Debreceni Egyetem

Informatikai Kar

SZAKDOLGOZAT

Web alapú lokalizációs alkalmazás megvalósítása

Készítette:

Molnár Károly

programozó matematikus

Témavezető:

Belső: **Dr. Papp Zoltán**

egyetemi adjunktus

Külső: **Diviánszky Ákos**

producer

Invictus-Games Kft.

Debrecen

2008

Tartalomjegyzék

I. Bevezetés

1. Témaválasztás

1.1. Szakdolgozatom témája

Ez a dolgozat egy konkrét webes alkalmazás fejlesztését írja le PHP és MySQL segítségével. Az alkalmazás célja, hogy mind a szoftverfejlesztőknek, mind a végfelhasználóknak legyen egy olyan közös, publikus felületük, amelynek segítségével a szoftverek honosítása egyszerűbbé válhat. Egy olyan hely, amely mindkét tábor számára hasznos lehet: a fejlesztők értékes munkaórákat spórolhatnak meg, míg a felhasználók nyomon követhetik a szoftverek honosítási állapotát - sőt maguk is szerkesztőkké válhatnak.
A dolgozatban bemutatott alkalmazás a honosítás folyamatát kívánja megkönnyíteni. A fejlesztők projekteket regisztrálhatnak be, és azok lokalizálandó állományait tehetik közzé. A fejlesztőkkel kapcsolatban álló lokalizálást végző cégek, a végfelhasználók, vagy bárki, aki az oldalra téved megnézheti és szerkesztheti ezen állományok különböző nyelvű változatait.
Az alkalmazás célja, hogy a közösség által szerkesztett és elfogadott honosítások készülhessenek.
A bemutatandó alkalmazás csak szöveg alapú tartalmak honosítását támogatja, ami egy későbbi továbbfejlesztésben kiegészítendő multimédiás tartalmak (audio, video) támogatásával.

1.2. A téma általános bemutatása

1.2.1. Globalizáció

„Az egész világra kiterjedő folyamat, amely a gazdaság, politika, a kultúra egységesülésének irányába hat.” [2]. A szoftverfejlesztők azt szeretnék, ha programjaik minden környezetben könnyen használhatóak lennének függetlenül attól, hogy milyen nyelvterületen használják őket. Ennek elérése érdekében támogatni kell más országok, nyelvek szokásait, nyelvtani szabályait.

1.2.2. Internacionalizáció és lokalizáció

A számítógépek elterjedésével és a szoftveripar növekedésével felmerült az igény arra, hogy az alkalmazások a felhasználó anyanyelvén kommunikáljanak a felhasználókkal.

Az internacionalizálás a szoftvertervezésnek egy része, amelynek következtében a szoftver különféle nyelvekre és régiókra adaptálható anélkül, hogy változásokat kellene alkalmazni az alkalmazás logikájában.

A lokalizáció (honosítás) olyan eljárás, amelynek során egy szoftvert adott nyelvre vagy régióra alakítanak át úgy, hogy a nyelvre jellemző komponenseket és lefordított szövegeket adnak hozzá.

1.2.3. Szabad munkaerő kiaknázása

Az internetet rengeteg ember használja szabadidejében különösebb cél nélkül. Az ilyen emberek idejét, tudását és vágyát, hogy valami hasznosat, maradandót alkossanak, számos internetes portál használja ki, többek között a YouTube és a Wikipedia is. Egy honosítást segítő alkalmazás is kihasználhatná ezt a fajta önműködő folyamatot.

1.2.4. Lokalizálási szempontok

- Nyelv
  - Számítógép által előállított szövegek
    - Karakterkódolás; a legújabb rendszerek Unicode-ot használnak a karakterkódolási problémák megoldása érdekében.
    - Különböző számrendszerek.
    - Szövegírás iránya; balról jobbra Európában, míg jobbról balra Perzsiában és az arab országokban.
    - Egy adott nyelven belül a szavak írásbeli eltérése.
      pl: *localization* (en-US), *localisation* (en-GB)
    - Szövegfeldolgozási különbségek.
  - Szövegek grafikai reprezentációja (nyomtatott anyagok, szöveget is tartalmazó bitképek)
  - Beszéd (hangok)
  - Film vagy videó feliratozása

- Kultúra
    - Képek és színek: érthetőség és kultúrális elfogadottság kérdése
    - Nevek és címek
    - Állam által engedélyezett számok (pl: személyi igazolvány szám) és útlevelek
    - Telefonszámok, címek és irányítószámok
    - Pénznem
    - Súlymértékrendszer (pl: metrikus és angolszász mértékegységek)
    - Közlekedési szabályok (pl: bal- és jobbkormányos autók)
- Írásbeli megkötések
    - Dátum/idő formátum, beleértve a különböző naptárak használatát
    - Időzónák (pl: UTC - nemzetközi környezetben)
    - Számok formázása (tizedes jelek, elválasztójelek helye)
- A termék vagy szolgáltatás bármely más olyan külső formája, amely a közmegegyezés tárgyát képzi

2. Alkalmazásfejlesztés webre

2.1. PHP

A PHP az egyik legnépszerűbb webes szkript nyelv. Az interneten megtalálható oldalak nagy százalékát PHP segítségével fejlesztették ki. Népszerűsége abban rejlik, hogy könnyen tanulható, hasonlít az ANSI C-hez, de nincsenek benne konvenciók, keverhető a megjelenítés az alkalmazás logikájával, segítségével könnyen generálható dinamikus tartalom, jó szövegfeldolgozó képességekkel bír és széles körű támogatást nyújt adatbázisok kezeléséhez. A PHP fő használati formája a szerver oldali programozás, szkriptek írása.

2.1.1. Objektumorientáltság

A PHP kifejlesztésekor nem rendelkezett objektumorientált vonásokkal.
A PHP3 vezette be az objektumorientáltság alapjait, amelyet a PHP4-ben továbbfejlesztettek. A PHP5 megjelenésével pedig már az objektumorientált programozás teljes eszközrendszere a programozók rendelkezésére áll, miközben máig megtartották a függvény-alapú és objektum orientált programozás, vagy ezek keveréke közötti választás lehetőségét.

2.1.2. A PHP5 új lehetőségei

Már a fejlesztés korai szakaszában nyilvánvalóvá vált, hogy a számos elérhető PHP verzió közül a legújabbat, a PHP5 -öt fogom használni, kiaknázva a benne rejlő új lehetőségeket:

Láthatósági módosítók

A PHP5 bevezeti a 3 megszokott láthatósági módosítót: `public`, `protected`, `private`. Az olyan attribútumok, illetve metódusok, amelyekhez explicit módon nem deklaráltunk módosítót, publikus láthatóságúak lesznek.

Öröklődés

Az objektumorientált programozás egyik legfontosabb jellemzője az öröklődés. A PHP nem támogatja a többszörös öröklődést. Az osztályok közötti szülő-gyermek kapcsolatot az

`extends` kulcsóval adhatjuk meg. Egy osztály egyetlen szülőosztályára a `parent` kulcsszóval hivatkozhatunk.

Konstruktor, destruktor

A konstruktor olyan metódus, amely automatikusan meghívódik, amikor példányosítunk egy objektumot. A PHP5-ben ezt kétféleképpen tehetjük meg: az osztályon belül definiáljuk a `__construct()` metódust vagy létrehozunk egy, az osztály nevével megegyező nevű metódust.

A destruktor olyan metódus, amely automatikusan meghívódik, amikor egy objektumot megsemmisítünk. Ez általában a PHP szkript befejezésekor történik meg. Explicit módon a `__destruct()` metódus deklarálásával hozhatjuk létre. Destruktort nem kötelező írnunk, mert a PHP tartalmaz GC-t (garbage collector; automatikus szemétgyűjtő mechanizmus).

Metódusok túlterhelése

Ahogyan azt az objektumorientált nyelvekben megszokhattuk, az alosztályban felüldefiniálhatjuk a szülőosztály metódusait. Ezt úgy tehetjük meg, hogy a túlterhelni kívánt metódus nevével azonos nevű metódust hozunk létre az alosztályban. Ha a metódus deklarációjában magdjuk a `final` kulcsszót, akkor a leszármazott osztályokban nem lehet azt túlterhelni.

Kivételkezelés

A PHP5 bevezette a kivételkezelést. Saját kivételeket deklarálhatunk az `Exception` ősosztályból. PHP-ben is `a try/catch` szerkezetet használhatjuk kivételkezeléshez. Az olyan kódrészt, amely hibát okozhat, a `try` blokkba kell tenni. Ha kiváltódik egy kivétel, akkor azt a `catch` ág kapja el. Minden `try` blokkhoz tartoznia kell legalább egy `catch` ágnak. Több `catch` ág esetén több fajta kivételt kezelhetünk le.

A `throw` kulcsszó segítségével dobhatunk kivételt.

Ha egy blokkban kivétel váltódik ki, akkor a PHP megkeresi az első megfelelő catch blokkot. Ha egyet sem talált, akkor hibát dob (Fatal Error).

Objektumok életciklusa

Minden objektum életcikusa addig tart, amíg az őt definiáló szkript futása be nem fejeződik. A PHP nem rendelkezik globális hatáskörrel, viszont vannak olyan eszközök, amelyek segítségével megtarthatjuk objektumainkat.

Hasznos függvények

`class_exists()`: segítségével megtudhatjuk, hogy egy adott osztály definiálva van-e.

`method_exists()`, `property_exists()`: segítségével megtudhatjuk, hogy egy adott metódus, illetve attribútum létezik-e egy osztályon belül.

`get_declared_classes()`: az aktuálisan elérhető osztályok listáját adja meg.

`get_class_methods()`, `get_class_vars()`: egy adott osztály metódusainak, illetve attribútumai-nak listáját adja meg.

`get_class()`: segítségével megtudhatjuk egy adott objektum osztályának nevét

get_parent_class: segítségével megtudhatjuk egy adott objektum vagy osztály szülő osztályát

`is_a()`: segítségével megtudhatjuk, hogy egy objektum leszármazottja vagy tagja-e egy osztálynak

`is_subclass_of()`: segítségével megtudhatjuk, hogy egy adott objektum egy megadott osztály egy alosztályához tartozik-e.

`serialize()`: segítségével bármely PHP értéket tárolhatunk és továbbíthatunk anélkül, hogy a szerkezetük és/vagy típusuk elveszne. A megadott értéket karaktersorozatba kódolja. Így bárhol tárolható. Ha vissza szeretnénk kapni az értéket, használhatjuk az unserialize() függvényt.

