a megjelenést a *FlatLaf Look and Feel* 2.4-es verziója szolgáltatja (a világos és a sötét módot is). A letisztult, illetve rendkívül hasznos constraint kezelése miatt a felhasználói felület felépítéséért a *MigLayout* 3.7.4-es verziója a felelős.

. ábra: a MigLayout felhasználása a kódban. Az itt olvasható összes string az elrendezéshez ad utasítást.

A teljes JVM-en futó tesztkörnyezetért a *JUnit 5*-on belül található *JUnit Platform* tehető felelőssé.

## Fejlesztői környezet

A képen szöveg, monitor, képernyőkép, képernyő látható

Automatikusan generált leírásA program egésze a *JetBrains IntelliJ IDE*-ben készült (31. ábra). A fejlesztői felület rendkívül hasznosnak bizonyult mind a valós idejű hiba és warning figyelmeztetésekért, a kódoptimalizálással kapcsolatos (vagy annak vélt) gyorstippekért és belejavításokért, valamint az integrált, gyorsan konfigurálható futtatási környezetért is. Ezenkívül a Gradle tool-ok támogatásának köszönhetően a függőségek beszerzésével, valamint a build script automatizálásával is rengeteg monoton pillanattól spórolt meg. A verziókezelő rendszerrel történő integráció miatt külső szoftver, vagy git parancsok használata nélkül is pár kattintással tudtam fetchelni, vagy pusholni a jelenlegi projektet.

. ábra: az IntelliJ integrált verziókezelő panele

## A konfigurációs fájlok

Az adatbázis mellett a program több adatforrást is (kvíz preferenciák, általános beállítások, jelvények stb.) „*.cfg*” konfigurációs fájlokból nyer ki (**Hiba! A hivatkozási forrás nem található.**). Az összes ilyen fájl a főkönyvtáron belül található *config* mappában helyezkedik el.

A legtöbb felépítése a könnyű beolvasás és kiírás érdekében nagyon egyszerű: soronként egy utasítást tartalmaznak, és egy-egy sortöréssel szekciókra vannak osztva (**Hiba! A hivatkozási forrás nem található.**). A szekció első elemeként található általában a vezérlő (leginkább változó, vagy arraylist) neve, majd a következő sor(ok)ban az értéke(ik). Arraylist esetén az utolsó elem utáni sorral a szekciót egy END szócska zárja le, egy érték megadása esetén erre nincs szükség.

32. ábra: az igeragozás kvíz preferenciáit tartalmazó konfigurációs fájl

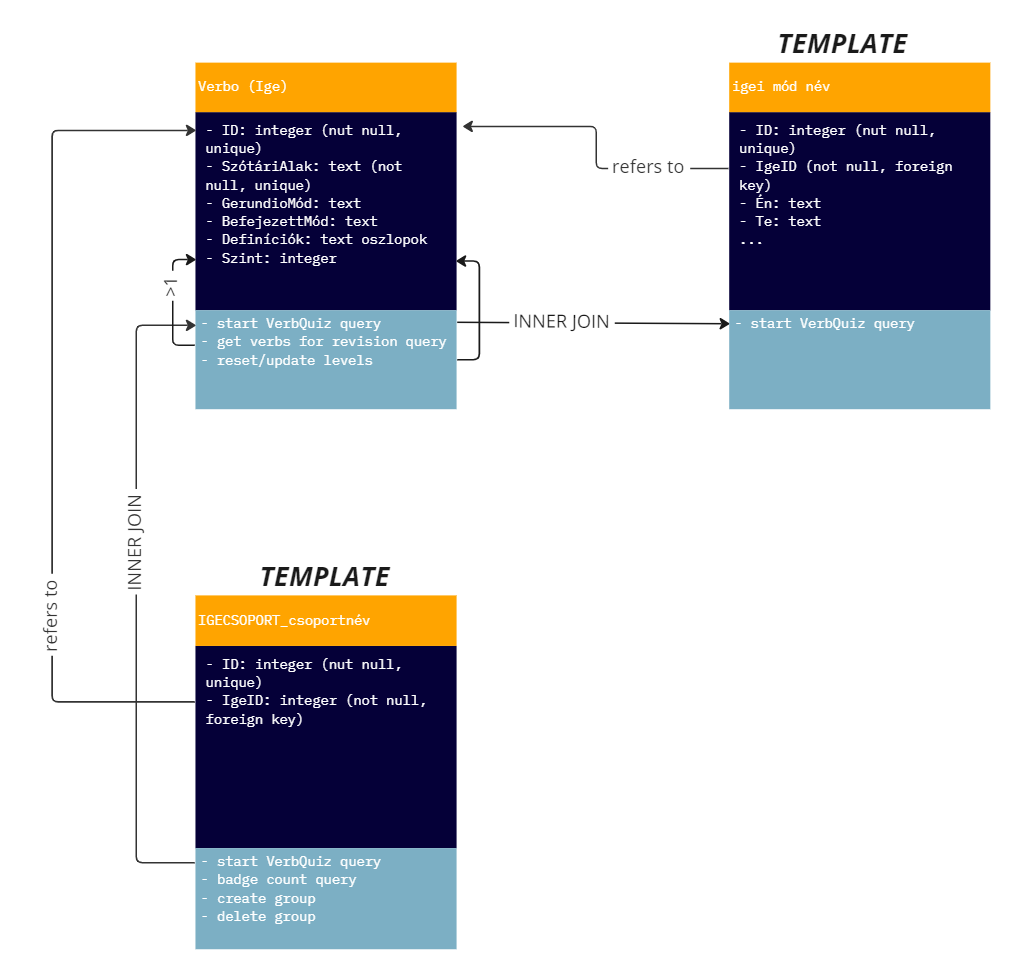
33. ábra: az igeragozás kvíz preferenciáit tartalmazó konfigurációs fájl

## Külső adatszerkezetek (lokális táblák)

A képen szöveg látható

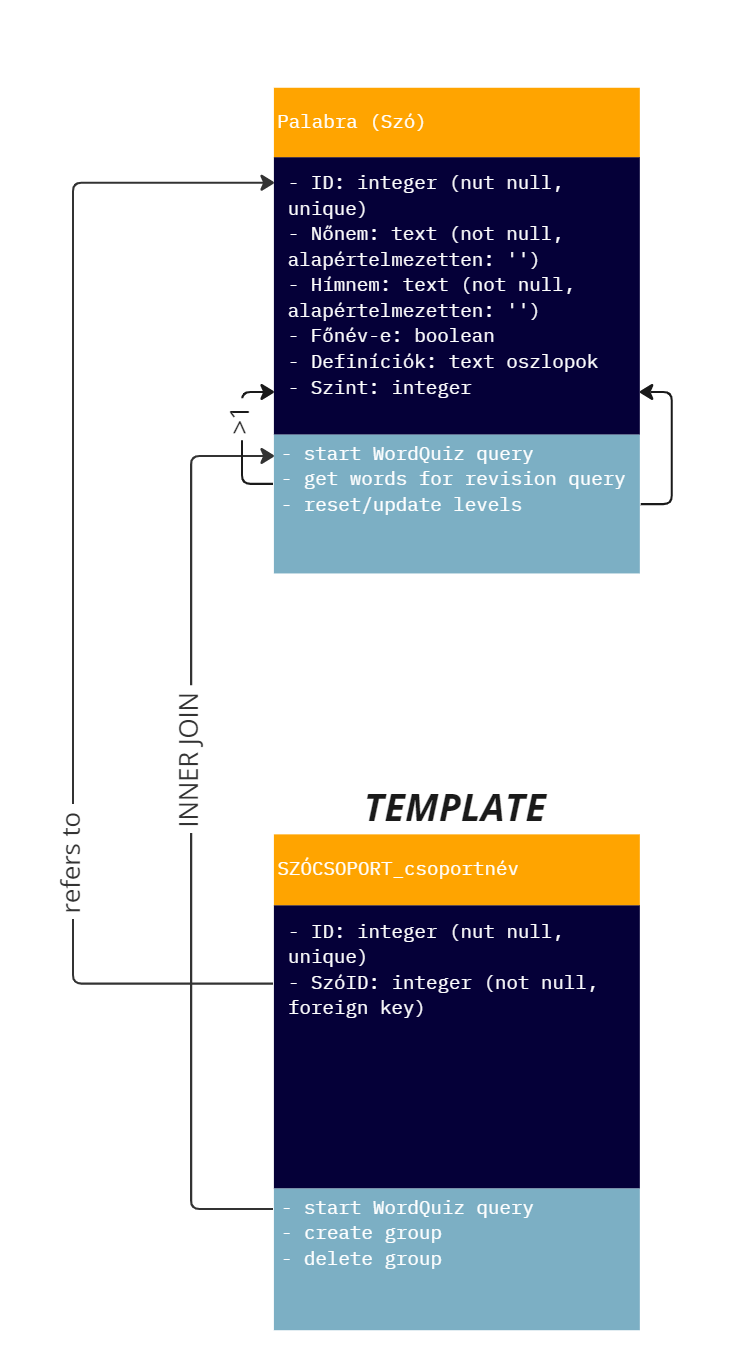
Automatikusan generált leírásAz SQLite egy egyszerű és kompakt adatbázismotor, így igyekeztem a benne foglalt táblákat és a köztük kialakuló kapcsolatrendszert is letisztultan alakítani.

