**Rendszerterv**

Ebben a dokumentumban részletesen összegezni fogjuk, hogy az általunk választott projekt hogyan, milyen alkotóelemekből fog felépülni, ezen elemek között milyen kapcsolatok vannak, illetve milyen technológiákat kívánunk alkalmazni.

*A rendszer architektúrája:*

A következőkben bemutatjuk, hogy milyen architektúrát használunk fel a projektünkben, illetve bemutatjuk az egyes komponenseket.

A projektünk egy regisztrációs és beléptető rendszert valósít meg, melyben a már regisztrált felhasználóknak lehetőségük lesz a saját fiókjukba való belépésre, profiljuk személyre szabására/adataik módosítására.

A rendszer első alkotóeleme egy webes felhasználói felület lesz. Ezen a felületen keresztül lesz lehetőségük a felhasználóknak a rendszer által nyújtott egyes funkciókat igénybe venni, navigálni.

A második fő komponens a funkciókat tényleges megvalósító rész, a back-end. Ezen rész a következő alkomponensekből épül fel:

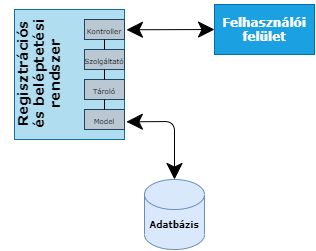
* kontroller réteg
* szolgáltatói réteg
* tároló (repository) réteg
* model réteg.

A felhasználói felülettel a kontroller réteg áll közvetlen kapcsolatban.

A harmadik komponens az adatbázis réteg, ahol a felhasználó személyes adatai, illetve a rendszer futása során, a rendszer helyes működéséhez szükséges egyéb információk tárolódnak. Ez a réteg a back-end komponenshez a model rétegen keresztül kapcsolódik: a model réteg az adatbázis kódbeli reprezentációja.

A rendszeren belül a kommunikáció a következőképpen zajlik:

A felhasználói felületen végzett műveleteket a kontroller réteg kezeli. A kontroller réteg feladata, hogy az utasításokat továbbítsa a megfelelő szolgáltatásnak. A szolgáltatói rétegben található a rendszer üzleti logikája. A szolgáltatások a repository rétegen keresztül lekérdezéseket intéznek a model réteghez. A model réteg visszaküldi a lekérdezett adatot a kontroller számára, ami pedig ezt visszajuttatja a felhasználói felületnek.

Az architektúra felépítése:

*Általunk választott technológiák:*

A projekt front-end részének megvalósítása a React könyvtár segítségével történik meg.

A back-end rész megvalósítása Java nyelven, valamint a Spring keretrendszer felhasználásával történik.

Ez a keretrendszer rengeteg olyan eszközzel rendelkezik, amelyek nagyban megkönnyítik webes alkalmazások készítését, valamint segítségével biztonságosabb alkalmazásokat lehet készíteni, illetve felhasználhatjuk annak kiváló adatbázisokkal való támogatottságát.

A választásunk az adatbázis terén a MySQL-re esett, ugyanis ez különösen népszerű és alkalmas megoldás webalkalmazások készítése esetén, annak biztonságossága, illetve jó skálázhatósága miatt.

*A rendszert felépítő modulok:*

* regisztrációs modul
* felhasználókat kezelő modul
* megerősítésért felelős modul
* email-eket kezelő modul
* biztonsági modul

A regisztrációs modul az egyik központi modul a fentiek közül: ez kezeli a felhasználói felület felől érkező regisztrációs kéréseket. A regisztrációhoz szükséges visszaigazoló email-ben küldött link aktiválása, így ez a modul fel fogja használni a megerősítésért, és az email-ek küldéséért felelős modulokat is. Sikeres regisztráció esetén az egyes felhasználókhoz jogokat is rendelünk, így ez a modul kapcsolatban áll felhasználókat kezelő modullal is.