2.2. MySQL

A PHP alól leginkább használt adatbázisrendzser a MySQL, amely a legnépszerűbb nyílt forráskódú SQL adatbázis kezelő rendszer. A MySQL gyors, megbízható, gyorsan fejlődő és egyszerűen használható.

2.2.1. A MySQL5 új lehetőségei

A MySQL5 egyik nagy előnye, hogy immár lehetőség nyílik tárolt eljárások használatára is. Ennek lényege, hogy az adatkezeléshez kapcsolódó több helyen alkalmazott eljárásokat magában az adatbázisban helyezhetjük el. Ennek előnye, hogy egy helyen kell megírni és

nyilvántartani a kódot, de többen, több helyről is hozzáférhetnek. Tárolt eljárás használatakor gyorsabb lesz a végrehajtás is, mivel előre feldolgozva tárolja a műveleteket az adatbázis.

2.3. AJAX

Az AJAX (aszinkron JavaScript és XML) webfejlesztési technikák egy csoportja, amellyel interaktív webes alkalmazásokat hozhatunk létre. Ajax segítségével az alkalmazások kliensen futó részei a háttérben aszinkron kaphatnak adatokat a szervertől anélkül, hogy a betöltött lap megjelenítésébe és viselkedésébe beavatkozna.
A szerveren lévő adatok az `XMLHttpRequest` objektum használatával érhetők el.

2.3.1. Előnyei

- Sok esetben egy weblap számos olyan tartalommal rendelkezik, amelyet több lapon is meg kell jeleníteni. Hagyományos esetben az ilyen tartalmakat az oldal minden egyes újratöltésénél be kellene ismét olvasni. Azonban Ajax használatával az alkalmazás kérheti csak a változó tartalom frissítését, ezáltal nagyságrendekkel lerövidítve a betöltési időt.
- Az aszinkron kérések interktívabbá tehetik a webböngésző felhasználói felületét, gyorsabban lehet a felhasználó beviteleire reagálni, és az oldal különböző részei frissítődhetnek egymástól függetlenül. Az alkalmazás gyorsabbá, felhasználóbarátabbá válik.
- Használatával a szerverhez való kapcsolódások száma csökkenthető, hiszen a szkripteket és stíluslapokat elég egyszer lekérni.

2.3.2. Hátrányai

- A dinamikusan létrehozott oldalak nem regisztrálódnak a böngésző előzményei közé, így a böngésző előző oldalára való visszalépés nem az Ajax-ot használó oldal előző állapotára ugrik vissza, hanem az előtte látogatott utolsó oldalra.
- A dinamikusan előállított oldalakat a felhasználók nehezen tudják könyvjelzőként elmenteni.
- A neten lévő legtöbb keresőrobot nem futtatja le a szkripteket, ezért az alkalmazásnak támogatnia kell a tartalmak hagyományos úton való elérhetőségét.

- Azok a felhasználók, akik olyan böngészőt használnak, ami nem támogat JavaScript-et vagy Ajax-ot, vagy a JavaScript le van tiltva, nem fogják tudni használni a technológia előnyeit. Hasonlóan a mobil eszközök (telefonok, PDA-k) sem támogatják a JavaScript-et.
- Nincs jól bevált módszer arra, hogyan lehetne az Ajax alkalmazásokat hatékonyan tesztelni.

2.4. Védelem

A webes alkalmazások az Interneten keresztül bárki számára elérhetőek, ezért szembe kell nézniük annak a kockázatával, hogy illetéktelenek manipulálhatják őket.
Ezek a támadások nagyrészt a megfelelő érvényesítés és szűrés nélküli közvetlen felhasználói bevitelben elrejtett utasítások formájában jelennek meg.
A támadók akkor sikeresek, ha sikerül a program adott helyen lévő szemantikáját úgy megváltoztatniuk, hogy az eredeti helyett a saját utasításaikat hajtsa végre az alkalmazás.
Az eredmény minden esetben jogosulatlan információszerzés, adatmódosítás vagy akár adatvesztés is lehet.

2.4.1. Lehetséges támadási felületek

PHP támadások
A PHP támadások egy része a dinamikus változó kiértékelés sebezhetőségével hozhatók kapcsolatba. A PHP támogat "változó változókat", olyan változókat, amelyek más változók neveit értékelik ki.
Például, ha az URL-ben lévő paramétereket az alábbi módon dolgozzuk fel:

```
$safevar = "0";
foreach ($_GET as $key => $value)
  $$key = $value;
```

, akkor a támadó a honlap címében megadja a "`safevar=1`" paramétert, amivel sikerült megváltoztatnia egy létező, rejtett változó értékét.

A PHP támadások másik formája a fájl betöltések / feltöltésekben hagyott sebezhető részek kihasználása.

Például, ha a felhasználó döntésére bízzuk, hogy honlapunk melyik részén töltjük be:

```
<form>
   <select name="File2Load">
      <option value="news">hírek</option>
      <option value="options">beállítások</option>
   </select>
   <input type="submit">
</form>
```

, akkor a választást nagy valószínűséggel az alábbi PHP kóddal dolgozzuk fel:

```
$color = 'blue';
if (__isset($_GET['File2Load']))
  include($_GET['File2Load'] . '.php' );
```

Ez a módszer több támadási felületet is hagy. Egyrészt a HTML kódot bármikor átírhatja saját gépén a támadó, és megadhat más betöltendő fájlt vagy egyszerűen a lap címében (“`valami.php?File2Load=news`”) lévő `File2Load` paraméternek ad meg olyan fájlt, amely akár más szerveren található és a támadó által írt utasításokat tartalmazza.

SQL támadások

A webes alkalmazások többsége megbíznak a relációs adatbázisrendszer (Relational Database Management System, RDBMS) alapú szerverekben, amelyek mutatnak némi sebezhetőséget SQL támadásokra Ennek a sebezhetőségnek az alapja abban rejlik, hogy az SQL utasítások karakterláncokból épülnek fel. Lehetséges támadási pontok lehetnek például az űrlapok szövegdobozai vagy URL-ek.

Például felhasználó bejelentkeztetésekor az SQL lekérdezés így nézhet ki:

```
SELECT Username
FROM Users
WHERE Username = 'Username' AND Password = 'Password'
```

Tegyük fel, hogy a validálás az alapján történik, hogy a lekérdezés adott-e vissza sort vagy sem. Ha a támadó egy létező felhasználói nevet ad meg, de a jelszót nem ismeri, képes bejelentkezni azzal, hogy az űrlap jelszó mezőjébe az alábbi szöveget írja be:

```
"password' OR '1'='1".
```

Ekkor a lekérdezés a következőképpen néz ki:

```
SELECT Username
FROM Users
WHERE Username = 'Username' AND Password = 'password' OR '1'='1'
```

, ami minden esetben visszatér legalább egy sorral, így a program beengedi.

2.4.2. Mire kell figyelnünk?

A legtöbb probléma annak a hibás feltevésnek a következménye, hogy a bemenő adatok helyesek. Ezt a jóhiszemű feltételezést használják ki a támadók. Ahhoz, hogy programunk a triviális támadások ellen védve legyen, a következőket kell a fejlesztés során szem előtt tartanunk:

- Szűrjük a felhasználói adatbevitelt
  - reguláris kifejezéssel csak a megfelelő karaktereket engedjük meg:

    [^0-9a-zA-z]
  - amennyiben a felhasználó e-mail címet adhat meg, csak a következő karaktereket engedjük:

    [@ _ , . -]
  - a numerikus értékeket mindig adott határok közé szorítsuk
  - az input hosszát minden esetben maximalizáljuk
- Javascript és SQL adatfeldolgozás esetén minden felhasználói adatbevitelt időzőjelekkel zárjunk.
- Biztosítsuk, hogy a kliens által a szervernek küldött adatokat (például űrlapok rejtett mezői, sütik) a kliens ne manipulálhassa.
- Az SQL adatbázisban csökkentsük a felhasználó privilégiumait (pl: `drop table`)

II. Lokalizáció készítő- és nyilvántartó rendszer

1. A bemutatandó alkalmazás alapvető működése

1.1. Fogalmak

1.1.1. Felhasználók típusai

Adminisztrátor

A web alkalmazás fejlesztőinek és üzemeltetőinek szűk csoportja, akik hatáskörébe tartozik többek között az alkalmazás karbantartása, hírek írása, projektek felvételének elfogadása, projektek törlése és az egyéb felhasználók jogainak kezelése.

Projekt menedzser

A regisztrált felhasználók egy olyan csoportja, akik projekteket regisztrálhatnak, és fordítandó fájlokat tölthetnek fel.

Szerkesztő

A regisztrált felhasználók egy olyan csoportja, akik a projektek honosítását végzik.

Végfelhasználó, eseti felhasználó

A felhasználók egy olyan csoportja, akik regisztráció nélkül megtekinthetik és letölthetik a lokalizált fájlokat.

1.1.2. Technikai fogalmak

Lokalizálandó projekt

A projekt menedzserek által az alkalmazásba honosítás céljából beregisztrált projekt.

Referencia fájl

A lokalizálandó projekt egy feltöltött fájlja, amely alapján a projekt honosítása készül.

Referencia nyelv

Egy lokalizálandó projekt alapnyelve. A projekt referencia fájljai ilyen nyelvű szövegeket tartalmaznak.

Lokalizált fájl

Egy adott projekt egy referencia fájljának a referencia nyelvtől különböző nyelvű változata.

Lokalizációs egység (unit)

A referencia fájl a fordítás szempontjából egy összefüggő része, amely egy-két szótól egészen akár hosszabb leírásig terjedhet.

Lokalizációs csoport (group)

Lokalizációs egységek logikailag összetartozó csoportja.
Például: csoport – egy párbeszéd; egységek – a párbeszédben lezajló mondatok.

Referencia fájl elemző (parser)

Olyan algoritmus, amely képes felismerni és elkülöníteni egy adott referencia fájlban lévő csoportokat és egységeket. Az elemző mindig egy adott projekthez, azon belül is egy (vagy több) konkrét referencia fájlhoz tarozik.

Lokalizációs egység állapota

Egy egységnek különböző állapotai lehetnek attól függően, hogy a fordítás mely fokozatában áll:

- *új*: olyan egység, amelyhez a fordítók még nem nyúltak.
- *lefordított*: olyan egység, amely már fordításra került.
- *lefordítandó*: olyan egység, amelyet már egyszer lefordítottak, viszont a referencia fájlban lévő tartalom megváltozott, ezért érvénytelenné vált a fordítás, ismét le kell fordítani.

Lokalizált fájl teljessége

Százalékos mérőszám, ami azt méri, hogy egy adott lokalizált fájl mennyire van honosítva. Értéke: a „lefordított” állapotú lokalizációs egységek száma osztva az összes egyég számával.

1.2. Követelmények meghatározása

Tervezés előtt tisztázni kell, hogy a leendő alkalmazás milyen alapvető funkciókat lásson el, milyen kényelmi szolgáltatásokat nyújtson. Az elkészítendő alkalmazás követelményeit elsősorban a honosítási feladatok gyakorlati tapasztalatai határozták meg.

A következő alapkövetelmények merültek fel:

- térben és időben korlátlanul elérhető
- többszintű felhasználó-kezelés
- nyílt, regisztrálás nélkül is elérhető
- letisztult, könnyen tanulható felhasználói felület
- projektek regisztrálása
- lokalizációs projektfájlok feltöltése, szerkesztése, törlése
- projektfájlokból más nyelvű verzió (lokalizált fájl) készítése
- lokalizált fájlok feltöltése, szerkesztése, törlése
- lokalizációs fájlok tárolása
- lokalizált fájlok szerkesztésekor látni lehessen a hozzá tartozó referencia fájl megfelelő egységeit és csoportjait
- a fájlok tartalmát az arra jogosult felhasználók szinkronban, egyidejűleg szerkeszthetik
- a fájlok módosításainak nyomon követése, korábbi verzió visszatöltése
- egy adott lokalizálásban résztvevők kezelése
- az egyes változásokról értesítés küldése a résztvevőknek
- a projektfájlok és a hozzá tartozó lokalizált fájlok letöltése
- biztonságos rendszer megvalósítása

További szolgáltatások:

- hírek
- fórum
- lehetőség az egyes projektekhez hozzászólás írására
- cserélhető felhasználói felület (témák kezelése)
- a referencia fájlok minden lokalizációs egységeihez megjegyzés írása, ami plusz információt nyújthat a fordítók számára a fordítandó szöveg szövegkörnyezetéről

Egyéb lehetőségek:

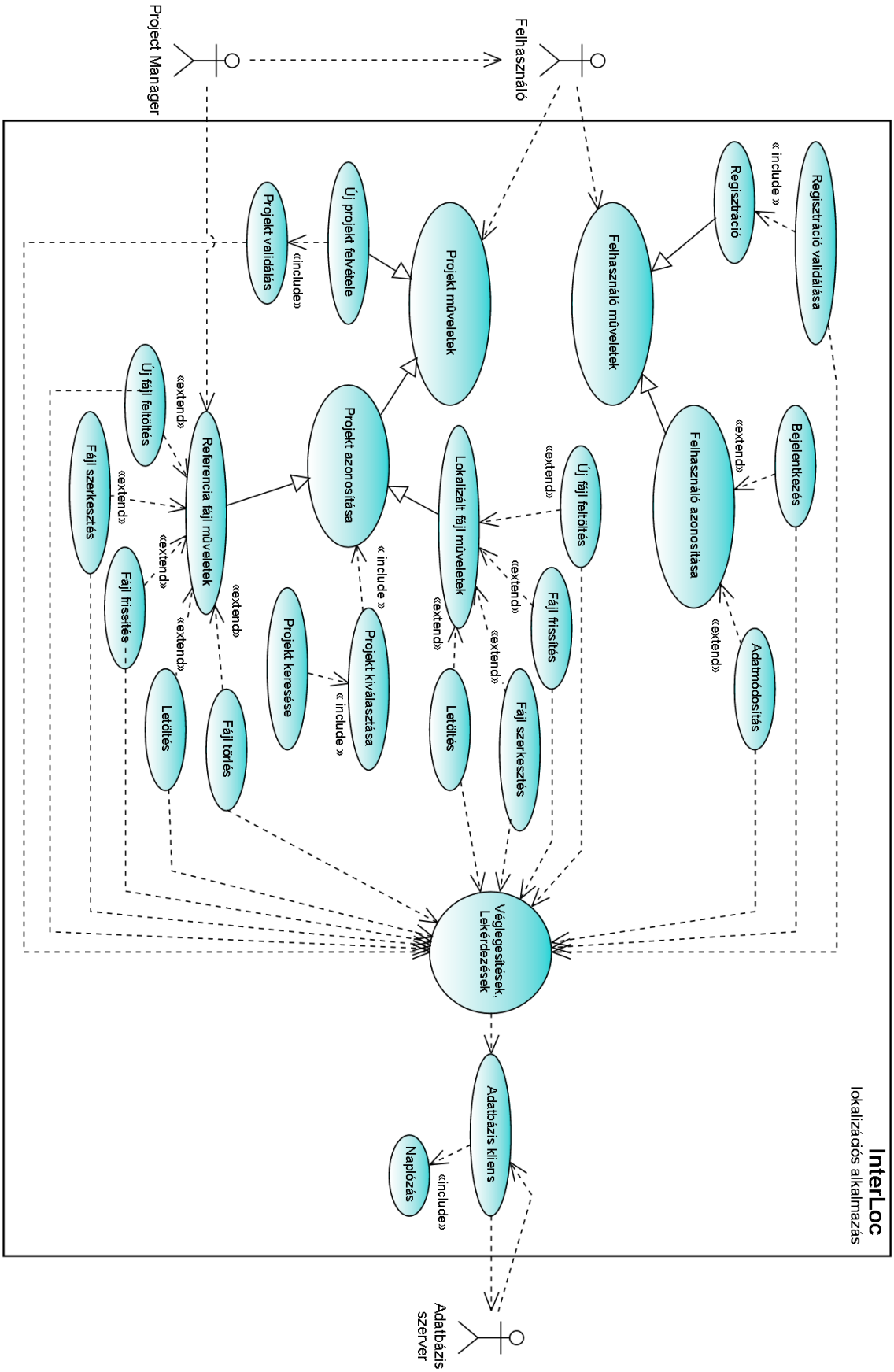
- belső levelezés, értesítés változásokról
- külső bővítmények kezelése: Google Translator, Altavista Babelfish, …

1.3. Diagramok

Miután felvázoltuk a rendszer felé támasztott követelményeket, vizuálisan is rögzíthetjük UML (Unified Modeling Language) diagramok formájában. Az UML diagramok azért hasznosak, mert így a tervezőkön kívül más személyek (megrendelők, programozók) is könnyen megérthetik a rendszer működését.

1.3.1. Használati eset diagram

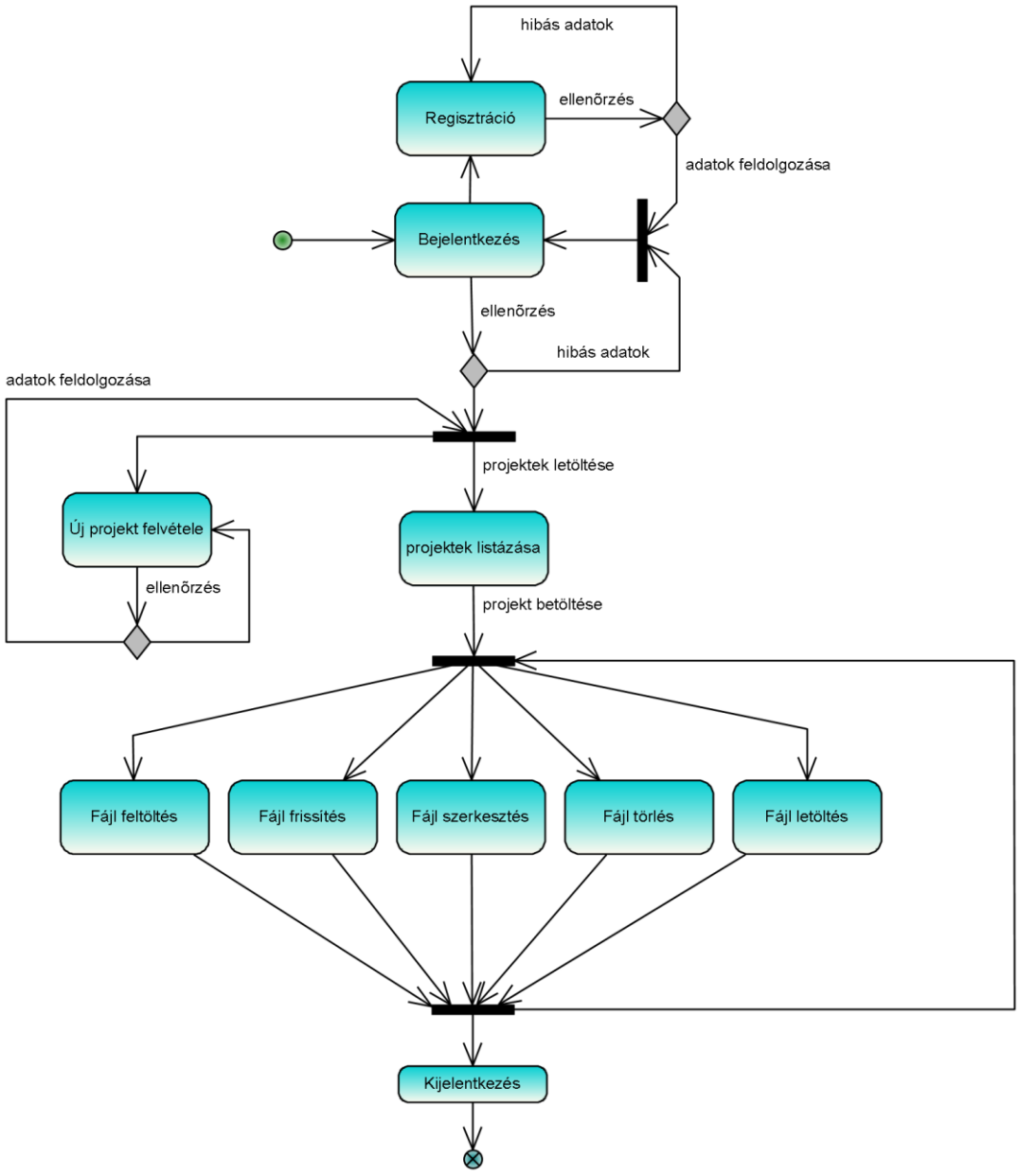
Az egyik legalapvetőbb UML diagram a használati eset diagram, amellyel vizuálisan, ábrák segítségével is definiálhatjuk az alkalmazással szemben támasztott követelményeket.
A használati eset diagramon elsősorban a felhasználtókat és az egyéb, rendszerhez kapcsolódó "elemeket" kell feltüntetni. Ezeket közös néven "aktoroknak" nevezzük és "pálcikaemberekkel" jelöljük. Jelen alkalmazásban a Felhasználó, Project Manager és Adatbázis nevű aktorokat definiáltam *(1. ábra)*.
A diagramon lévő ellipszisek a használati esetek, amelyek a rendszer funkcióit, külső kapcsolódási pontjait írják le.
A használati eset diagram segítségével egyértelművé válnak a rendszer funkciói, határai, az egyes felhasználók szerepkörei és az általuk elérhető funkciók. A rendszer esetleges hiányosságaira is fényt deríthet. Ugyanakkor a programozók számára egyértelművé válik,
hogy milyen funkciókat kell megvalósítaniuk. Vannak kiemelt szerepkörű használati esetek, ilyen például a Felhasználói műveletek, Projekt műveletek,
Két használati eset közötti kapcsolatot sztereotípiának nevezzük és szaggatott nyíllal jelöljük. A jelen esetben alkalmazott egyik sztereotípia az "`<<extends>`" sztereotípia, ami azt jelenti, hogy a nyíl kezdetén lévő használati eset által képviselt viselkedés opcionális a nyíl végén lévő használati esetre nézve. A másik alkalmazott sztereotípia az "`<<include>>`", amely azt mutatja, hogy a nyíl elején lévő használati eset által képviselt viselkedést a nyíl végén lévő használati eset magában foglalja.



1. ábra: Az alkalmazás használati eset diagramja

1.3.2. Állapot-átmenet diagram

Az állapot-átmenet diagram segítségével nyomon követhetjük, hogy milyen folyamatok fognak a rendszeren belül lezajlani. A rendszert állapotokkal jellemezzük, és egyik állapotból a másikba művelet, esemény hatására megy át. A rendszer tetszőleges időpillanatban a lehetséges állapotok egyikében van.



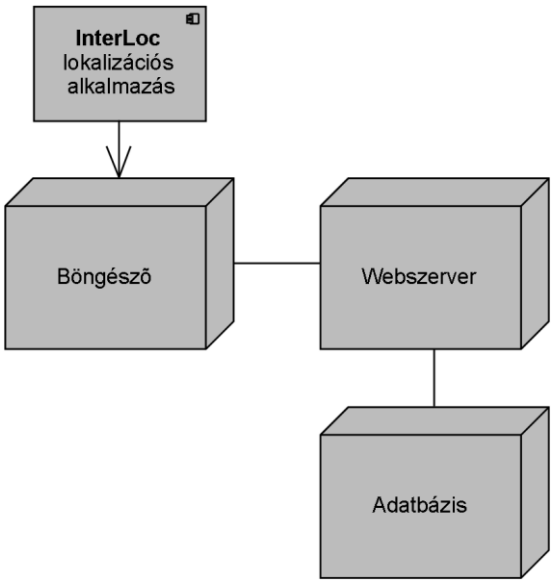
2. ábra: Állapot-átmenet diagram

1.3.3. Alkalmazási diagram

Alkalmazási diagrammal a rendszer elemeit és a köztük lévő fizikai kapcsolatot adhatjuk meg.

Az alkalmazás-diagram alapeleme a csomópont, amely egy számítógépes elemet, általában egy hardvert jelent.

Konkrét esetben:



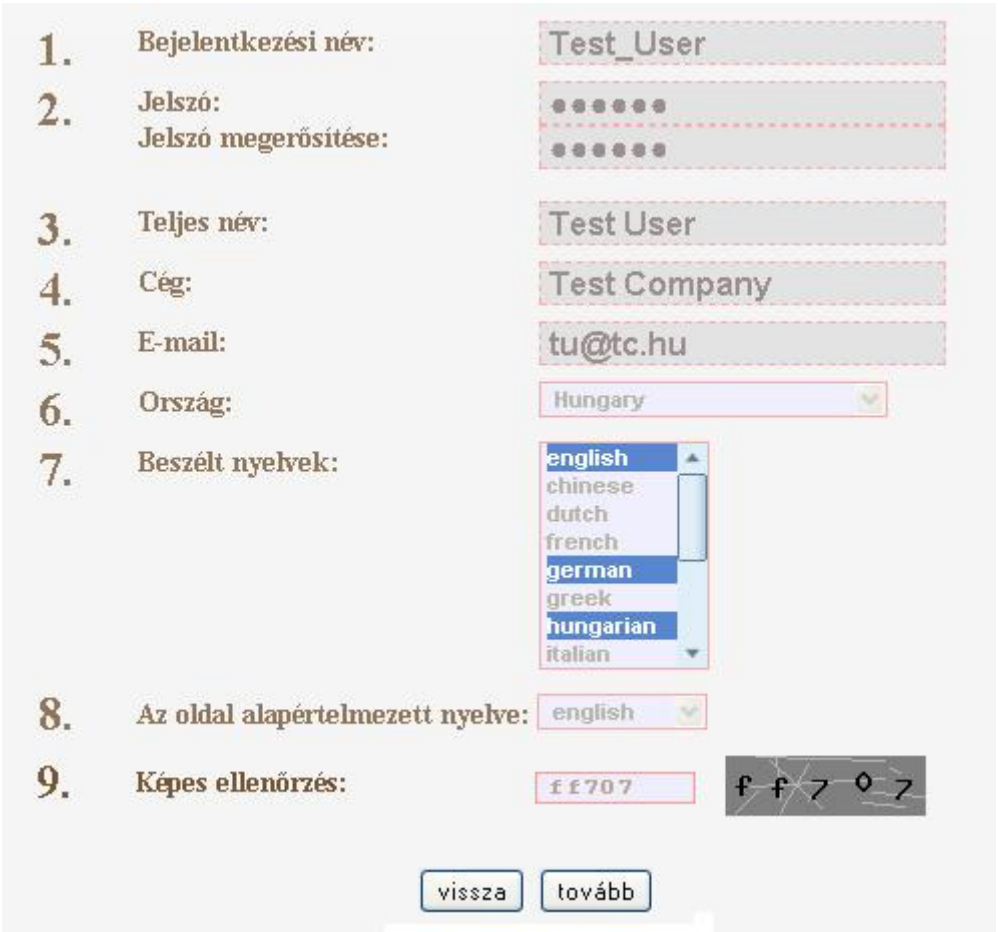
3. ábra: Alkalmazási diagram

- egy böngésző: ahol az alkalmazás megjelenik
- egy web szerver: ahol az alkalmazás fut
- egy adatbázis szerver: ahol az alkalmazás adatokat tárol

1.4. Forgatókönyvek

1.4.1. Felhasználó regisztrációja

A főoldal jobb felső részén a bejelentkező űrlap mellett található a regisztrációs űrlapra mutató hivatkozás.



4. ábra: Új felhasználó regisztrálása űrlap

Egy új felhasználó a regisztráció során egyesével, egymás után adhatja meg az űrlapon szükséges adatokat. Minden megadott információt ellenőriz a rendszer és amennyiben nem megfelelő, nem engedi tovább a következő lépésre. Lehetőség van visszalépésre is. A validálás egyké eset kivételével a kliensen történik JavaScript segítségével.

A regisztráció során kötelezően kitöltendő adatok és azok hibaesetei:

- **felhasználói azonosító**: beviteli mező. Egyedi azonosító, amely egy adott felhasználót egyértelműen azonosít. A bejelentkezés ezen azonosító megadásával történik.
  Hibaesetek:
  - a megadott azonosító túl rövid: minimum 5 karakter hosszúnak lennie kell.
  - tiltott karaktert tartalmaz: a helyes azonosító az alábbi reguláris kifejezéssel adható meg:

    [^0-9a-zA-Z]
  - már létezik ilyen azonosító: a rendszer már tartalmaz a megadottal megegyező azonosítót. Az ellenőrzés a szerveren (adatbázisban) történik Ajax segítségével.
- **felhasználói jelszó és annak megerősítése:** beviteli mezők. A felhasználónak az azonosítója mellett a jelszavát is meg kell adnia a sikeres bejelentkezéshez.
  Hibaesetek:
  - a két jelszó nem egyezik meg.
  - a jelszó túl rövid: minimum 5 karakter hosszúnak lennie kell.
  - a jelszó túl egyszerű: a nagyobb biztonság érdekében a jelszónak tartalmaznia kell kisbetűt, nagybetűt és számot is.
- **teljes név:** beviteli mező. A felhasználó teljes neve, amellyel az alkalmazásban megjelenik.
- **e-mail cím:** beviteli mező. A felhasználó valós e-mail címe.
  Hibaesetek:
  - a megadott cím formátuma helytelen: az e-mail címnek tartalmaznia kell egy ’@’, majd azt követően egy ’.’ karaktert.
- **ország:** legördülő lista.
- **beszélt nyelvek:** legördülő lista több kijelölési lehetőséggel. A felhasználó megadhatja az általa beszélt nyelveket.
- **alapértelmezett nyelv:** legördülő lista. A web alkalmazás megjelenési nyelve. Kiválaszthatja, milyen nyelven szeretné használni az alkalmazást.
- **képellenőrzés:** beviteli mező és kép. A nagyobb biztonság érdekében a képen lévő számokat és betűket be kell írni a beviteli mezőbe.
  Hibaesetek:

- a megadott karakterek nem egyeznek a képen lévő karakterekkel. Az ellenőrzés Ajax segítségével történik.

Amennyiben minden szükséges információt megadott a felhasználó, a szerver regisztrálja őt és előállít – a megadott e-mail cím és egy pszeudovéletlen szám segítségével – egy lenyomatot, amelyet „megerősítő kódnak” nevezünk. Ezután a felhasználó részére, a regisztrációs űrlapon megadott e-mail címre egy megerősítő üzenetet küld, amely egy hiperhivatkozást tartalmaz. A hivatkozás magában foglalja a megerősítő kódot. A regisztráció csak azután válik véglegessé, miután a felhasználó a hivatkozásra kattintott, ezzel biztosítva az alkalmazást a valós és hiteles regisztrációról. Ha ezt a felhasználó 48 óra elteltével is elmulasztja, a regisztrációja automatikusan törlődik és azonosítója felszabadul.
A sikeres regisztrációt követően a felhasználó automatikusan bejelentkezve a főoldalra kerül.

1.4.2. Felhasználó bejelentkezése

A főoldal jobb felső részén található a bejelentkező űrlap.

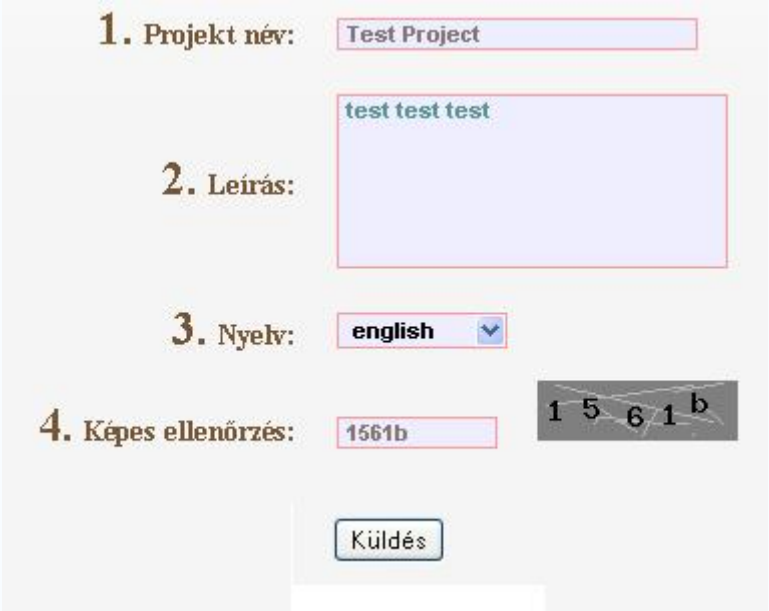
5. ábra: Felhasználó bejelentkezése űrlap

Bejelentkezni csak a véglegesen regisztrált felhasználók tudnak a regisztrációban megadott azonosító és jelszó beírásával, továbbá lehetőség van annak megjelölésére, hogy az alkalmazás minden első betöltése esetén automatikusan léptesse be őt. Ez ugyan nem túl biztonságos, viszont gyakori használat esetén praktikus megoldás lehet.

Elfelejtett jelszó:
A bejelentkezési űrlap mellett található egy hivatkozás annak érdekében, ha egy regisztrált felhasználó elfelejtené a jelszavát. A jelszó e-mail címre való kiküldése biztonsági okokból nem lehetséges, mivel azt a rendszer nem direkt tárolja az adatbázisban, hanem kódolva. Helyette a felhasználó azonosítóját beírva a rendszer az eltárolt e-mail címre új, automatikusan generált jelszót küld ki, amelyet a felhasználó belépés után bármikor megváltoztathat.

1.4.3. Új project regisztrálása

A projekt menedzser szintű felhasználók új projekteket regisztrálhatnak a rendszerbe.



6. ábra: Projekt regisztrálása űrlap

Egy új projekt regisztrációjának logikája hasonlít a felhasználó regisztrációhoz: a projekt menedzsernek egyesével, egymás után kell megadnia az űrlapon a szükséges adatokat. Minden megadott információt validál a rendszer és amennyiben nem megfelelő, nem engedi tovább a következő lépésre. A validálás nagyrészt a kliensen történik JavaScript segítségével.

A regisztráció során kötelezően kitöltendő adatok és azok hibaesetei:

- **neve**: beviteli mező. A regisztrálandó projekt egyedi neve.
  Hibaesetek:
  - a név túl rövid: egy projekt nevének minimum 5 karakter hosszúnak kell lennie.
  - A projekt neve foglalt: a megadott névvel már létezik projekt. Az ellenőrzés a szerveren történik Ajax használatával.
- **rövid leírása:** beviteli mező. A projektről rövid leírást kell megadni, amely megismerteti a felhasználókat a projekttel. A leírásban szereplő szavak kulcsszavakként szerepelhetnek a részletes keresésekben.

- **nyelve:** legördülő lista. A projekt alapnyelvét meg kell adni, innen fogja majd tudni a rendszer, hogy milyen nyelvű referenciafájlokat fog tartalmazni.

A sikeres regisztrációt követően az alkalmazás a létrehozott projekt oldalára ugrik.

1.4.4. Projekt kiválasztása

A látogatók a "projektek listája" menüpontból elérhető oldalon tudnak a regisztrált projektek közül egy konkrétat kiválasztani. A listában minden projektről az alábbi információk jelennek meg:

- **név**: a project neve
- **feltöltő**: annak a felhasználónak a teljes neve, aki létrehozta a projektet
- **létrehozva**: a regisztrálás dátuma
- **utolsó módosítás**: az az időpont, amikor a projecten utoljára módosítottak. Ilyen módosítás az új referencia fájl feltöltése, új lokalizált fájl léterhozása, szerkesztések, feltöltések.

Az alkalmazás egy későbbi továbbfejlesztésében az alábbi információkkal lehetne bővíteni:

- **referencia fájlok száma**
- **átlagos teljesség**: a projectben található összes lokalizált fájl átlagos teljessége.
- **relatív teljesség**: a projectben található azon lokalizált fájlok átlagos teljessége, amely olyan nyelvűek, amelyeket a felhasználó beszél.
- **érdeklődés mértéke:** grafikailag megjelenített mérőszám, ami azt jelzi, hogy mennyire népszerű a project.

7. ábra: Regisztrált projektek listája

Rendezés

A listázott projekteket minden megjelenített oszlop szerint lehet növekvő és csökkenő sorrendben is rendezni azáltal, hogy a felhasználó az egyes oszlopok nevére kattint.

Keresés

Lehetőség van a megjelenített lista szűkítésére is keresési feltétel megadásával. A keresés úgy történik, hogy a megadott szótöredéket először a projektnevek között keresi meg, majd a feltöltők nevei között.

Kiválasztás

A kiválasztandó projekt sorában bárhova kattintva a projekt oldalára ugorhatunk.

A projekttel kapcsolatos további funkciók a projekt oldaláról érhetők el (*6.ábra*).

8. ábra: Projekt oldal

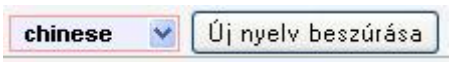
1.4.5. Új referencia fájl feltöltése

A projekt oldalon található az új referencia fájl feltöltésére szolgáló űrlap. Projekt fájlokat csak az a projekt menedzser szintű felhasználó tud feltölteni, aki az adott projektet beregisztrálta a rendszerbe. Az alkalmazás egy későbbi továbbfejlesztésében implementálandó egy olyan sajátosság, amely lehetőséget biztos arra, hogy egyéb projekt menedzserekre is kiterjeszthető legyen ez a jog, azaz több „üzemeltetője" legyen egy projektnek.
Új referencia fájlt csak és kizárólag az a felhasználó tölthet fel, aki a projektet létrehozta. Ez a jogkör a későbbiekben bővülni fog.

9. ábra: Új referencia fájl feltöltése űrlap

1.4.6. Új lokalizált fájl létrehozása

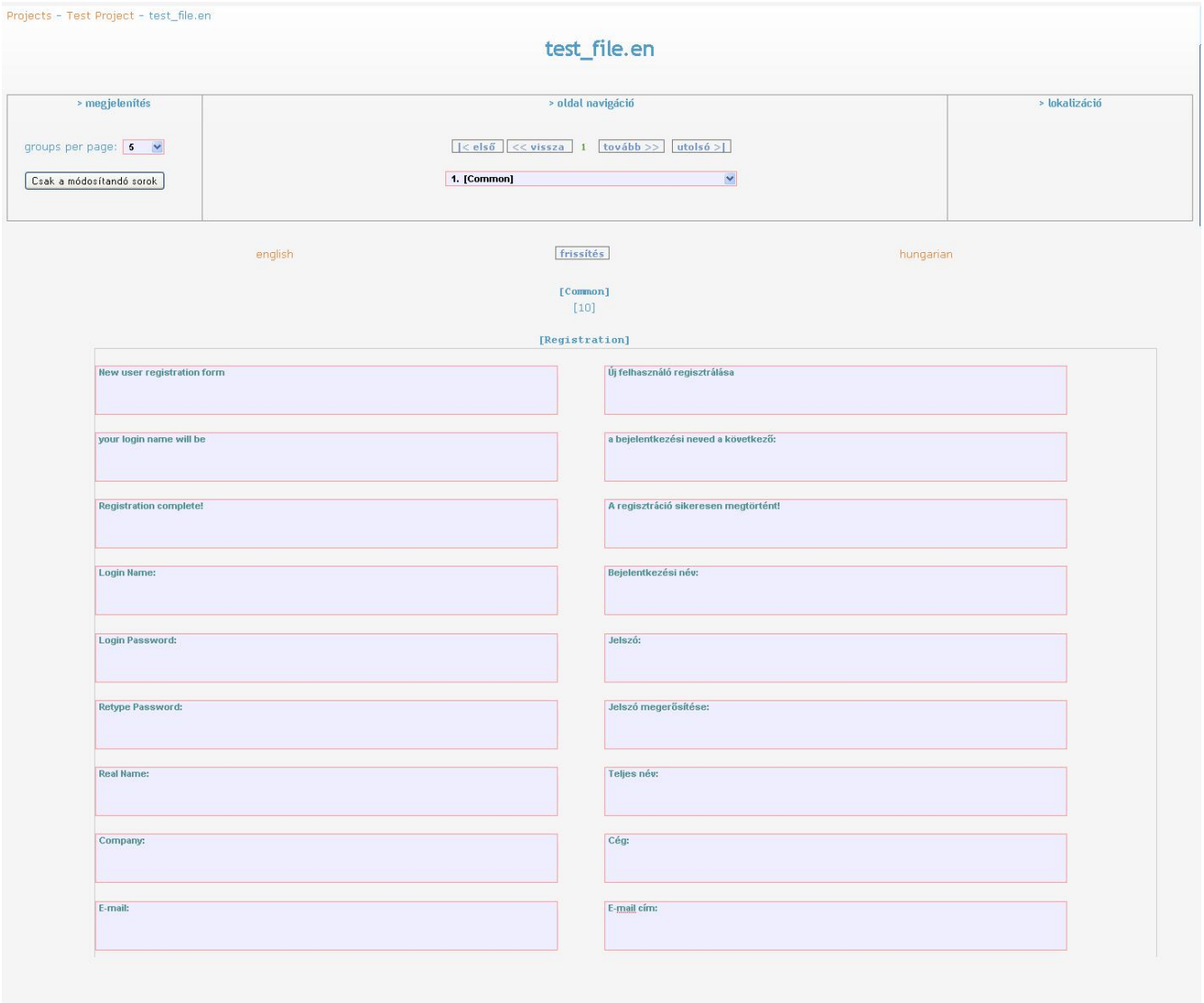
Egy adott referencia fájlhoz egy addig még nem létező nyelvű lokalizációs fájl létrehozása mindössze abból áll, hogy a felhasználó kiválasztja a megfelelő nyelvet és elküldi az űrlapot. Az alkalmazás beregisztrálja az új fájlt az adatbázisba, majd újratölti az oldalt.
Új lokalizált fájlt bármely regisztrált és bejelentkezett felhasználó létre tud hozni.



10. ábra: Új nyelv beszúrása űrlap

1.4.7. Fájlok szerkesztése

A projekt oldalon egy fájlra kattintva megjelenik a szerkesztés oldal.



11. ábra: : Lokalizált fájl szerkesztése

Mivel a lokalizációs fájlok mérete és ezzel együtt a lokalizációs egységek száma is egészen nagy lehet, az egy oldalon történő teljes megjelenítés kezelhetetlenné és átláthatatlanná válna, ezért oldalakra tagoltan lehet a fájl teljes tartalmához hozzáférni. Az egy oldalon megjelenő lokalizációs csoportok száma alapértelmezetten 15, de a felhasználó bármikor megváltoztathatja ezt az oldal felső részén található legördülő menü segítségével. A változtatás természetes következménye az oldalak számának megváltozása.

A lapozásra több lehetőség is van:

- oldalszámra kattintás: az aktuális oldal mellett az előző és következő öt oldalszám ki van írva.
- léptető gombokra kattintás: az oldalszámok mellett négy gomb található, amelyek rendre az első, az előző, a következő és az utolsó oldalra mutatnak.
- csoport kiválasztása: az oldalszámok alatt egy legördülő menü található, amelynek elemei a fájlban lévő csoportok nevei. Egy csoportra kattintva az az oldal jelenik meg, amelyiken a csoport található.

Lehetőség van arra, hogy a fájlnak ne a teljes tartalmát jelenítse meg az alkalmazás, hanem csak azokat a lokalizációs egységeket, amik nincsenek még az adott nyelvre lefordítva. Ezzel a funkcióval a fordítás a teljesség közeledtével gördülékenyebben haladhat.

A fájl tartalma csoportokra bontva jelenik meg, minden csoport alatt annak egységei két oldalra rendezve: baloldalt a referencia fájlban található szöveg, míg jobb oldalon a lokalizált fájlban lévő szöveg látszik. Így egy lokalizált egység fordítása közben a referencia szöveget is láthatja a szerkesztő.
Az alkalmazás egy későbbi továbbfejlesztésében hozzáadandó egy olyan lehetőség, amely megengedi más, nem referencia fájl megjelenítését is a bal oldalon, vagy akár a fájl különböző nyelvű változatainak egyszerre történő megjelenítését is, hiszen előfordulhat, hogy a szerkesztő nem beszéli a referencia nyelvet.
Egy csoport nevére kattintva annak tartalma elrejthető; előfordulhat, hogy egy hosszabb csoporttal nem szándékozik foglalkozni a felhasználó, ezért annak elrejtése átláthatóbbá teheti az aktuális oldalt. A csoportnévre való újabb kattintással ismét megjeleníthető a tartalom.
Minden lokalizációs egység mellett található egy jelzés, amely mutatja annak állapotát.

A jobb oldali oszlop tartalma nem módosítható, hiszen csak referenciaként, segítségként jelenik meg. A bal oldali oszlop szövegei csak a bejelentkezett felhasználók számára módosíthatóak.
A módosítás annyiból áll, hogy a szerkesztő az adott szövegmező tartalmát módosítja, majd amint a mező elveszti a fókuszt (egérrel elkattint belőle, TAB billentyű megnyomásával a következő egységre ugrik…), az új tartalom azonnal az adatbázisban mentésre kerül. A mentés folyamata a felhasználótól rejtve történik, őt semmilyen várakozásra vagy az oldal újratöltésére nem kényszeríti.

Egy adott lokalizált fájl szerkesztése így konkurens módon történhet: akár több felhasználó is szerkesztheti egy időben. Amint egy új fordítás kerül a fájlba, az egy időben azt szerkesztő többi felhasználónál frissül az adott lokalizációs egység tartalma, így azok azonnal értesítést kapnak a változásról.
Előfordulhat azonban olyan eset, amikor két vagy több felhasználó ugyanazt az egységet módosítja. Ilyenkor az eltérések feloldása úgy történik, hogy amikor az egyik felhasználó befejezi az egység szerkesztését, a másik felhasználónál nem frissül be egyből a tartalom, hanem lehetőséget kap arra, hogy összehasonlítsa az új szöveget a saját változtatásával és döntsön arról, hogy elfogadja-e a másik fordítást vagy figyelmen kívül hagyja és folytatja saját fordítását.

A referencia fájlok szerkesztése mindössze annyiban különbözik a lokalizált fájlokétól, hogy ott csak egyetlen oszlop látható, hiszen nincsen olyan más fájl, amit referenciaként felhasználhatna a felhasználó.
Referencia fájlt csak a Projekt Manager módosíthat.

1.4.8. Fájlok letöltése

A projekt oldalon bárki letöltheti a lokalizált fájlok aktuális verzióját. A „Letöltés” gombra kattintáskor a rendszer megnézi, hogy az adatbázisban tárolt aktuális tartalom ki van-e már mentve a szerverre a projektnek megfelelő szerkezetű fájlként. Ha nincs, akkor ezt megteszi. Ezután, illetve ha már létezik fájlként a legfrissebb tartalom, mindössze megkeresi azt, és a böngészőkbe beépített fájlmentés funkcióval felajánlja annak mentését.

2. A bemutatandó alkalmazás technikai leírása

A következőkben az alkalmazás fejlesztése során elkészített főbb osztályokat és állományokat, valamint azok funkcionalitását mutatom be.

2.1. Bemutatás a PHP oldaláról

A fejlesztés során igyekeztem különválasztani a felhasználói felületet az üzleti logikától.

2.1.1. A `database` osztály

A `database` osztály segítségével érhetjük el az adatbázisunkat. Az osztály egy általános interfészt biztosít az adatbázisműveletekhez, amely alatt bármilyen adatbáziskezelő rendszert használhatunk, legyen az MySQL, PostgreSQL vagy MsSQL.
Tulajdonságai közé tartozik az adatbázishoz való kapcsolódás, valamint a szükséges műveletek elvégzése: lekérdezés, módosítás, beszúrás, törlés, tranzakciók kezelése és tárolt eljárások hívása.

Attribútumok:

- `host`: az adatbázisrendszer
- `user`: az adatbázisrendszer felhasználójának neve
- `pass`: az adatbázisrendszer felhasználójának jelszava
- `dbas`: az alkalmazás által használt adatbázis neve
- `connection`: a kapcsolat azonosítója állandó kapcsolat esetén
- `fetch_mode`: a lekérdezés eredményének egy sorát

2.1.2. A `User` osztály

A `Users` adatbázis táblának megfelelő osztály. Tartalmazza egy felhasználó adatbázisban tárolt adatait, továbbá olyan metódusokat, amelyekkel felhasználói műveleteket lehet elvégezni.

Attribútumok

- `data`: asszociatív tömb, amely a Users adatbázistábla megfelelő sorának adatait tartalmazza.

- `log`: a `Log` osztály példánya. Bővebben kifejtve a 2.1.6. részben.

Konstruktor

Példányosításkor megnézi, hogy van-e a felhasználó gépén az alkalmazáshoz tartozó süti elmentve. A program akkor tárol sütit, ha a felhasználó bejelentkezéskor bepipálta az "automatikus bejelentkezés" mezőt. Ammennyiben talál egyet, a benne tárolt felhasználói névvel és jelszóval megpróbálja bejelentkeztetni őt.

Metódusok

- `login()`: belépteti a felhasználót a megadott névvel és jelszóval.
- `setData()`: a `data` attribútumot feltölti a felhasználó adatbázisban tárolt adataival.
- `logOut()`: kijelentkezteti a felhasználót és megszünteti a munkamenetet.
- `isLogged()`: eljárás, amely be nem jelentkezett felhasználó esetén a kezdőoldalra átirányítja a felhasználót. Védett tartalom megtekintésekor hívódik.
- `register()`: a regisztrációs űrlap által küldött védett paraméterek alapján – , amelyek a POST környezeti változóban tárolódnak – regisztrálja a felhasználót.
- `serializeIt()`: karakterlánccá alakítja a példányt a későbbi felhasználás érdekében. A szerializált objektumot munkamenetben menti.
- `deserializeIt()`: a munkamenetben tárolt karakterláncot visszaalakítja a megfelelő objektummá.
- `sendEmail()`: E-mail-t küld PHP segítségével a webszerverhez tartozó SMTP kliensen keresztül.

2.1.3. A `project` osztály

A `Projects` adatbázis táblának megfelelő osztály. Tartalmazza egy project, szoftver adatbázisban tárolt adatait, továbbá minden olyan metódust, amelyekkel az adott project-en műveleteket lehet elvégezni.

Attribútumok

- `data`: asszociatív tömb, amely a `Projects` adatbázistábla megfelelő sorának adatait tartalmazza.
- `log`: a `Log` osztály példánya. Bővebben kifejtve a 2.1.6. részben.

Konstruktor

Az osztály egy üres konstruktort tartalmaz.

Metódusok

- `set()`: a paraméterben megadott azonosítójú project adataival tölti fel a példányt
- `setData()`: a `data` attribútumot feltölti az adatbázisban tárolt adatokkal.
- `create`(): az új project létrehozásakor kitöltött űrlap védett paraméterei alapján létrehoz egy új projectet.
- `delete`(): a példányban szereplő projectet a hozzá kapcsolódó fájlokkal együtt törli.
- `upload_newref`(): új referenciafájlt tölt fel a projecthez. A fájlt az űrlap elküldése után a `FILES` környezeti változóban éri el.
- `create_newloc(refFileID, langID)`: a paraméterben átadott referencia fájlhoz létrehoz egy új nyelvű verziót.
- `upload_refUpdate(refFileID)`: az adott referencia fájlból feltölt egy új verziót.
- `upload_locUpdate(locID)`: egy adott lokalizált fájlból új verziót tölt fel.
- `changeContent_loc(locID,unitID,content)`: módosítja az adott lokalizált fájl egy egységének tartalmát.
- `changeContent_ref(refID,unitID,content)`: módosítja az adott referenciafájl egy egységének tartalmát.

- `getParserID(locID)`: visszaadja annak a fájlfeldolgozó osztálynak az azonosítóját, amellyel az adott lokalizált fájlt be tudjuk olvasni.
- `downloadFile(filepath,filename)`: a szerveren lévő lokalizált fájl letöltését intézi.
- `serializeIt()`: karakterlánccá alakítja a példányt a későbbi felhasználás érdekében. A szerializált objektumot munkamenetben menti.
- `deserializeIt()`: a munkamenetben tárolt karakterláncot visszaalakítja a megfelelő objektummá.

Minden olyan esetben, ahol fájlokkal dolgozik az osztály, a felhasználó által az űrlap kitöltésekor megadott fájlt a PHP `$_FILES` környezeti változójából éri el.

2.1.4. A `log` osztály

A hibakeresés és nyomkövetés fontos eszköze lehet a log fájlok készítése.
Az osztályt a `user` és `project` osztályok példányosítják.

Attribútumok

- `logLevel`: a logolás szintje. Megmondja, hogy milyen típusú logok kerüljenek fájlba mentésre. A log sorok négy fajtáját különböztetem meg: változtatás, lekérdezés, hiba, debugolás.
- `path`: a fájl elérési útvonala.

Konstruktor

Az osztály példányosításkor paraméterként megkapja a mentendő fájl útvonalát. Ez a `user` osztály esetén „`log/users/aktualis_user_id`”, míg a `project` osztályban „`log/projects/aktualis_project_id`”.

Metódusok

- `log( logLevel, tartalom)`: megnezi, hogy az első paraméterként megkapott szintet kell-e menteni. Ha igen, akkor a második paramétert időkóddal együtt a fájlba írja.

2.1.5. A `config.php` állomány

Ez az állomány tartalmazza az alkalmazás által használt minden konstans és globális változó definícióját.
Itt állítom be az adatbázishoz való kapcsolódás paramétereit, a logolás módját, az alkalmazás működését befolyásoló paramétereket, valamint az adatbázisba tükrözött konstansokat.

2.1.6. A `core_func.php` állomány

Ez az állomány olyan általános függvényeket és eljárásokat tartalmaz, amelyekre a fejlesztés során számos helyen szükség volt, legyen szó akár általános PHP használatról, akár a konkrét alkalmazás logikájáról. Éppen ezért a fájl betöltése a `config.php` –ban történik meg.

2.1.6. Egyéb osztályok és állományok

A `Themes` osztály és a hozzá tartozó állományok

Az oldal különböző kinézeteinek (témáinak) megjelenítéséért felelős osztály. A bejelentkezett felhasználó által beállított témának megfelelően az alkalmazás ”`themes/`” könyvtárában lévő témák közül kiválaszt egyet és a benne lévő CSS fájlt betölti. A CSS-hez képek is tartozhatnak.

Az `array_to_js` osztály

A fejlesztés során számos esetben szükség volt arra, hogy PHP tömböket adjak át a JavaScriptnek. Ezt a procedúrát könnyítette meg nagyban az osztály.

A `safeCode` állomány

Az állomány a regisztrációknál alkalmazott képes ellenőrzéseket valósítja meg. Létrehoz egy kép objektumot, amelyen pszeudovéletlen számok és betűk találhatók. Ez a kép jelenik meg a regisztrációknál.

A `locale` osztály és a hozzá tartozó állományok

Ez az osztály felel magának az alkalmazásnak a lokalizálhatóságáról. Egyetlen metódusa egy azonosítót kap paraméterül. A referencia fájl és az elkészített lokalizált fájlok a „`/locale/`” könyvtárban találhatóak. Az osztály által választott fájl:

- bejelentkezett felhasználó esetén a beállított nyelvű, amennyiben az nem létezik, az alapértelmezett angol nyelvű fájl.
- nem bejelentkezett felhasználó esetén a böngésző nyelvével megegyező, amennyiben az nem létezik, az alapértelmezett angol nyelvű fájl.

A metódus azzal a szöveggel tér vissza, amely a kiválasztott fájlban a kapott azonosítóval szerepel.

2.2. Bemutatás a MySQL oldaláról

2.2.1. Adatbázis

Az alkalmazás számos adatbázistáblát használ. Ezekben a táblákban tárolom a felhasználók adatait, a szoftverek fájljainak és azok lokalizálásainak adatait, valamint a fórumra vonatkozó adatokat.

A táblákat az alkalmazás telepítésekor kell létrehozni az adatbázisban.

A következőkben az egyes táblák szerkezetét fogom bemutatni.

2.2.2. `Users` és `UserStats` táblák

Mindkét adatbázis tábla a felhasználókról tárol adatokat.

A `Users` tábla nagyrészt a felhasználó regisztráláskor megadott adatait tartalmazza. Az alapadatok mellett a felhasználók jogait, a regisztráció véglegesítését és a honlapon használt téma azonosítóját tartalmazza. A felhasználók jelszavát *md5* titkosítással tárolom, így sem én, sem

az illetéktelen támadók nem tudják visszafejteni őket. Ennek következtében elfelejtett jelszó esetén kap egy újat a felhasználó, amelyet később módosíthat.

A `UserStats` tábla felhasználókhoz tartozó statisztikákat tárol, úgymint a regisztráció idejét, a bejelentkezések számát és a honlapon töltött idő hosszát. Továbbá tartalmaz egy külső kulcsot, amely a `Users` tábla elsődleges kulcsára hivatkozik.

A táblákat az alábbi SQL utasítással hoztam létre:

```
CREATE TABLE `users` (
  `ID`            int(11)          NOT NULL auto_increment,
  `LoginName`     varchar(20)      default NULL,
  `LoginPasswd`   varchar(50)      default NULL,
  `Name`          varchar(50)      default NULL,
  `Company`       varchar(50)      default NULL,
  `Email`         varchar(50)      default NULL,
  `Country`       varchar(30)      default NULL,
  `PortalLangID`  smallint(11)     default NULL,
  `SpokenLangIDs` varchar(50)      default NULL,
  `Permissions`   int(11)          NOT NULL default '0',
  `Validated`     int(11)          NOT NULL default '0',
  `Theme`         int(11)          NOT NULL default '0',
  `AutoLogin`     int(11)          NOT NULL default '0',
  PRIMARY KEY  (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE `userstats` (
  `ID`            int(11)          NOT NULL default '0',
  `RegDate`       datetime         NOT NULL default '0000-00-00
00:00:00',
  `LoginNum`      int(11)          NOT NULL default '0',
  `TimeSpent`     time             NOT NULL default '00:00:00',
  `IsOnline`      tinyint(4)       NOT NULL default '0',
  PRIMARY KEY  (`ID`),
  CONSTRAINT `FK_userstats_1` FOREIGN KEY (`ID`)
                      REFERENCES `users` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC;
```

2.2.3. `Projects` tábla

Ebben a táblában tárolom a regisztrált projektek alapadatait, többek közt a nevét, nyelvét, annak a felhasználónak az azonosítóját, aki beregisztrálta, valamint a megjegyzésekhez szükséges azonosítót.

A táblát az alábbi SQL utasítással hoztam létre:

```
CREATE TABLE `projects` (
  `ID`                int(11)     NOT NULL auto_increment,
  `CompanyName`       varchar(30) character set utf8 default NULL,
  `Name`              varchar(50) character set utf8 default NULL,
```

```
  `Description`         text        character set utf8,
  `LangID`              int(11)     default NULL,
  `OwnerID`             int(11)     NOT NULL,
  `DateCreation`        datetime    NOT NULL,
  `DateLastMod`         datetime    NOT NULL,
  `CommentTopicID`      int(11)     NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `FK_projects_1` (`LangID`),
  KEY `FK_projects_2` (`OwnerID`),
  KEY `FK_projects_3` (`CommentTopicID`),
  CONSTRAINT `FK_projects_3` FOREIGN KEY (`CommentTopicID`)
                             REFERENCES `forum_topics` (`ID`),
  CONSTRAINT `FK_projects_1` FOREIGN KEY (`LangID`)
                     REFERENCES `languages` (`ID`),
  CONSTRAINT `FK_projects_2` FOREIGN KEY (`OwnerID`)
                     REFERENCES `users` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC;
```

2.2.4. `Projects_files_ref` tábla

Ez a tábla a rendszerben lévő referencia fájlokról tartalmaz információkat.

A "`ProjectID`" hivatkozik a `Projects` tábla azon rekordjára, amely projekt referencia fájlja.

Ezen kívül tárolja a fájl nevét, nyelvének azonosítóját, a létrehozásra vonatkozó információkat, valamint annak a fájl elemzőnek az azonosítóját, amely az adott fájl tartalmát képes feldogozni.

A táblát az alábbi SQL utasítással hoztam létre:

```
CREATE TABLE `project_files_ref` (
  `ID`            int(11)          NOT NULL auto_increment,
  `ProjectID`     int(11)          NOT NULL default '0',
  `Name`          varchar(30)      default NULL,
  `LangID`        smallint(6)      NOT NULL,
  `ParserID`      smallint(6)      default '1',
  `GenDate`       datetime         default NULL,
  `GenUserID`     int(11)          default NULL,
  PRIMARY KEY  (`ID`),
  KEY `FK_project_files_ref_1` (`ProjectID`),
  KEY `FK_project_files_ref_3` (`GenUserID`),
  CONSTRAINT `FK_project_files_ref_1` FOREIGN KEY (`ProjectID`)
                                  REFERENCES `projects` (`ID`),
  CONSTRAINT `FK_project_files_ref_3` FOREIGN KEY (`GenUserID`)
                                  REFERENCES `users` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;
```

2.2.5. `Project_files_loc` tábla

Ez a tábla a rendszerben lévő lokalizált fájlokról tartalmaz információkat.

A "`RefFileID`" hivatkozik a `Projects_files_ref` tábla azon rekordjára, amely referencia fájl lokalizáltja.

Ezen kívül tárolja a nyelv azonosítóját, a létrehozásra vonatkozó információkat és a teljesség mértékét.

A táblát az alábbi SQL utasítással hoztam létre:

```
CREATE TABLE `project_files_loc` (
  `ID`            int(11)           NOT NULL auto_increment,
  `RefFileID`     int(11)           default NULL,
  `LangID`        smallint(11)      default NULL,
  `Flags`         int(11)           NOT NULL,
  `Progress`      smallint(6)       default NULL,
  `GenDate`       datetime          NOT NULL,
  `GenUserID`     int(11)           NOT NULL,
  `LastUpdated`   datetime          NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `FK_project_files_loc_1` (`RefFileID`),
  KEY `FK_project_files_loc_2` (`GenUserID`),
  CONSTRAINT `FK_project_files_loc_2` FOREIGN KEY (`GenUserID`)
                            REFERENCES `users` (`ID`),
  CONSTRAINT `FK_project_files_loc_1` FOREIGN KEY (`RefFileID`)
                            REFERENCES `project_files_ref` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC;
```

2.2.6. `Project_files_data_info`, `Project_files_data` táblák

Ez a két tábla tárolja az egyes referencia és lokalizált fájlok tartalmait egységekre bontva.

A `Project_files_data_info` a lokalizációs csoportokról és egységekről tárol alapvető információkat:

- `LocID`: az őt tartalmazó fájl azonosítója
- `UnitID`: az egység fájlon belüli egyedi azonosítója

- `UnitType`: az egység típusa: egység, csoport, megjegyzés
- `ParentUnitID`: egység esetén az őt tartalmazó csoport azonosítója
- `State`: az adott egység lokalizációs állapotát jelöli: új, lefordított, lefordítandó, törölt

Az egységek ezen tulajdonságai a honosítás folyamata során nem, vagy csak ritkán változik.

A `Project_files_data` a lokalizációs csoportok és egységek tartalmát tárolja:

- `LocID`: az őt tartalmazó fájl azonosítója
- `UnitID`: az egység fájlon belüli egyedi azonosítója
- `UnitData`: az egység szöveges tartalma
- `Comment`: egységhez rendelt rövid megjegyzés (egyenlőre nem használt)

Az egységek tartalma a honosítás során többször is megváltozhat.

A tulajdonságok és tartalmak különválasztásának célja az volt, hogy az adatbázistábla méretét csökkentsem, illetve a `Project_files_data` tábla szerkezete – egy „`verziószám`” mezővel kibővítve - alkalmas arra, hogy fordítási előzményeket tároljon.

A táblákat az alábbi SQL utasítással hoztam létre:

```
CREATE TABLE `project_files_data_info` (
  `LocID`         int(11)          NOT NULL default '0',
  `UnitID`        varchar(30)      NOT NULL default '',
  `UnitFileID`    varchar(50)      NOT NULL,
  `UnitType`      smallint(6)      default NULL,
  `ParentUnitID`  int(11)          NOT NULL,
  `State`         smallint(6)      NOT NULL,
  PRIMARY KEY  (`LocID`,`UnitID`),
  CONSTRAINT `FK_project_files_data_info_1` FOREIGN KEY (`LocID`)
                              REFERENCES `project_files_loc` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;


CREATE TABLE `project_files_data` (
  `LocID`         int(11)          NOT NULL default '0',
  `UnitID`        varchar(30)      NOT NULL default '',
  `UnitData`      text,
  `Comment`       varchar(255)     default NULL,
  PRIMARY KEY  (`LocID`,`UnitID`),
```

```
  CONSTRAINT `FK_project_files_data_1` FOREIGN KEY (`LocID`)
                                REFERENCES `project_files_loc` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;
```

2.2.7. Tárolt eljárások

A fejlesztés során az 1.4. menüpontban megismertetett forgatókönyvek gyors és precíz végrehajtása céljából számos tárolt eljárást hoztam létre. Ezek közül a legfontosabbak:

ChangeContentLoc

Egy adott lokalizációs fájl egyik lokalizációs egységének a tartalmát változtatja meg.

A korábbi tartalmat eltárolja a `project_files_data_history` táblában, majd beállítja a `project_files_data` tábla megfelelő sorában az új szöveget és a `project_files_data_info` tábla adott sorában „lokalizált” állapotúvá állítja az adott egységet.

A `Projects` osztály `changeContent_loc` metódusa hívja meg.

Formális paraméterlistája:

Paraméter neve:	Típusa:	Leírás:
locID	INT	lokalizált fájl azonosító
unitID	INT	a fájl egy egységének azonosítója
newContent	TEXT	az egység új értéke

ChangeContentRef

Egy adott referencia fájl egyik egységének a tartalmát változtatja meg.

Először meghívja a *ChangeContentLoc* tárolt eljárást, hiszen a tartalom megváltoztatásának forgatókönyve megegyezik a lokalizált fájléval. Ezt követően a `project_files_data_info` táblában a referencia fájlhoz tartozó minden lokalizált fájl sorában beállítja a lokalizált fájlok azon sorának állapotát “nem lokalizált”-ra.

A `Projects` osztály `changeContent_ref` metódusa hívja meg.

Formális paraméterlistája megegyezik a *ChangeContentLoc* formális paraméterlistájával.

UploadNewRef

A paraméterben megadott adatok alapján a `project_files_data_info` és `project_files_data` táblákat új sorokkal bővíti.

A `Projects` osztály `upload_newref` metódusa hívja meg.

Formális paraméterlistája:

Paraméter neve:	Típusa:	Leírás:
refID	INT	referencia fájl azonosító
unitIDs	LONGTEXT	az egyes egységek azonosítójának listája
unitTypes	LONGTEXT	az egyes egységek típusának listája
pUnitIDs	LONGTEXT	az egyes egységek szülőjének azonosítói
unitDatas	LONGTEXT	az egyes egységek szöveges tartalmai

Az aktuális paraméterlista egy lokalizációs fájl elemzése után állítódik össze.

UploadRefUpdate

A paraméterben átadott adatok alapján a `project_files_data_info` és `project_files_data` táblák adott referencia fájl azonosítójú soraiban változtatásokat eszközöl:

- ha egy egység már létezik a táblákban, akkor meg kell változtatni a tartalmát, ezért meghívja a *ChangeContentRef* tárolt eljárást.
- ha egy egység még nincs a táblákban, akkor új tartalomként kell kezelni mind az adott referenciafájlban mind a hozzá tartozó összes lokalizált fájlban: új sorokkal bővíti a táblákat.
- minden olyan egységet, ami a táblákban létezik, de az aktuális paraméterként megkapott adatokban nincs benne, törli a táblákból.

A `Projects` osztály `upload_refUpdate` metódusa hívja meg.

Formális paraméterlistája megegyezik a *UploadNewRef* formális paraméterlistájával.

CreateNewLoc

Egy adott referencia fájlhoz hoz létre új nyelvű lokalizált fájlt.

Létrehozáskor az új fájl tartalma meg fog egyezni a referenciafájl aktuális tartalmával és minden lokalizációs egysége „nem lokalizált” állapotú lesz.

A `Projects` osztály `create_newloc` metódusa hívja meg.

Formális paraméterlistája:

Paraméter neve:	Típusa:	Leírás:
refID	INT	referencia fájl azonosító
newLangID	INT	az új nyelv azonosítója a `languages` táblában
userID	INT	a létrehozó felhasználó azonosítója

SetLocProgress

Egy adott lokalizált fájl jóságát állítja be az alapján, hogy a `project_files_data_info` táblában a hozzá tartozó egységek hány százalékának állapota „lokalizált”.

Formális paraméterlistája:

Paraméter neve:	Típusa:	Leírás:
locID	INT	lokalizált fájl azonosító

III. Lezárás, tanulságok

Az alkalmazás fejlesztése kezdetén egyértelművé vált, hogy PHP alapokon fog megvalósulni MySQL adatbázist használva. Először a MySQL 4-es verzióját használtam, viszont amikor a tárolt eljárások alkalmazása szóba került, nem volt kérdés, hogy váltani kell az 5-ös verzióra. Továbbá a jövőbeli tervek között szerepel az MSSQL-re való áttérés, amely sokkal több funkcióval bír az ingyenes MySQL-hez képest.

Fejlesztés közben témavezetőm, Ákos és más külső felhasználók tesztjei felbecsülhetetlen értékűek voltak, és már korai stádiumban rámutattak a terveket jól mutató, de gyakorlatban nehézkésnek bizonyuló megoldásokra, hiányosságokra.

Kezdetben egy olyan webes alkalmazás kifejlesztése volt a cél, amely referencia fájlokat kezel, hozzájuk más nyelvű változatokat lehet rendelni, és a feltöltött fájlok tartalmát meg tudja változtatni. Az első elkészült változatban egy konkrét fájlt csak egy felhasználó tudott egyszerre módosítani. Ahogy a szerkesztés elkezdődött, a rendszer zárolta a fájlt, és csak mentéskor oldódott fel. Viszont a gyakorlati használat során előjöttek ennek a megoldásnak a hátrányai: a nehezen kezelhetőség, és az, hogy a fájlok fordítása gördülékenyebben történjen.
Ezért bevezettük a fájlok konkurens hozzáférhetőségét, amely figyelembe véve mind a programlogikát, a használhatóságot és a gyorsaságot tökéletes megoldásnak bizonyult. A felhasználók így már egymástól függetlenül egy időben tudnak egy fájlon párhuzamosan dolgozni.

Bár a fejlesztés első szakasza a tervezés volt, a fejlesztés különböző fázisaiban szükség volt nagyobb tesztekre, amelyek fényt derítettek az esetleges hibás tervezési lépésekre, ezzel együtt új igények merültek fel, és emiatt a terveket módosítani kellett. Menet közben merült fel az igény arra, hogy az alkalmazás minél szélesebb körben testre szabható, skálázható legyen.

A felhasználói felület folyamatos változás alatt állt. Mindig kerültek be új funkciót, tűntek el régi megoldások, így vált fokozatosan minél inkább felhasználóbaráttá.
A lokalizációs egységek megjelenítése a szerkesztési lapon például többször változott.

Kezdetben egy sornyi szövegbeviteli mező állt rendelkezésre, de hamar látszott, hogy ez nem megfelelő. Később a szerkesztést megkönnyítendő opciókat és jelöléseket vezettünk be, amelyek növelik az átláthatóságot, könnyítik a fordítást. Lehetőség lett megjegyzések írására, és később kerül bevezetésre külső, ingyenes fordítóeszközök használatával az egyes szövegekhez fordítási tippek adása.

A mélyebb tesztekből derült ki a rendszer megfelelő védelmének a hiánya. A nem megfelelő szerkezetű fájlok feltöltése és különböző szkript és SQL támadások hibákat, nem megfelelő működést okoztak. Ezek javítása, illetve a későbbiekben előfordulható ehhez hasonló hibák elkerülése kulcsfontosságú volt, a fejlesztés során állandóan szem előtt kellett tartani.

A fejlesztéssel párhuzamos folyamatos tesztelés eredményeképpen remélem sikerült olyan alkalmazást írnom, amely az alkalmazás-fejlesztők, a szerkesztők és a honosításokat kereső felhasználók számára is átlátható és a mindennapi gyakorlatban is könnyen és gyorsan használható eszközzé válik.

Irodalomjegyzék

[1] Wikipedia: Internationalization and localization
http://en.wikipedia.org/wiki/Internationalization_and_localization

[2] Wikipedia: AJAX (programming)
http://en.wikipedia.org/wiki/AJAX

[3] PHP online dokumentáció
http://php.net

[4] MySQL online dokumentáció
http://dev.mysql.com/doc/