### Igeragozás kvízhez kapcsolódó táblák

Az igeragozás kvízekhez a főtábla **Verbo** névre hallgat. Itt kerül tárolásra az összes felhasználható ige, valamint azok a két, személyes névmástól független *participio* alakjai, továbbá a definícióik és az elhibázások számosságát tartalmazó szintjük. Erre a táblára hivatkozik az összes **Igemód** tábla, azok *VerbID* nevű oszlopainak köszönhetően. Ezek a táblák tartalmazzák a megoldásokat a különböző igei módokban és időkben, egy-egy oszloponként minden egyes alanyestre. Az **Igecsoport** template alapján felépülő táblák az ID mellett egy, a szintén főtáblára hivatkozó *VerbID*oszloppal rendelkeznek.

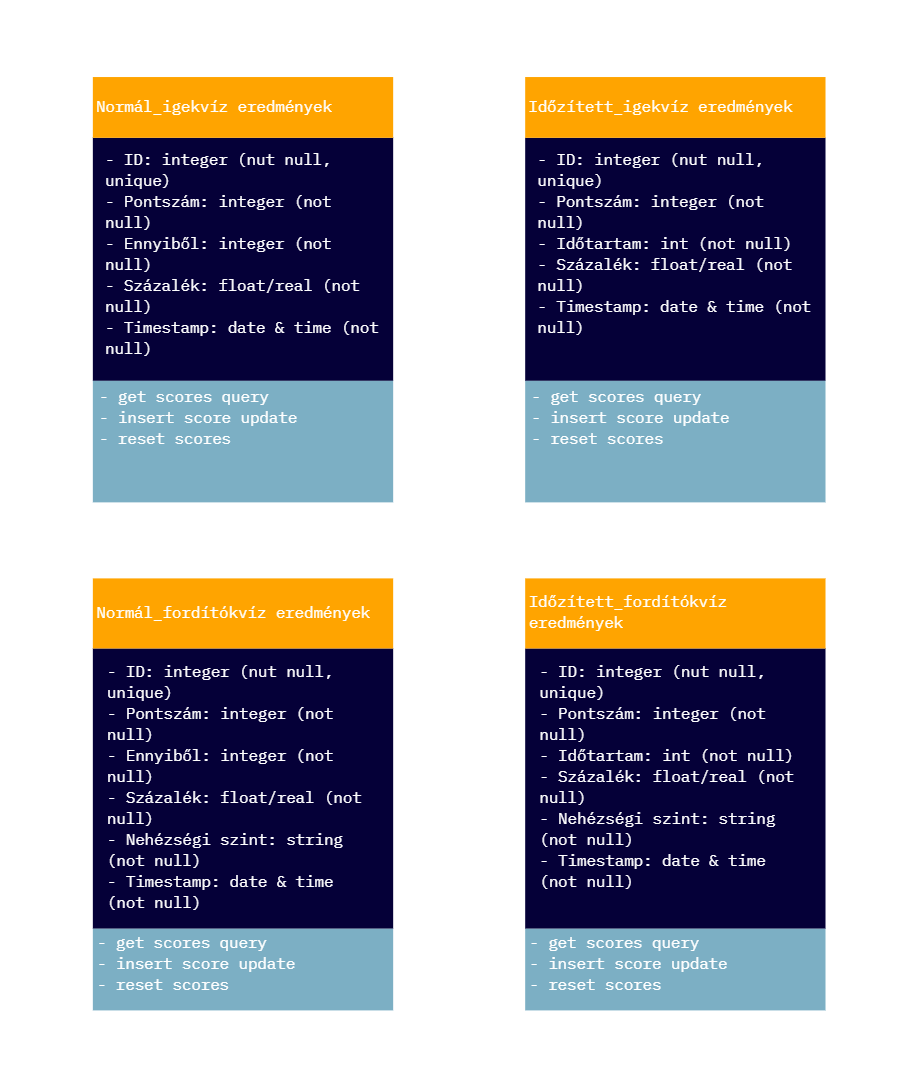
34. ábra: a program egyik fő adatforrásainak, az igéket tartalmazó táblák szerkezete

### Szófordítás kvízhez kapcsolódó táblák

A szófordítás kvíz esetében a struktúra a kevésbé komplikált összetevőknek köszönhetően egyszerűbb. A **Palabra** főtáblán található minden egyes szó lehetséges nő- és hímneme, valamint a definíciói, és az Ige főtáblából átemelt hibaszintet jelző oszlop is (35. ábra). Ezenkívül szavanként egy boolean érték is tárolásra kerül, amelynek köszönhetően a program felismerheti, hogy a szó főnevet, vagy melléknevet jelöl (ez fontos a névelők kitételének szükségessége esetén). Az ige id-re hivatkozó szótárak mellett külön táblákban kerülnek tárolásra a főnevek számára elkészített szótárak. A másik főtáblára való hivatkozáson kívül a felépítésük ellenben teljesen megegyezik.

35. ábra: a főneveket tartalmazó táblák adatszerkezete

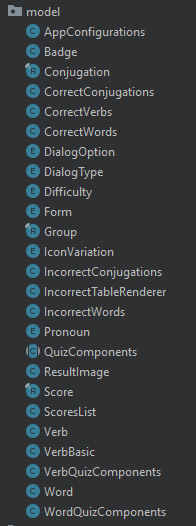
### Eredményeket eltároló táblák

Az adathalmaz teljessége, amelyet a program felhasznál a kvízek generálásához és ellenőrzéséhez így eltárolásra kerül az előzőek alapján. A felhasználó haladását szinten tartó hibaszint is a tárolt szavak mellé van integrálva. A kvíz befejezésekor kapott kiértékeléseket viszont egy teljesen új környezetben kell eltárolni. Ehhez 4 hasonló felépítésű tábla áll az alkalmazás rendelkezésére (36. ábra).

36. ábra: az eredményeket tároló négy tábla

Az időzített játékmódból kinyert eredmények esetén az időtartam egész számként kerül eltárolásra az alkalmazásban. Ennek az indoka a felhasználó általi sorba rendezés lehetőségének megőrzése. Az ezen táblákból nyert adatok kimutatásával foglalkozó osztály felelős ennek az egész számnak az olvasható visszafejtéséért.

## Belső adatszerkezetek (modellek)

A Java source fájlok között a már említett MVC-felépítés okán a nézettől és a vezérlőktől elkülönítve, saját osztályokban találhatóak meg a beolvasáshoz megfelelő struktúrát garantáló, vagy már előre megírt, nyers adatokat eltároló modellek.

37. ábra: az összes megtalálható modell az alkalmazásban

A felsorolásban látható (37. ábra) legtöbb osztálynak az elnevezése egyértelműen leírja annak tartalmát is, de közülük több is részletesebb magyarázatot igényelhet.

A személyes névmások (*Pronoun*), igemódok (*Form*) és nehézségi szintek (*Difficulty*) terén azért esett az *enum*ként való eltárolásra a választásom, mert a tartalmuk állandó és garantáltan változatlan marad. Egy-egy *enum* fájl egy modell összes lehetséges változatának megadása mellett tartalmaz egy stringgé alakítással és egy stringből enumra konverzióval foglalkozó függvényt is, hogy ne kelljen szükségtelenül további osztályokat létrehoznom az ezen modellek GUI-ban való kiírásához, vagy az onnan kapott potenciális enum értékek értelmezéséhez.

38. ábra: példa egy enum fájlra (személyes névmások tárolása)

Bár a *rekordok* egy boilerplate-től mentes és hatékony módjai a lekérdezések által nyert adatok tárolásának, sajnos a főbb adatforrásait a programnak (igék és fordítások) nem tudtam hatékony rekordokat felhasználó szerkezetben implementálni. Az igék összes konjugáltjának eltárolásához HashMap használatára volt szükség, amelynek bevezetését, mint attribútum, (a vele foglalkozó függvények miatt) átláthatóbbnak találtam egy osztályon belül. Helyettük viszont a felhasználó által generált összes adatot rekordokként tárolja a programom. Így például a négy eredménytábla egyikébe feltöltésre kerülő, vagy a négy közül bármelyikből lekérdezhető adat mindegyike egy rekord-osztályt használj csak fel.

Az átnézésre váró igék adatai is rekordokban tárolódnak (*Conjugation*), tekintettel arra, hogy már elég kevesebb információt tartalmazniuk, mint az eredeti osztály, ahonnan származnak. A JTable-ben való megjelenésükhöz viszont így is szükség volt egy-egy kiegészítő osztályra a modellek között (*Correct-Incorrect Conjugations/Words*), hogy az oszlopneveket és a rekordok listáját egy-egy függvény meghívásával a táblába lehessen helyezni.

### Ige és szó kvíz komponensek

39. ábra: QuizComponents osztály

Ezek a modellek felelősek a kvíz preferenciák eltárolásáért, valamint szolgáltatásáért a kvízjátékokhoz. Mindkét (*VerbQuizComponents és WordQuizComponents*) komponenseket tartalmazó osztály a *QuizComponents* absztrakt osztályból öröklődik, amely az alábbi adattagokkal rendelkezik, valamint ezek *getter* és *setter* metódusaival (amelyeket az egyértelmű felhasználási módjuk miatt a további osztályok részletezésekor sem sorolok fel):

* *isVerb (boolean):* megadja, hogy a komponens osztály egy igekvíz preferenciáit tartalmazza-e.
* *isNormal (boolean)*: *true* esetén hagyományos játékmódhoz használható preferenciákat tárol, *false* esetén időzítetthez.
* *wordAmount (int):* hagyományos játékmód esetén az összes feladatsor (azaz a megoldásra váró ragozatlan igék, vagy fordítások) száma.
* *durationMin (int):* időzített játékmód esetén a játék hossza percben.
* *durationSec (int):* időzített játékmód esetén a játék hossza másodpercben.
* *selectedGroups (Group ArrayList):* a feladatsorok generálásához felhasználható szótárak rekordjait tárolja el.

Két publikus, statikus adattag található még az absztrakt osztályban, ezek a decimális formátum (pl. az kvíz eredmény %-a), illetve a timestamp-ek elkészítéséhez használatos formátum template-jeit jelölik.

A *QuizComponents* ezenkívül még két felülírható metódust tartalmaz, az *isWorkingCorrectly*, amely argumentumok szüksége nélkül egy boolean értéket ad vissza az alapján, hogy az osztály adattagjainak aktuális beállításai alapján az alkalmazás képes lenne-e generálni egy kvízjátékot, illetve a *printStats*. Ez utóbbi csak a fejlesztési folyamatban vált hasznomra, gyors visszajelzésként kiírta a konzolra a meghívás időpontjában az aktuális komponenseket tartalmazó osztály adattagjainak az állapotát.

A képen szöveg látható

Automatikusan generált leírásA két, alfejezet elején említett leszármazott ebből az osztályból kevés új adattaggal bír. A *WordQuizComponents* a testreszabási felületéről már felismerhető *Difficulty* enumból vár csak egy kiválasztott értéket, illetve az *articlesNeeded* boolean tárolja el, hogy a megfelelő névelő kitétele a főnevek elé szükséges lesz-e a kvízjáték alatt.

40. ábra: a VerbQuizComponents két gyermeke

A *VerbQuizComponents* egy igekvíz összeállításához az absztrakt osztály adattagjain kívül két *ArrayList*-et tartalmaznak még:

* A *selectedPronouns* a *Pronoun* enumban felsoroltak közül, a kvízben felhasználandó személyes névmásokat tárolja.
* A *selectedForms* pedig ugyanígy, a *Form* enummal mint típusaként a feladatsorok generálásához választható igeidőket és -módokat.

A vizuális komponensek megjelenítésének, valamint egyéb tárolási és lekérdezési algoritmusok egyszerűsítésének érdekében található még két további boolean az osztály adattagjai között. A *participioPasadoSelected,* illetve a *participioPresenteSelected* a két, a szoftverfejlesztés korai szakaszaiban kellő galibát okozó speciális formák megkövetelését tartja számon. Ezen két igemód elkülönítésére a többi, *Form* enumban felsorolttól az alanytól függetlenül is megoldható mivoltuk miatt volt szükség.

Szintén ebben az osztályban megjelenik egy darab új metódus is. Az *integer*rel visszatérő *getTotalNumberOfVerbs* függvényt az igeragozás kvíz egy már említett különlegessége követeli meg: a hagyományos módban megadható és felhasznált *wordAmount* nem feltétlenül egyezik meg a ténylegesen megoldásra váró igeragozások számával (kizárólag akkor van megegyezés, ha a felhasználó csak egy igemódot és alanyt választott ki a listából). Az összes beérkezendő input számára viszont nagy szükség van többek között a kvíz befejezésekor történő kiértékelés során, ilyen esetben ez a függvény kerül meghívásra a pontos szám megszerzéséhez.

### A képen szöveg látható Automatikusan generált leírásIgék (Verb és VerbBasic osztályok)

41. ábra: az asszociációs kapcsolatban lévő Verb és VerbBasic osztályok

A *Verb* modell elsődleges felhasználási forrása az igeragozás kvízben megjelenő igék és azokhoz szükséges formák eltárolása. Fontos megemlíteni, hogy ennek, valamint a *Word* osztálynak nincsen közvetlen kapcsolata egy, a saját kvíztípusának preferenciáit tartalmazó *QuizComponent* osztállyal sem. *ArrayList*-ek típusaként kerül példányosításra mind a kettő, és a létrejöttük, illetve a feltölteni kívánt adattagjaik számosságát természetesen már befolyásolják az előbb említett osztályból származó példányok. Az ezzel kapcsolatos algoritmus viszont a *Controller* osztályok ismertetései között kerül bemutatásra.

Ez a modell kevés, de annál összetettebb attribútumokból álló készlettel rendelkezik. Elsőként rögtön egy másik modell osztály típusát használja fel: a *VerbBasic* az igék alapvető tulajdonságait leíró adattagjainak, valamint alanytól független formáinak erőteljes elkülönítése miatt jött létre. Az attribútumai felsorolva:

* *ID (int):* az ige a lokális adatbázis *Verbo* táblájában szereplő ID-ja
* *infinitivo (String):* az ige ragozatlan alakja
* *presento (String)*: az ige *Gerundio*, azaz *Participio Presente* alakja
* *pasado (String)*: az ige *Participio Pasado* alakja
* *definitions (String ArrayList)*: az ige összes magyarra fordított jelentését tartalmazó (maximum 3 elemből álló) lista

A 40. oldalon található *igekvízhez kapcsolódó táblák*at ábrázoló képen (34. ábra) is leolvasható, hogy több (három darab) oszlop is felel az igék magyar jelentéseinek kellő mennyiségű társításához. Míg a lekérdezések feldolgozása közben könnyebb ennek a három oszlopnak a tartalmait egyenként feltölteni az *ArrayList*-be, kiíráskor szükségessé vált egy függvény, amely összefűzve megjelenít mindent ebből az adattagból. Ebben segít a *getDefinitions* metódus, amely a *StringBuilder*-t felhasználva egy sorba, vesszőkkel elválasztva elhelyezi az összes tárolt definícióját az igének. Az eredményt pedig String típusú függvény révén visszaadja.

A *Verb* osztály első (*basic*) elemének tisztázása után az osztály az ige összes lehetséges konjugáltjait a *forms* nevű HashMap tagban tárolja. Ez a 3D tároló először is magát az igemódot (a *Form* enum választható elemeinek egyikét) használja fel kulcsnak, és ahhoz rendel egy másik HashMap-et értéknek. A második szintjén a kulcs a *Pronoun* enumból szerezhető valamelyik személyes névmás, az érték pedig a két eddigi kulcsot kombinálva kapható ragozott alakja az igének, stringként visszaadva. Tekintettel, hogy egy ilyen, teljesen kész HashMap elkészítése könnyen körülményessé válhat, az osztály konstruktora nem vár el egy kész 3D tárolót az inicializáláshoz. Az *appendVerbForm* metódusával egyenként hozzáadhatóak a konjugáltak, a megfelelő forma után egy 2D tárolóban a személyes névmások és az értékek megadásával.

### Szavak/fordítások (Word osztály)

Túllendülve a modellek nehezén, a *Word* objektumok már akár barátságosnak is nevezhetőek (42. ábra). A modell elnevezése azonban támadható, amiért gyakorlatilag az igék is szavaknak minősülnek, a modelljük mégsem ebből az osztályból származtatható. Ez a modell elsődlegesen természetesen a szófordítás kvízben A képen szöveg látható

Automatikusan generált leírásmegtalálható fordításra váró szavakat (főneveket és mellékneveket) tartalmazzák azok magyar jelentéseivel. Az igék tárolásához hasonlóan itt is van egy adattag az adatbázisból beszerezhető *ID-*hoz (ez egyébként a kiértékeléskor történő sql parancsok végrehajtásánál kerül felhasználásra. Ezek a parancsok az elhibázott/kijavított szavak szintjeit frissítik). Ezenkívül csak a két lehetséges megoldás Stringje szerepel még itt nőnemben (*femenino*), illetve hímnemben (*masculino*). Egy boolean változó (*isNoun*) felel még a főnevek és melléknevek elkülönítéséről.

42. ábra: a Word osztály

## Az alkalmazás kerete és a statikus osztályok

### MainWindow

Ez az osztály a szoftver kiindulópontja. A *JFrame*-ből származtatott *MainWindow* inicializálja a Java Swing könyvtár legszükségesebb elemeit, ezáltal gyakorlatilag innen indul ki a felhasználó felület felépülése.

A program futtatásához szükséges *main* függvény is itt található, amely csupán csak inicializálja az osztályt, amelyben tartózkodik. Az adattagok kizárólag statikus osztályokból állnak, amelyek később inicializálásra kerülnek a konstruktorban:

* *config (AppConfigurations):* ebbe a modellbe töltődnek be, illetve kerülnek futás idejű tárolásra a szoftver környezetével kapcsolatos általános adatok. Ez magában foglalja a verziószámot, a lokális adatbázis verzióját és a *beálítások* panelen elvégezhető összes preferenciának mivoltát.
* *verbComps (VerbQuizComponents), wordQuiz (WordQuizComponents):* a felhasználó által legutoljára mentett igeragozás, illetve szófordítás kvíz preferenciáit tartalmazzák. Korábbi mentés hiányában, vagy annak károsulásának esetén egy *default* példány töltődik be, amelyhez az objektum validált működéséhez szükséges eszközök a *ConfigIO* osztályban vannak definiálva (lásd következő alfejezet).
* *local (Local Database):* a *Database* absztrakt osztályból leszármaztatott *Local* osztály, ezt meghívva csatlakozik a főkönyvtáron *database* mappáján belül elhelyezkedő adatbázishoz.
* *dialog (Dialog):* az összes párbeszédpanel instance létrehozásáért felelős osztály.

Kiemelten fontos összetevő az itt található *switchPanels* függvény. Ez a felelős azért, hogy a paraméterként kapott korábbi panelt (*prev*) törölje a *Content pane*-ről, majd kicserélje azt a másik paramétér panelre (*next*), illetve elvégezze a GUI frissítését a JSwing rendkívül fontos és hasznos *repaint()* és *revalidate()* metódusainak felhasználásával. Sajnos, amiért ezen metódusoknak és a *Content pane*-nek is állandóan szüksége van egy közvetlen mutatóra a *JFrame* felé, ahonnan származnak. Így csakis a *SwitchPanels* miatt megkövetelt a *MainWindow* valamilyen formában történő átadása az összes példányosított *view* osztálynak.

### Input/Output config

A *ConfigIO* vezérlőosztályban írtam meg az összes beolvasással és kiírással kapcsolatos függvényt. Tekintettel az osztály statikus mivoltára, nem tartalmaz adattagokat, sem privát metódusokat.

A benne található függvények közül néhány kellően absztrakt ahhoz, hogy több, egymástól teljesen eltérő események bekövetkeztekor is visszatérjen az alkalmazás a meghívására. Ilyenek például a *readLines*, vagy a *readSQL* metódusok. Az előbbi a paraméterként Stringként átadott útvonal alapján megnyitja a kívánt fájlt (*UTF-8 kódolásban*), majd a *BufferedReader* segítségével beolvassa az összes sorát, és eltárolja azokat egy String ArrayList elemeiként. A beolvasás befejezésekor visszaadja a teljes tömblistát. A *readSQL* szintén egy fájlt olvas be, viszont a korábbihoz képest a szigorúbb megkötése, hogy az mindenképp egy SQL utasítást kell, hogy tartalmazzon. Ezt a teljes SQL kódot egy Stringként adja át az őt meghívó változónak. Ezáltal vállal ez a függvény egy fontos felelősséget az összes programon belüli query és update elkészítésében, a beolvasott template fájl tartalmának átadásával (lásd az ezt követő *Database* osztályt).

Leginkább ezen a területen, de többször is felhasznált függvények még a *default* komponens-osztály gyártok: *getDefaultVerbComps* és *getDefaultWordComps*. Mindkettő törzse példányosít egy-egy *QuizComponent*-et, majd feltölti őket úgy, hogy kellően kezdetleges/általános, mégis azonnal verifikálható változatúak legyenek. Így akár testreszabás nélkül is felhasználhatóak egy-egy kvíz generálásához.

Az osztály további metódusai nagyjából egy mintára épülnek fel: a *read/write Verb/Word Components* szavakból összefűzött függvénynevek sok sort tartalmaznak, de mindannyiuk egyértelmű utasításokat teljesítenek; be kell olvasniuk, vagy épp ki kell írniuk a (*MainWindow* osztályban felsorolt) globálisan felhasznált statikus változók egyikét. Ugyanezen sémára épül fel az alkalmazás konfigurációkat (*AppConfigurations*) kezelő I/O függvények törzse, és hasonlít rá a bélyegeket tartalmazó fájllal foglalkozó *updateBadgeFile* is.

### Database absztrakt osztály

Az előző alfejezetben részletezett, gyakorlatilag az összes fájlkezelést magára vállalt osztályhoz hasonlóan az adatbázissal kapcsolatos kommunikációt is teljesen elkülönítve és egy helyen összegezve képzeltem el lebonyolítani. Ezt sikerült is megvalósítanom a *Database*-ben.

A rengeteg String konstans a fejlécében a program alkönyvtáraiban megtalálható sql templatefájlok elérhetési útjait sorolja fel. Ezt követően az első, és inicializálás szempontjából legfontosabb függvénye az osztálynak a *connect*. A törzsében található egy közvetlen hivatkozás a JDBC dependenciára, majd azt követően az adatbázishoz való csatlakozáshoz szükséges konfigurációs változó létrehozása következik be. Ezzel a változó megkísérli a kapcsolódást. Sikertelen eredmény esetén a dobott kivételt ugyanitt lekezeli a program. Amennyiben sikeres a kapcsolódás, a *Database* osztály *connected* adattagját igazra módosítja.

Az alkalmazás rengeteg *query-*t indít (pl. *verbQuiz, wordQuiz megoldások*), az erre szakosodott függvények újból egy minta alapján épülnek fel. Egy *query*-vel két metódus foglalkozik. Az egyiket (*process* függvények) mindig a megfelelő vezérlő (*controller*) osztály hívja meg. Ez megkapja a *query*-hez szükséges összes adatot paraméterben, majd a *ConfigIO* osztályban található, az előző oldalon is említett *readSQL* függvényt felhasználva kicseréli a kapott sql parancs templateben a szükséges mezőket a releváns adatokkal. A kész parancsot legtöbbször, mint a saját visszatérési értékeként, átadja a másik (*build*) függvénynek. Itt történik meg az adatbázis elérhetősége esetén a *Statement* létrehozása, majd a sikeres lekérdezés esetén a *ResultSet*ből kinyerhető nyers adatok modellekbe helyezése és eltárolása. Amint ezzel végzett a függvény, a sikeresen felépített modelleket tartalmazó (többnyire ArrayList) tárolót az elvárt típussal visszaadja a meghívottnak.

Az eredmények és hibás szavak tárolásához, módosításához és törléséhez az *insert, update,* vagy *delete* sql commandokra van szükség. Az ezeket kezelő függvények a lekérdezéseknél részletezettekhez igyekeznek hasonlítani felépítésükben. Lényeges különbség itt, hogy a statement sikeres végrehajtásán kívül nincs szükség további beérkező információ feldolgozására és átadására a vezérlők felé. Így az itt felhasznált *process* függvények visszatérés nélküliek, és az összes *process* függvényen belüli template fájlból – az annak az adott preferenciák alapján történő módosulása után – kinyert parancsot ugyanaz a meghívott metódus küldi el az adatbázis felé, az *executeUpdateQuery*. Ez a lekérdezéseknél használt *build* metódusokhoz hasonlóan először a csatlakozást ellenőrzi, majd utána hozza létre és hajtja végre a szükséges *statement*et.

## Vezérlők és nézetek

Az alkalmazás keretét, annak az elindulásához kapcsolódó elengedhetetlen tényezőket, illetve a fontosabb, magyarázatot igénylő modelleket ismertetve ideje azoknak a használatba kerülését is bemutatnom. Tekintettel arra, hogy a nézetek többé-kevésbé csak a vezérlő által kezelt adattagok és metódusok végpontjait igyekeznek vizuálisan a felhasználó elé tárni, a lényegre törőbb magyarázat szempontjából a vezérlőket a velük társuló nézetek osztályaival együtt sorolom fel, illetve részletezem.

A nézetek általános felépítési módszere a *view* package átpörgetése után gyorsan egyértelművé válhat: az itt található osztályok legtöbbje a JPanel-ből öröklődik. A konstruktoruk igyekszik inicializálni az összes lehetséges adattagot, és lehetőleg a legvégén meghívja a mindenhol azonos név konvenciót követő *initComponents* privát, visszatérés nélküli metódust. Ez a függvény felel a felhasználói felületen található összes vizuális elem tartalommal való feltöltéséért, illetve megjelenítéséért, vagy épp frissítéséért. Amennyiben egy panel bonyolultabb struktúrával rendelkezik, itt további, (al)paneleket visszaadó metódusok kerülnek meghívásra. Ezek a konvenciót követve hasonló elnevezéssel rendelkeznek, csak a pontos szerepükkel kiegészítve (pl. *initButtonPanel*). A kapcsolatot a kontrollerrel, mint attribútum veszik fel (*one-way data binding)*. Az oda-vissza hivatkozás szüksége esetén átadnak egy magukra mutató referenciát paraméterként (*1:1 relationship*).

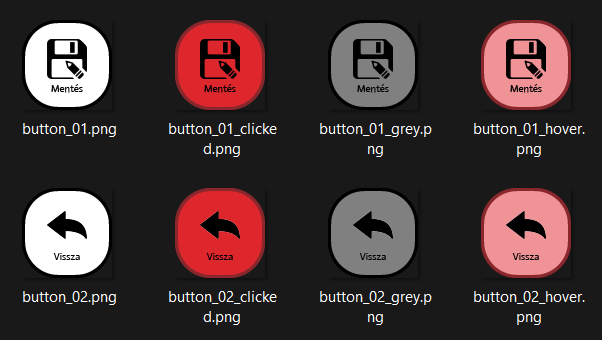
### Dashboard és MenuButton

A *Dashboard*, azaz a főmenü (4. ábra) struktúrája többnyire összeegyeztethető az előző oldalon leírtakkal, amelyben a *view* package-ben szereplő osztályok általános felépítését igyekeztem egységesíteni. A konstruktora inicializálja a panelt és az adattagjait, majd a külön *initComponents* metódusok megoldják, hogy a szükséges elemek megjelenjenek a képernyőn.

A view-controller kapcsolat viszont itt rögtön eltér a megszokottól. A kizárólag ezen a panelen meghozható változtatások és lehetséges paraméterek hiánya miatt, a *Dashboard* számára nem láttam szükségesnek egy definíció szerint operáló kontroller bevezetését. Ehelyett a *Dashboard* panel a *MenuButton* osztályt példányosítja, valahányszor egy gombot ad hozzá a képernyőhöz. Így egy kompozíciós kapcsolatot alakít ki az osztállyal, és a *MenuButton* példányok gazdaobjektumává válik.

A *Dashboard* attribútumai:

* *BUTTON\_NUMBER (konstans int)*: megadja a konstruktor törzsében elhelyezkedő, a gombok inicializálásért, majd később a panelra helyezéséért felelős ciklusok fordulásainak számát.
* *verbComps/wordComps (Verb/WordQuizComponents)*: referencia-rövidítésként funkcionálnak a MainWindow azonos nevű, statikus adattagjaira hivatkozva.
* *logo (JLabel)* és *logoIcon (ImageIcon)*: az alkalmazás logójának megjelenítéséért felelősek.
* *buttons (MenuButton ArrayList)*: a választható menüpontok gombjainak tárolója. A gombok képei mellett tartalmazzák az összes gomb saját eseménykezelőjét is.

A *MenuButton* a főkönyvtáron belül elhelyezkedő *img* mappa erős szerkezeti kötöttségei miatt sok különböző erőforrást használ fel viszonylag kevés információigénnyel. A *JLabel*-ből származtatott osztály példányosításához elegendő a jelenlegi panel nevének stringként történő átadása, és az inicializálni kívánt gombhoz rendelt szám megadása (43. ábra).

43. ábra: példa a gombok ikonjait tartalmazó fájlok rendszerezésére az img/buttons/settings alkönyvtárban

A paraméterként várt panel nevével megegyező nevű alkönyvtárban szereplő összes png fájl egy 1-től induló, inkrementáló számot tartalmaz közvetlenül kiterjesztése előtt. Ez a szám megfeleltethető egy gomb ID-nak, tekintve, hogy lehetővé teszi a JPanel osztályoikban a gombok legenerálását ciklusok segítségével. Ezzel együtt le is egyszerűsíti a gombok felhasználási folyamatát.

Miután a *MenuButton* hozzáadja a kapott paraméterekből felépített elérési út által beolvasott png fájlokat az *ImageIcon* típusú adattagjaihoz (*normalIcon, hoverIcon, clickedIcon, greyIcon*; utóbbi a sötét módban használatos), kicseréli a megjelenését az alapértelmezett (*normal*, vagy *grey*) változatra. Ezt követően hozzáadja saját magához a *mouseEntered, mouseExited, mousePressed* eseménykezelőket, hogy azok bekövetkeztekor a megfelelő *ImageIcon*-ra váltson (például, ha a kurzor belép a képernyőn lefoglalt területére, átvált a „*\_hover”* elnevezésű ikonra).

Végül a panel neve alapján kiválasztható metódusban egy-egy, a gomb ID-ját felhasználó switch-case elágazáson belül kiválasztásra kerül az elkészítendő, kattintáskor bekövetkező eseményfigyelő. Ugyanitt kerülnek lekezelésre a panelváltásból (amely az 52. oldal alján részletezett *MainWindow* osztály *switchPanels* metódusát felhasználva válik valóra) felmerülhető kivételek is.

### QuizSetup, QuizPreferences

Ezen párost tartom számon, mint az alkalmazás legösszetetteb részlegei. Olyannyira, hogy bár csak két osztály lett megnevezve az alcímben, több továbbit is együtt vonnak magukkal, ráadásul az említettek közül az egyik önmagában nem is létezik.

Az alkalmazás használata során kiszúrható, hogy a kvíz testreszabásáért felelős *VerbQuizSetup* és *WordQuizSetup* panelek bár külön füleken, de egy oldalon szerepelnek (ezt támasztja alá például az a jelenség is, hogy a mentés nélküli módosítások nem vésznek el, ha a felhasználó csak át-, majd később visszakattint ugyanazon fülre). Ezt a flületet a *SetupPane* panel garantálja, amely a nézettel foglalkozó osztályok szabványának teljes mértékben eleget tesz. Emelett a ténylege betöltött szerepe mindössze annyi, hogy egy *JTabbedPane*-be fűzi a két alá tartozó osztályt.

A *Verb-* és *WordQuizSetup* egy közös absztrakt osztályból történő származtatását - bár név és betölteni való funkciójuk alapján célravezetőnek tűnnének – nem találtam kellően összeegyeztethetőnek. Ellenben a vezérlőikkel, amelyek mindketten a *QuizPreferences*-ből öröklődnek. Ez leginkább annak köszönhető, hogy bár mindkét kvíznek van megegyező testreszabási lehetősége (szótárak, normál/időzített mód és az azokhoz tartozó számok), megjelenésben nem kötött az alpanelek elhelyezkedése a képernyőn. Ez onnan eredeztethető, hogy a szófordítás kvízhez sokkal kevesebb a módosítható preferencia, ezáltal a szükséges oszlopok száma is.

A képen szöveg látható

Automatikusan generált leírásA két *QuizSetup* és a hozzájuk kapcsolódó *QuizPreferences* társaik közötti reláció egy megszokott, teljesen hagyományos view-kontroller kapcsolatot idéz. A *Setup* panelek szolgáltatják a felhasználó számára a rádiógombokat/jelölőnégyzeteket és input mezőket, és ezek tárolásra kerülhetnek egy újonnan létrehozott, típusnak megfelelő *QuizComponents* példányban. Ennek a véghezviteléhez pedig egy-egy GUI-ban megjelenő gombhoz hozzárendelt eseménykezelőt kell csak aktiválni, amely meghívja a vezérlő egyik metódusát (*savePrefs*), ahol a komponenseket tartalmazó példány felépítése csupán egy részfeladat.

44. ábra: absztrakt bemutatása egy QuizComponents példány felépítésének kapcsolatáról a QuizSetup osztályok néhány közös elemeivel a QuizPreferences metódusain keresztül

A frissen elkészült *QuizComponents* példány nem kerül felhasználásra, amennyiben utána hibát talál benne a *validateForm* függvény. Ilyen esetben a kapott hibát továbbadva a *Setup* panelnek, a probléma egy *JLabel* feliratának módosításával kiírásra kerül a felhasználó felé. Ellenkező esetben pedig a verifikált komponensek eltárolódnak, és el is indulhat a kvíz.

### A kvízpanelek és vezérlőik

A képen szöveg látható

Automatikusan generált leírásElérkezett egy csúcspontja a fejlesztői dokumentációnak. Az alkalmazás legfőbb összetevői természetesen nem mások, mint a *Verb-*, illetve a *WordQuiz*, és az azokhoz társuló vezérlők. Az ezen panelek hibátlan felépítését garantáló összetevőkhöz szükséges ismeret javarésze már viszont részletezésre került.

45. ábra: legfontosabb adattagok és kapcsolódási pontok egy igeragozás kvíz lefutása során

Ahogy azt a 45. ábra igyekszik bemutatni az igeragozás kvíz példáján keresztül, a vezérlő (*VerbQuizController, WordQuizController*) többé-kevésbé magáénak tudja tulajdonítani a legtöbb fontos háttéradatot és -metódust, de akadnak kivételek. Időzített kvíz esetén például a nézetet irányító osztály kezeli az időzítő elindítását és leállítását.

Egy másik különlegessége a *view* package *Verb/WordQuiz* osztályainak, hogy a hagyományos view-controller kapcsolatuk mellett a *Dashboard – MenuButton*-hoz hasonló kompozíciós kapcsolatban állnak a *Section* osztályból származtatott példányokkal. Ahogy a *Dashboard* esetében, úgy itt is a nézetosztály tölti be a gazdaobjektum szerepét.

A *Section* és alosztályai különlegessége, hogy bár a vezérlők közé tartoznak, a *JPanel*ből vannak származtatva, akárcsak egy nézetosztály. Mégis azért találtam az elhelyezésüket találóbbnak a controllerek között, mert tartalmazzák az egy-egy feladatsor kiértékeléséhez szükséges megoldást is.

Mindkét kvíztípus *Section*je tartalmaz egy *JTextField*-et a felhasználó megoldási kísérleteként kapott bemenő adat megszerzéséhez (*input*). Az ebből kinyert szöveg összehasonlítása a *solution* adattaggal adja meg a kiértékelés eredményét, és ettől függően vált át a szövegmező mellé illesztett *checkLabel* *JLabel* ikonja a pipa, vagy az áthúzás gif-re.

Egy kvíz létrejöttekor először inicializálódik a megfelelő controller, és csak azt követően kezdődik meg a vizuális elemek megjelenítése. Az igekvíz *initComponents* metódusának lefutásakor a választott személyes névmások számának, illetve a hozzájuk adható *Gerundio* és *Participio Pasado* formuláktól függően jelenik meg valamennyi *VerbSection* a panelen. Szófordítás kvíz esetében összesen mindig csak egy *WordSection* van jelen.

A szoftver a kvíz kezdetekor és a lefutása során végbemenő utasításainak javarésze tisztázható a vezérlő adattagjainak és metódusainak ismertetésekor. A fontosabb attribútumok általánosítása:

* *quiz (VerbQuiz, WordQuiz)*: az *1:1* kapcsolat kialakításához itt kerül tárolásra a nézetosztály referenciája.
* *comps (VerbQuizComponents, WordQuizComponents)*: a következő adattag mellett a kvízekhez kapcsolódó legfontosabb információkat tartalmazó adatforrás.
* *verbs/words (Verb/Word arrayList)*: a kvíz felépülésének az elején, a controller konstruktorából meghívott *Database* osztály *processVerb/WordQueries* metódusból nyert, a megadott komponensek alapján megszűrt *Verb/Word* példányok tárolója, amely a feladatsorok legfontosabb elemeit szolgáltatja.
* *score (int):* 0 az értéke inicializáláskor, majd körönként frissül a *nextRound* függvényen belül. A kvízben szerzett aktuális pontszámot tárolja.
* *outOf (int)*: szintén 0 értékkel indul, viszont vétett hiba esetén is inkrementálódik az értéke, így a kvíz befejezésekor megadja a maximum elérhető pontszámot.
* *iteration (int)*: folyamatosan a jelenlegi kör sorszámát tartalmazva ez az adattag is nullával kezd, és egy-egy feladatsor kiértékelése után kerül növelésre az értéke.
* *time (int)*: mivel a játékos az időzített módban előbb is befejezheti a játékot, az *outOf*-hoz hasonlóan ez az adattag a játék során ténylegesen eltelt időt tartja számon és adja vissza, majd azt adja vissza annak végén.
* *currentVerb/currentWord (Verb/Word)*: a jelenlegi feladatsorban kiírt és megoldásra váró ige, vagy szó és annak összes tulajdonságát, valamint megoldását tartalmazó példánya.
* *currentForm (Form, csak igekvíz esetén)*: ha lett kiválasztva alanytól függő igeidő, a jelenlegi feladatsorban kiírt és a megoldásokat ez alapján elvárt formára alakított igemód.
* *mistakes (String ArrayList)*: a hibás felhasználói inputot eltároló lista, a kiértékeléshez szükséges.
* *correctConjugations/correctWords (Conjugation/Word)*: azok az igék/szavak, amelyekre a várt megoldás(oka)t a játékos hibátlanul adta meg.
* *incorrectConjugations/incorrectWords (Conjugation/Word)*: azok az igék/szavak, amelyekre a várt megoldás(oka)t a játékos hibásan adta meg.

A vezérlőkben használt fontosabb metódusok:

* *nextRound()*: kétféleképpen kerül meghívásra. A kvíz konstruktorában, illetve a *Küldés* *JButton* eseménykezelőjében jelenik meg, amely kattintáskor aktiválódik. Először ellenőrzi, hogy a meghívása előtt inkrementált *iteration* adattag elérte-e a komponenseket tartalmazó osztályból kinyerhető *wordAmount* értékét. Amennyiben igen, meghívja a *finishQuiz* metódust. Különben módosítja a *currentVerb/currentWord* attribútumot a *verbs/words* következő indexének értékére, illetve frissíti a GUI-t és a szövegmezőket tartalmazó *Section*-ök megoldásait. Igekvíz esetén van egy *isFirst* paramétere, ez csak a konstruktorból történő meghíváskor ad vissza igaz értéket. Az első meghíváskor történő speciális *Section*-ök felé történő paraméterátadások miatt van rá szükség.
* *evaluateSections(last: bool)*: kiértékeli minden egyes *Section-re* a kapott inputot. Helyes válasz esetén növeli a *score* értékét, és hozzáadja a *CorrectWords/Conjugations*-höz a jelenlegi igét, vagy szót. Helytelen válasz esetén ugyanezen elemet értelemszerűen az *IncorrectWords/Conjugation*-höz adja hozzá. Az *outOf* értékét mindkét esetben inkrementálja. Amennyiben a kapott paraméter igaz (csakis időzített kvíz esetén), nem értékel ki helytelen megoldásokat, hogy az idő lejárta miatt kihagyott szövegmezők ne okozzanak váratlan hibajelzéseket a kiértékeléskor.
* *finishQuiz()*: időzített kvíz esetén kiszámolja az eltelt időt, majd létrehozza az EndQuiz panelt, amelyre át is váltja a felhasználói felületet.

### EndQuiz és QuizResults

A kvíz befejezése után a vizuális kiértékelés előtt a *Verb/WordQuizResults* vezérlőosztályok konstruktoraiban történik meg a szükséges statisztikák kiszámítása. A kapott hibátlan és hibás szavak *ArrayList*-jeit feltölti az azokat könnyedén egy-egy *JTable*-be illeszthető verziójú osztályokba (*Correct/Incorrect Words/Conjugations*). Ezt követően módosítja az *outOf* változó értékét a valósra annak függvényében, hogy normális, vagy időzített módban volt-e játszva a kvíz, majd kiszámolja a *score / outOf* képlettel a kapott százalékot. Később ezen érték felhasználásával építi fel a nézetben az *EndVerb/WordQuiz* osztály az eredményt mutató csíkot (*JProgressBar*).

A vezérlő ezek után még további ellenőrzéseket és adatbázis frissítéseket is elvégez a metódusaival:

* *insertScore()*: amennyiben a játék 1, vagy annál több elért pontszámmal fejeződött be, új rekordot ad hozzá a kapott statisztikákkal és egy timestamp-el a megfelelő játéktípus és -mód eredményeit eltároló táblába. Ehhez felhasználja a *Database* osztály szükséges metódusát, a további hasonló függvényekkel együtt.
* *updateIncorrectLevels()*: frissíti a *Verbo/Palabra* tábla *Level* oszlopának értékeit, ha legalább egy hibás megoldás született. Minden hibásan megoldott szó jelenlegi szintjéhez hozzáad 3-at.
* *updateCorrectLevels()*: frissíti a *Verbo/Palabra* tábla *Level* oszlopának értékeit, ha legalább egy hibátlan megoldás született. Minden hibátlanul megoldott szó jelenlegi szintjéből levon 1-et, amennyiben ez az érték még nem 0.
* *checkForBadges()*: ellenőrzi, hogy a kvíz kitöltése során a játékos elért-e egy bármelyik, még meg nem szerzett bélyeghez szükséges szintet. Ehhez két *COUNT* lekérdezést indít, és annak eredményével meghívja a következő metódust.
* *validateBadge()*: kiszámolja a kapott lekérdezések eredményéből a lehetséges bélyegszerzési szinteket. Ha a játékos elérte az egyik szintet, a *ConfigIO* ideillő metódusát meghívva hozzáad egy új bélyeget az azt tároló fájlhoz (*config/badges.cfg*).

A sikeres update-ek és ellenőrzések után frissül a nézeti oldalon a két *JTable* a helytelenül/helyesen megadott válaszokkal. A panel a *SetupPane* mintájára két külön fülön tárolja el a két táblát. Üres tábla esetén egy default model töltődik be, amely értesíti a felhasználót, hogy nem szedett össze hibás/hibátlan megoldást. Továbbá a két listát tartalmazó *JTabbedPane* megfelelő füle elszürkül, és elvész a kiválasztásához szükséges engedély.

### A képen szöveg látható Automatikusan generált leírásEredmények és vezérlőik

. ábra: az Achievements osztály adattagjai

Az *Achievements* nézetosztály az előzőhöz hasonlóan a *JTabbedPane* füles elrendezését használja fel, hogy több panelt egyszerre tartalmazhasson. Itt jelennek meg az eredmények, illetve az átnézésre váró igék és fordítások, valamint az összegyűjtött bélyegek is. Míg az első három JTable-ben kerül előhívásra, addig a bélyegek soronként négyesével jelennek meg, amennyiben a konfigurációs fájl tartalmaz akár egyet is.

A *JTable-*k kitöltéséhez az *AchievementsController* megfelelő metódusai kerülnek meghívásra. Mindkét átnézésre váró fül tartalmaz egy *JComoBox*-ot, ahol kiválasztható a megtekinteni kívánt szótár. Valahányszor a felhasználó a *ComboBox* (pl. *verbRevisionComboBox*) aktuális elemét egy másikra módosítja, annak értéke (azaz a kiválasztott szótár neve) átadódik paraméterként a vezérlőhöz. Az ezt megkapó *updateVerb/WordRevisionList* függvények visszatérési értéke maga a *JTable*, amit az *Achievements* osztálynak így már elég csak megjelenítenie a felhasználói felületen. Ezek a függvények egy *getVerb/WordRevisionList* metódus meghívásával töltik fel a tábla összes elemét és oszlopnevét tartalmazó osztállyal példányosított ArrayListet. Ebben a metódusban található meg a *Database* osztályból indított lekérdezés, valamint az onnan szerzett feldolgozott adat továbbadása is.

Az *updateScoresList* függvény egyszerre két *ComboBox (quizTypeComboBox* és *quizModeComboBox*) legördülő menü közül bármelyik módosításakor meghívásra kerülhet, így folyamatosan mindkettő értékét várja paraméterként (*isVerb, isNormal*). E dupla információ alapján ugyanezen függvény meghívja a szükséges metódust a *Database* osztályból a lekérdezéshez, majd a másik kettőhöz hasonlóan, a kapott adatokból felépíti a visszaadandó *JTable*-t. Tekintettel arra, hogy ez a tábla az egyetlen, amely nem csak String-eket tartalmaz, módosítani kellett az alapértelmezett modelljének típusait oszloponként.

### Beállítások és preferenciáik

A *Settings* osztály két alpanelt tartalmaz, az alkalmazással, illetve a kvízjátokkal kapcsolatos beállításokat. Mindkettő saját *JPanel*-t alakít ki a felületen, de az ott található mezők értékei ugyanabba a konfigurációs fájlba tárolódnak el: a *config/appconfigurations.cfg*-be.

Az ide történő kiíráshoz a *SettingsPreferences* controller segít. A *QuizPreferences*-hez hasonló felépítésű vezérlő legfontosabb kapcsolódási metódusa a *savePrefs*, amely rögtön egy másik, saját függvény meghívásával, a *setupConfig-*gal létrehoz egy új *AppConfigurations* objektumot. Ebben tárolja el a *Mentés* gomb eseményfigyelőjének aktiválódása előtti állapotát az oldalon található preferenciáknak. Amennyiben ez sikeres, a *ConfigIO* osztály *writeAppConfigurations* függvényét felhasználva kiírja a módosításokat ez említett fájlba, illetve jelzi a felhasználónak a mentés sikerességét. Ellenkező esetben is ugyanazon módszerrel, a *Settings* osztályon belüli egyik metódus (*writeOutErrors)* meghívásával jelez a hiba bekövetkeztéről.

## JUnit tesztek

. ábra: a konzol az igeragozás és szófordítás kvíz komponenseinek validálásáról szóló tesztek elvégzése után

A program teszteléséhez a *JUnit 5* környezetet használtam fel. Az egységtesztek elkészítése során a vezérlőkre, valamint azok modellekkel és nézetekkel kialakuló kapcsolataira koncentráltam. A tesztesetek létrehozása közben igyekeztem a lehető legváratlanabb tartalmat megadni inputnak, főleg a verifikációt igénylő metódusok tesztelésére.A képen szöveg látható

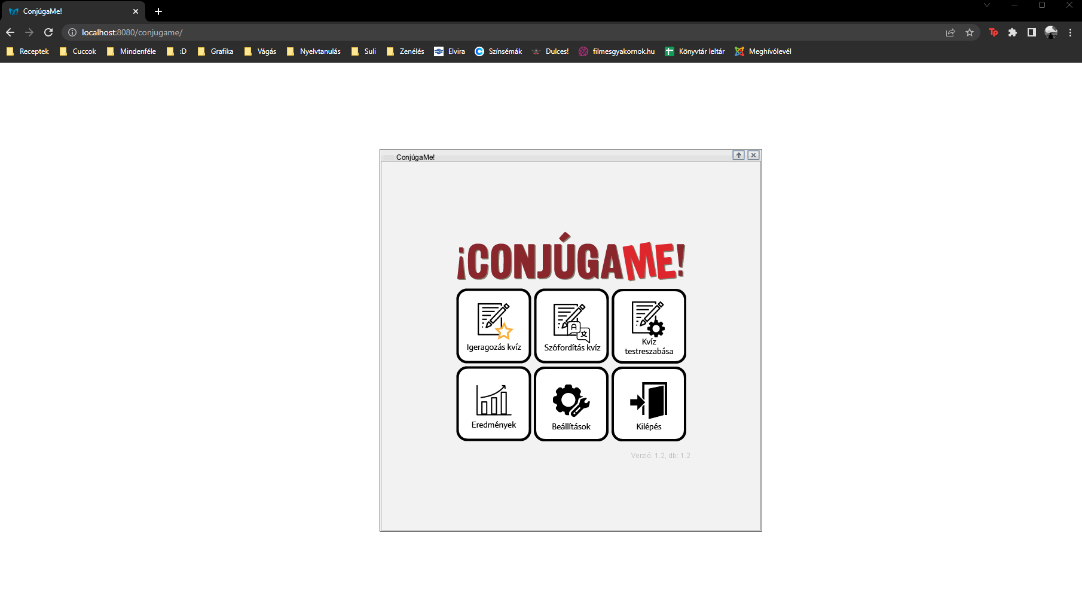
Automatikusan generált leírás

A tesztek megtalálhatóak a fejlesztői főkönyvtáron belül az *src/test/java* mappába navigálva. Az elindításukhoz az *IntelliJ IDE*-t használtam, ezzel felgyorsítva a tényleges tesztelési folyamatot. Az *IDE* megléte viszont nem feltétlenül szükséges, a tesztesetek a konzolból is elindíthatók az alábbi paranccsal:

*java -cp /path/to/junit.jar org.junit.runner.JUnitCore <test class name>*

## Böngészőbe integrált változat (WebSwing)

Az alkalmazás használható egy webböngészőben is. Az ebbe a környezetbe történő átültetéshez a *WebSwing* keretrendszert hívtam segítségül. A webes felület teszteléséhez szolgáló tömörített fájl elérhető a weboldalamon (*taughtmarci.web.elte.hu*) a telepítő és a portable asztali verziók mellett.

Kicsomagolás után az alkalmazás elindításához futtatni kell a főkönyvtárban található *webswing&admin.bat* fájlt. Ezzel pár másodperc múlva elindul egy lokális szerver alapértelmezetten a 8080-as porton. A szerver címét (*localhost:8080*) egy böngésző címsorába beütése után a felület egy felhasználónevet és jelszót kér a folytatáshoz, melyek alapértelmezetten az alábbiak: *„admin”* és *„pwd”*. Az ezt követően megjelenő listában már csak ki kell választani az egyetlen lehetőséget, a *ConjugaMe* alkalmazást, és ki is próbálható a játék böngészős felülete!

. ábra Chrome böngészőben futtatott ConjúgaMe

## Executable és telepítő

Az exe wrapper készítéséhez a *Launch4J* programot használtam fel. A build előtt a jar mellett hozzáadható volt a manifest fájl is, valamint pluszként megadtam a szükséges minimum JRE és JDK verziókat, így elavult verziók esetén a később fellépő hibák helyett azonnal figyelmeztet a frissítésről.

A képen szöveg látható

Automatikusan generált leírásA program telepítőjét és eltávolítóját egy *InnoSetup*-ban írható scripttel készítettem el. A scriptben kitértem wrapper mellett szükséges összes komponens include-ja után arra is, hogy azok jogosultságot adjanak a programnak a módosításukhoz.

49. ábra: az InnoSetup-ban elkészített script a telepítő generálásához

# Összefoglalás

Az elképzeléseim alapján a projekt teljesítése során eltelt időm körülbelül harmada a nyelvgyakorlással lett volna eltöltve. Ennél nagyobbat nem is tévedhettem volna, de utólag visszagondolva nem bánom, hogy a vártnál jóval mélyebben el kellett merülnöm újabb programozási technikák elsajátításában és felhasználásában a *ConjúgaMe* elkészítése során.

Azt hiszem, az elsődleges célokat sikerült elérnem a program jelenlegi implementációjával. Az eddigi tesztelők közül többen is folytatták a kvízjátékok kitöltését az általam kért feladatok elvégzése után is. Mivel az adatbázist saját kezűleg töltöttem fel adatokkal, azt hittem, hogy a tesztelésre már nem lesz semmi meglepő a feladatsorok között, de érezhetően szükségem van a kvízjátékokkal való gyakorlásra.

Úgy érzem, a szoftver megírásának kezdete óta a korábbi évekhez képest összehasonlíthatatlanul nagyobbat emelkedett a programkódok közötti komfortérzetem. Az elmúlt hónapok során a rengeteg weboldalt átböngészve kevés maradt a végére, amely a sokadik nekifutásra is teljesen érthetetlen maradt számomra (ahol pedig mégis volt, azoknál meg éreztem az égbekiáltó hiányosságot, és az visszarántott oda a tovább fejlődés érdekében). Bár sosem állt tőlem távol a Java használata, aggódtam, hogy az egyre inkább vita tárgyává váló objektumorientált nyelvek közül is az egyik koronázatlan királyt választottam ki az első komolyabb hangvételű ötletem megvalósításához. Visszanézve nem bántam meg a döntést, a platform-függetlenség és a programozást kényelmessé tevő dependenciák kellő bizalmat adtak, hogy később is újra előássam a Java-t egy-egy feladathoz, ha már egy kicsi relevanciája is lehet hozzájuk.

A programot illetően úgy gondolom, hogy rengeteg fejleszteni való része maradt, illetve minél többet dolgoztam rajta, annál inkább sikerült megálmodnom újabb feature-öket a felület és a nyelvgyakorlás gazdagabbá tételéhez. Ebből kiindulva még egy ideig biztos az életem szerves lesz az alkalmazás source kódja.

# További fejlesztési lehetőségek

Az egyik leggyakrabban felbukkanó hiánycikk az alkalmazásból a skálázhatóság volt a fejlesztés során. Ezt egy nagyobb tartalomfrissítés előtt mindképp szeretném bevezetni, de amiért a kód alapjait még teljesen más elvek alapján fektettem le, tartottam attól, hogy a refaktorálás túlságosan elnyújtaná a szakdolgozatom befejezési időpontját.

Egy másik iránya a továbbfejlesztésnek, amely gyakran jutott az eszembe, az alkalmazás online elérésének lehetővé tétele, az ezzel járó eszköztárral együtt (saját fiók, ahova mentésre kerülnek az eredmények, globális scoreboard, megosztható tartalmak).

A felhasználóbarát szó- és szótárhozzáadás szerepel még a tervek között, de ahhoz a jelenlegi adatbázis struktúra nincs megfelelően optimalizálva. Szeretném először csak a fejlesztőknek megadni a lehetőséget, hogy mire a felhasználói oldalról is kiegészíthetővé válnak a táblák, már bő mennyiségű adatforrás álljon a játékosok rendelkezésére.

Továbbá az igék modelljein belül a jelenleg túlságosan is általánosított formában implementált hiba-nyomkövetés helyett lehetne minden egyes, az alkalmazásban szereplő konjugáltnak egy saját hiba mérőszáma. Ez elősegítené az elkövetett hibák célzottabb kijavításának lehetőségét.

Az esztétika rovására felírható még a szófordítás kvízből nyert adatokat tartalmazó táblák kissé szétszórt megjelenése a két nem elkülönítése miatt. Ez lehetne egy újabb logika alapján egységesítve úgy, hogy közben a szavak neme továbbra is egyértelműen jelezve van.