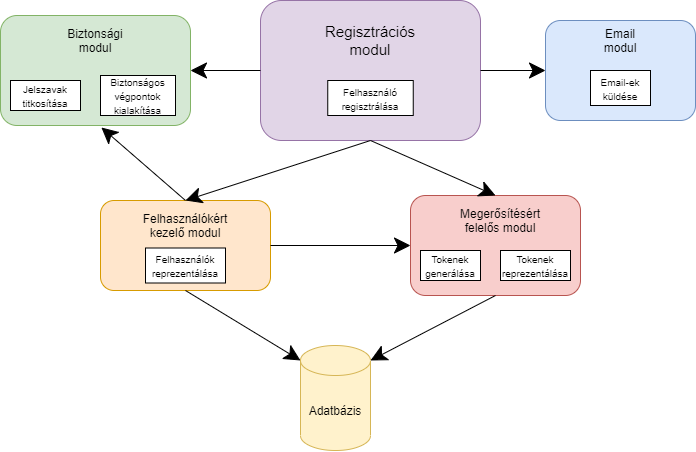
A megerősítésért felelős modul ún. tokeneket (karaktersorozatokat) generál, amelyek a felhasználó regisztrációját azonosítják. Ez a modul kapcsolatban áll az adatbázissal is, ugyanis a tokenek és azok egyéb adatai (pl. létrehozás/lejárat ideje) adattáblákban is eltárolódnak.

A felhasználókat kezelő modulban vannak reprezentálva az egyes felhasználók, emiatt ez a modul kapcsolatban áll az adatbázissal is. Továbbá a biztonsági modullal is kapcsolatban áll, mivel a felhasználók jelszavait titkosítva tároljuk el az adatbázisban, mely titkosítási folyamat a biztonsági modulban történik. Illetve az egyes felhasználók esetén számontartjuk, hogy az adott fiók megerősítésre került-e, ebből következően a megerősítésért felelős modullal is összeköttetésben áll.

Az email-eket kezelő modul nevéből értetődően konkrétan az email-ek küldéséért felelős.

A biztonsági modulnak fontos szerepe van, ugyanis itt kerül megvalósításra a jelszavak titkosítása, valamint itt vannak definiálva az általunk biztonságosnak ítélt végpontok.

Összefoglaló ábra a modulokról:

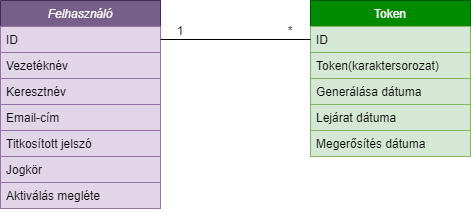


*Perzisztencia:*

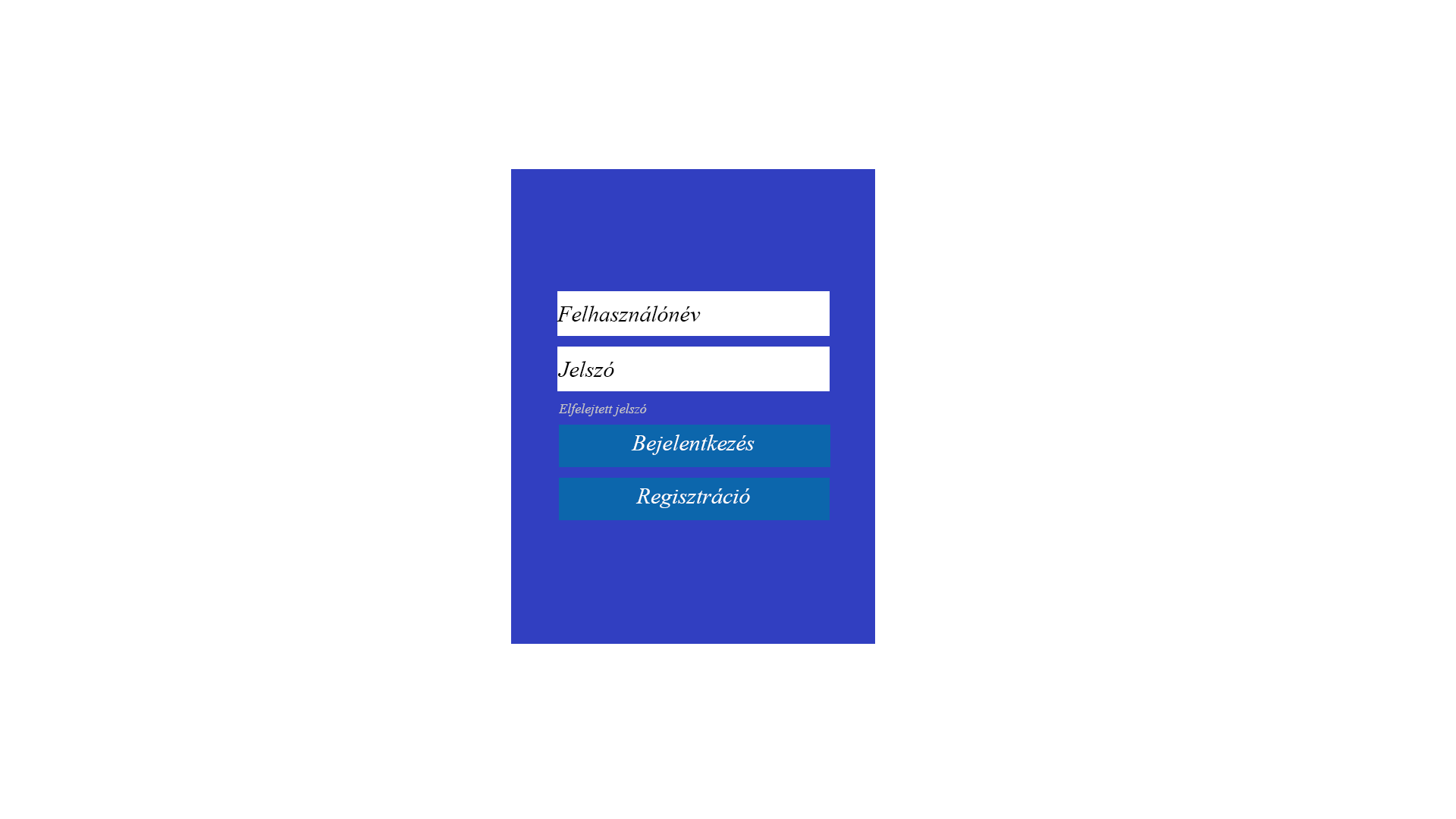
Mint ahogyan azt már korábban említettük, a perzisztencia megvalósítására egy relációs adatbáziskezelő rendszert, a MySQL-t választottuk. A program futása során két, egymástól elkülöníthető táblát fogunk felhasználni az adatbázison belül:

* az egyik táblában a felhasználó személyes adatait (név, email-cím, jelszó titkosított formában stb.) tároljuk el;
* a másik táblában a megerősítés során (fiók aktiválása, illetve a kétfaktoros autentikáció során) használatos tokeneket, és azok adatait (generálás dátuma, lejárati dátum, megerősítés dátuma) tároljuk el.

A két tábla kiegészül ezeken felül egy ID/azonosító mezővel is, amely a két tábla közti kapcsolatot szolgálja. Egy felhasználóhoz tetszőleges számú token tartozhat, viszont egy adott token pontosan egy felhasználóhoz tartozik egyszerre.

Az adatbázis felépítése:

*Használt biztonsági mechanizmusok:*

A rendszer back-end része Java Spring keretrendszer segítségével van megvalósítva, ezáltal lehetőségünk van a Spring Security által kínált funkciókat igénybe venni. Ez konkrétabban azt jelenti, hogy a felhasználók jelszavait Spring által biztosított titkosítási algoritmusokkal kódoljuk, amelyek így kerülnek tárolásra. Továbbá a Spring biztosítja a megbízható végpontok kialakítását is: ez azt jelenti, hogy korlátozva lesznek a nyilvánosan elérhető címek (URL-ek), amelyeken keresztül elérhetjük a kívánt erőforrást.

Concept art a programról: