Java alkalmazások

Tervdokumentáció

Gyakorlat

|  |  |
| --- | --- |
| **Készítették** | **Feladatkör** |
| Szabó Ferenc | Frontend, Login, Regisztráció, GitHub |
| Nagy Erik | Lekérdezés, Kapcsolat, Üzenetek, RESTful API |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Dokumentum azonosító: | L-K-GINFBAL-JAVALKA-1-LA01 |
| Verzió: | 1.1 |
| Dátum: | 2024. 11. 29. |

|  |
| --- |
| **Lektorálta** |
| Nagy Erik |

**©** Ferinek, nem tetszett a copyright jogi üzenet ezért valamit raktam ide, hogy szép legyen a dokumentáció.

Kellemes ebédszünetet, húsvétot, karácsonyt, nyaralást, és egyéb dolgokat kíván a „Csábító Tea Party”.

Tartalomjegyzék

[**1.** **Bevezetés** 4](#_Toc183782158)

[**2.** **Fejlesztési környezet** 4](#_Toc183782159)

[*2.1* *A fejlesztői környezet: IntelliJ IDEA és Spring boot* 4](#_Toc183782160)

[*2.2* *Verziókövetés: GitHub* 5](#_Toc183782161)

[*2.3* *Használt technológiák* 6](#_Toc183782162)

[**3** **Design** 7](#_Toc183782163)

[*3.1* *Általános kinézet és működés* 7](#_Toc183782164)

[3.1.1 Az oldal fejléce: 7](#_Toc183782165)

[3.1.2 Főoldal: 7](#_Toc183782166)

[3.1.3 Kapcsolatok: 8](#_Toc183782167)

[3.1.4 Üzenetek: 8](#_Toc183782168)

[3.1.5 Lekérdezés: 9](#_Toc183782169)

[3.1.6 Bejelentkezés 9](#_Toc183782170)

[3.1.7 Regisztráció 9](#_Toc183782171)

[3.1.8 Az oldal lábléce: 10](#_Toc183782172)

[*3.2* *Reszponzív megjelenés* 11](#_Toc183782173)

[**4.** **Fejlesztési keret: Frontend, Backend és Adatkezelés** 12](#_Toc183782174)

[*4.1* *Frontend* 12](#_Toc183782175)

[*4.2* *Backend* 13](#_Toc183782176)

[*4.3* *Adatkezelés* 27](#_Toc183782177)

[*4.4* *Tesztek és kiegészítő információk* 28](#_Toc183782178)

# **Bevezetés**

Jelen tervdokumentáció a „Java alkalmazások” elnevezésű tantárgy, a **Csábító Tea Party** csapataáltal fejlesztés alatt álló számítástechnikai projekt, **weboldal** fejlesztői környezetét, koncepcióját, design elemeinek tervezett megvalósítását és fejlesztési tevékenységeit ismerteti.

# **Fejlesztési környezet**

## *A fejlesztői környezet: IntelliJ IDEA és Spring boot*

Az **IntelliJ IDEA** egy professzionális és népszerű fejlesztői környezet (IDE), amelyet a JetBrains fejlesztett ki. Széles körben ismert a fejlett funkcióiról, mint például a kód intelligens kiegészítése, hibák automatikus felismerése, és a mély integrációja a modern programozási nyelvekkel és keretrendszerekkel. Az IntelliJ IDEA kiemelten támogatja a Java nyelvet és kapcsolódó technológiákat, beleértve a Spring Boot keretrendszert is. Ez az IDE különösen előnyös nagyobb projektekhez, mivel hatékony eszközkészletet biztosít a kód navigációhoz, refaktoráláshoz és hibakereséshez.

A **Spring Boot** egy modern, nyílt forráskódú Java keretrendszer, amely a Spring Framework kiterjesztése. A Spring Boot leegyszerűsíti a Java-alapú webes és mikroszolgáltatás-alkalmazások fejlesztését azáltal, hogy előre definiált konfigurációkat és beépített eszközöket kínál. A "konvenció a konfiguráció helyett" megközelítésének köszönhetően gyorsan beállítható és elindítható egy új alkalmazás. Az integrált eszközök, mint például a beépített Tomcat szerver és a Spring Data, lehetővé teszik a fejlesztők számára, hogy a fejlesztési folyamat során a vállalkozási logikára koncentráljanak, miközben a keretrendszer kezeli az infrastruktúra többi részét.

A „Java alkalmazások” tantárgyhoz tartozó projekt fejlesztése során csapatunk az IntelliJ IDEA és a Spring Boot használatát részesíti előnyben. Ez a fejlesztői környezet egy hatékony és rugalmas alapot biztosít az alkalmazásunk gyors és megbízható elkészítéséhez. Az IntelliJ IDEA által nyújtott fejlett fejlesztési funkciók, valamint a Spring Boot könnyű konfigurálhatósága és robusztussága lehetővé teszik számunkra, hogy hatékonyan kezeljük a projekt követelményeit. Az eszközök kombinációja zökkenőmentes munkafolyamatot biztosít, lehetővé téve a csapat számára, hogy a projekt technikai kihívásaira összpontosítson, miközben minimalizálja a fejlesztési környezet karbantartásával járó terheket.

## *Verziókövetés: GitHub*

A **GitHub**, mint vezető szoftverfejlesztési platform, lehetővé teszi a csapat számára, hogy hatékonyan kezelje a kódbázist, nyomon követhesse a változásokat, és elősegítse a kollaboratív munkát azáltal, hogy egy központosított helyen tárolja a projekt fájljait és dokumentációját. Az integrált verziókezelési rendszer előnyei közé tartozik a fejlesztési folyamatok nagyfokú automatizálása, a visszaállíthatóság, valamint a különböző fejlesztési ágak (branches) kezelése, ami lehetővé teszi a csapat számára, hogy párhuzamosan dolgozzon különböző funkciókon. A GitHub használatával a Csábító Tea Party csapata kihasználhatja a kódmegosztás és a közösségi hozzájárulás előnyeit is, így gyorsítva a fejlesztési ciklusokat és javítva a projekt átláthatóságát és nyomon követhetőségét.



## *Használt technológiák*

A Csábító Tea Party csapata a webfejlesztés során számos modern és bevált technológiát alkalmaz, hogy biztosítsák a webalkalmazásuk gyors, biztonságos és felhasználóbarát működését:

**HTML (HyperText Markup Language)**: Az internet alapját képező nyelv, amely a weboldalak struktúráját adja meg. Az HTML5 a legújabb verziója, amely további szemantikai elemeket és jobb multimedia támogatást kínál, ezáltal elősegíti a reszponzív webdesign kialakítását.

**CSS (Cascading Style Sheets):** A webes megjelenítésért felelős stíluslapokat biztosítja. A CSS3 továbbfejlesztett funkciói, például az animációk és átmenetek, esztétikailag vonzó és modern felhasználói élményt tesznek lehetővé.

**JavaScript:** A kliensoldali működésért felelős szkriptnyelv, amely lehetővé teszi az interaktivitást, például az űrlapok validációját vagy dinamikus tartalom betöltését. Továbbá, keretrendszerek, mint a **Vue.js** vagy **React**, segítenek még dinamikusabb felhasználói felület létrehozásában.

**Spring Boot:** Az alkalmazás backend-jének fejlesztéséhez használt keretrendszer. A Spring Boot leegyszerűsíti a webes és mikroszolgáltatás-alapú alkalmazások létrehozását, és támogatja az automatikus konfigurációt. Robusztus megoldásokat biztosít adatbázis-kezelés, REST API-k és munkamenetek kezelésére.

**Hibernate:** Egy népszerű objektum-relációs leképezési (ORM) eszköz, amely lehetővé teszi, hogy a Java osztályok közvetlenül kapcsolódjanak az adatbázis tábláihoz. Ez csökkenti az adatbázis-műveletek kódolási komplexitását.

**MySQL:** Nyílt forráskódú relációs adatbázis-kezelő rendszer, amely hatékony adattárolási megoldásokat nyújt. A Spring Data JPA segítségével könnyen integrálható a Java alapú alkalmazásokba.

**Thymeleaf:** Egy modern sablonmotor a HTML oldalak dinamikus generálásához. A Spring Boot-tal szoros integrációban működik, így egyszerűvé teszi az adatok megjelenítését és a felhasználói felülettel való kommunikációt.

**Maven:** Az alkalmazás építési folyamatainak kezelésére használt eszköz, amely automatikusan kezeli a függőségeket, és biztosítja a projekt struktúrájának tisztaságát és karbantarthatóságát.

**Reszponzív webdesign:** Az alkalmazás kialakításakor külön figyelmet fordítunk arra, hogy minden eszközön, beleértve a mobiltelefonokat és tableteket is, felhasználóbarát és esztétikus megjelenésű legyen. A reszponzivitás eléréséhez CSS Grid-et, Flexbox-ot és modern JavaScript eszközöket használunk.

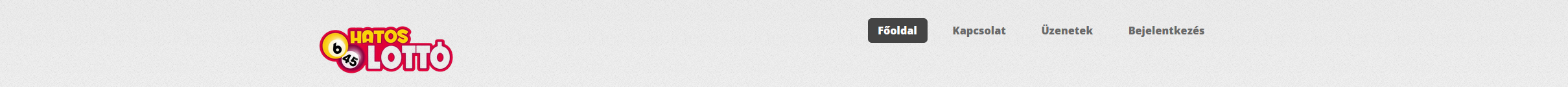
Ezek a technológiák együttesen biztosítják a fejlesztési folyamat hatékonyságát, az alkalmazás megbízhatóságát, valamint a felhasználói élmény magas színvonalát.

# **Design**

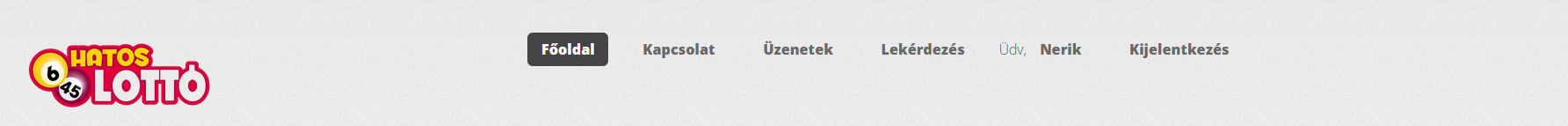
## *Általános kinézet és működés*

A weboldal kinézetének és működésének kialakítása során egy ingyenesen elérhető sablont használtunk, amelyet a Hatoslottó adatbázisának megjelenítéséhez és kezeléséhez igazítottunk. A sablon egyszerű és letisztult dizájnja lehetővé tette számunkra, hogy gyorsan és hatékonyan alakítsuk ki az oldal struktúráját. A weboldalt a lottó témához illeszkedő képekkel és színekkel tettük egyedivé, így az vizuálisan is harmonizál az oldal tartalmával. Ezek a változtatások nemcsak esztétikailag illeszkednek a Hatoslottó témájához, hanem a felhasználók számára egyértelmű, könnyen navigálható élményt is nyújtanak.

### 3.1.1 Az oldal fejléce:

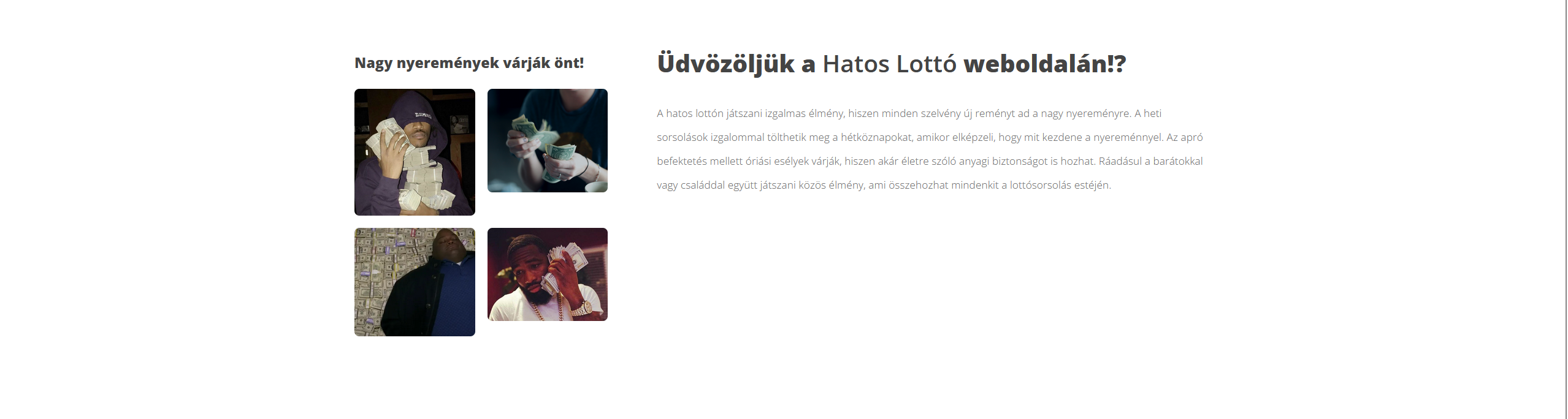
****

Az oldal fejlécében kerültek elhelyezésre a kattintható menüpontok, illetve egy logo.



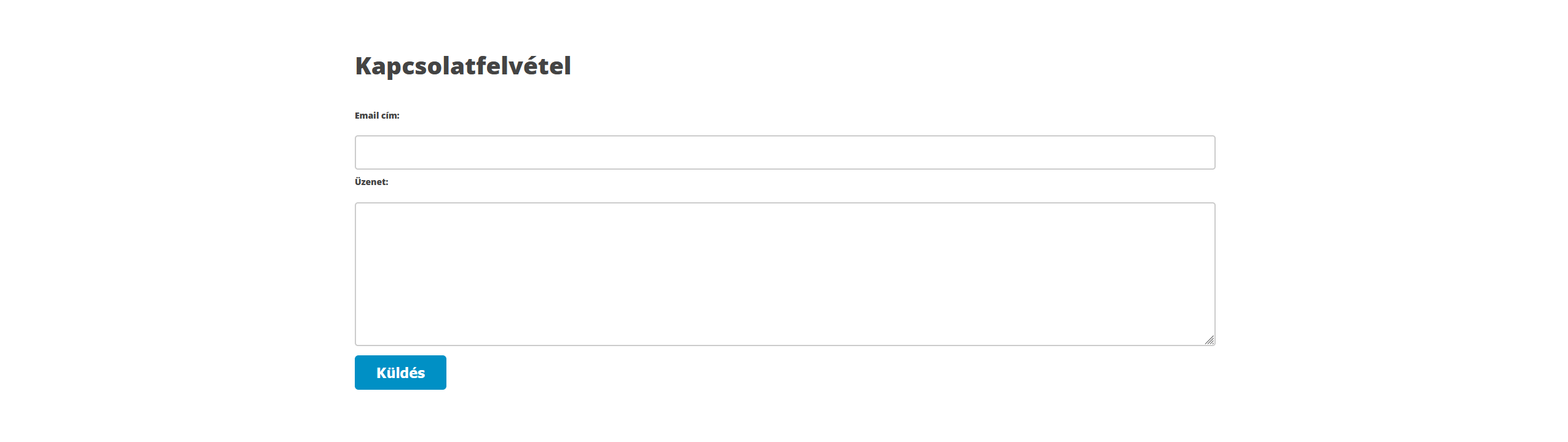
A fejléc némileg megváltozik bejelentkezést követően.

### 3.1.2 Főoldal:

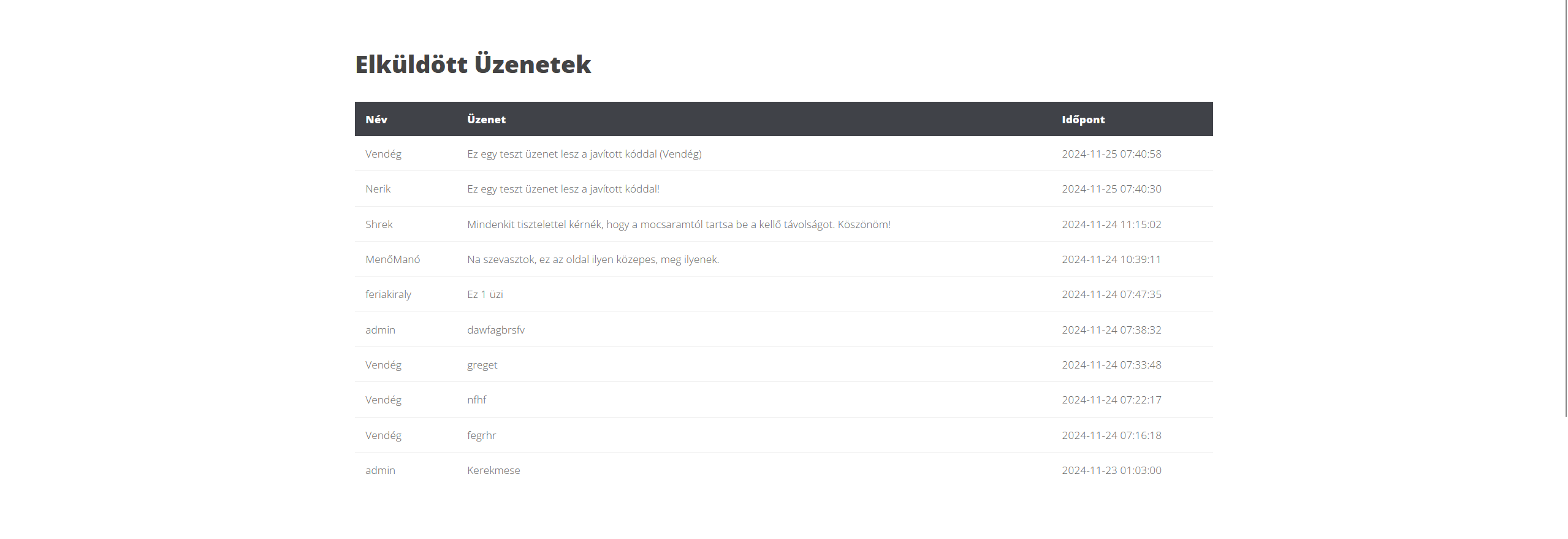
****

Az oldal főoldalán egy általános üdvözlő üzenet helyezkedik el, pár csalogató képpel együtt.

### 3.1.3 Kapcsolatok:

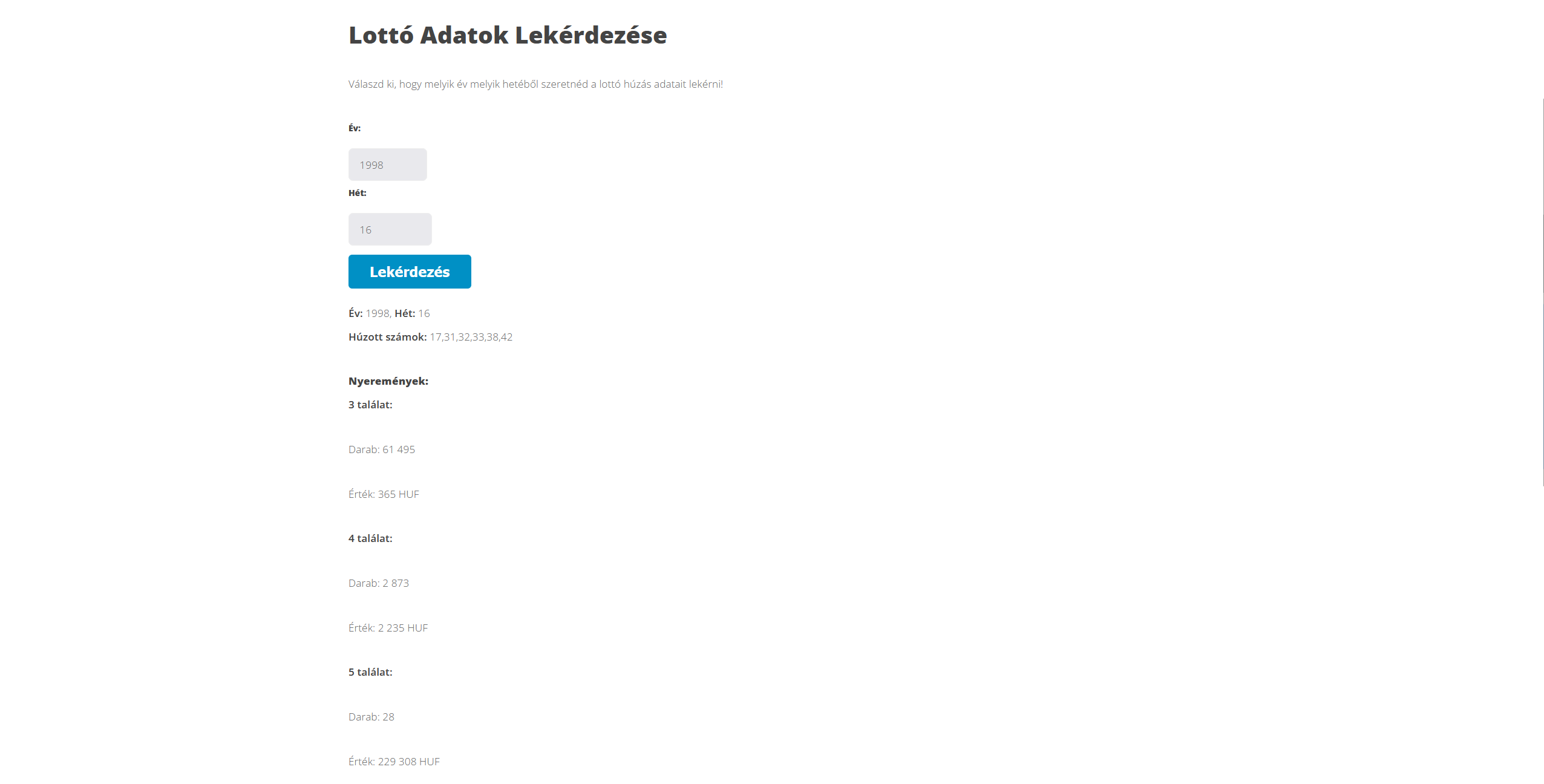
  
A Kapcsolat menüpontban van lehetőség az oldal üzemeltetőinek üzenetet írni, akár Vendég, akár bejelentkezett felhasználóként. Ha ezt bejelentkezett felhasználóval vesszük igénybe, úgy email cím megadása nem szűkséges.

### 3.1.4 Üzenetek:



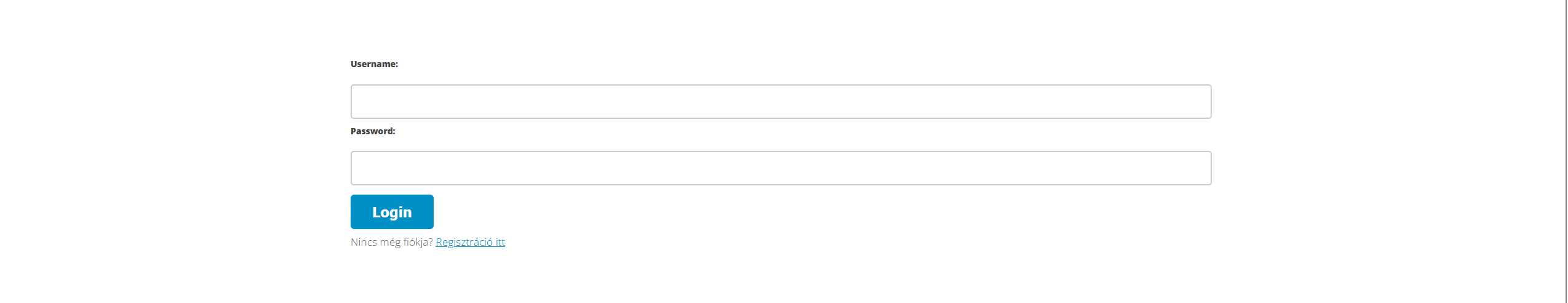
Az üzenetek menüpont alatt lehet megtekinteni azokat az üzeneteket amik a „Kapcsolatok” menüponttal lettek elküldve. Amennyiben olyan felhasználó küldte aki nem volt regisztrálva az oldalra, akkor „Vendég” jelenik meg. Amennyiben regisztrált felhasználó küldte ezt az üzenete, úgy a felhasználó név jelenik meg.

### 3.1.5 Lekérdezés:



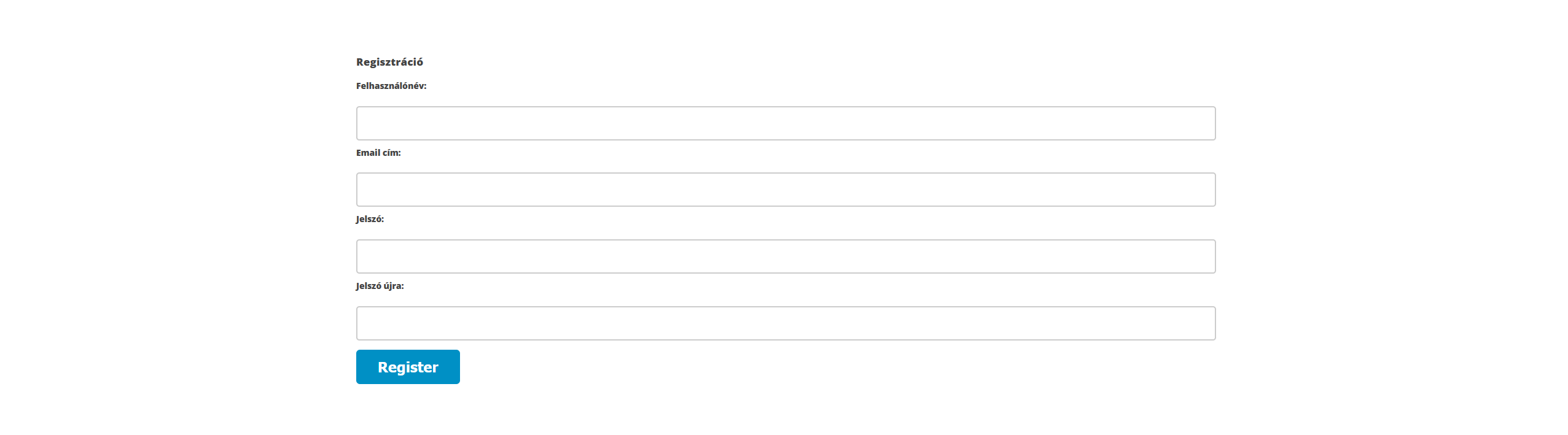
A lekérdezés menüpont egy rejtett menü, ami csak „admin” felhasználónak jelenik meg. Itt lehet lekérdezni a feladathoz kapcsolódó adatbázisból a lottó számok választott év, választott hetére. Kizárólag azokat az éveket és heteket tudjuk választani, amik léteznek az adatbázisban. Amennyiben volt Nyeremény azon a héten, úgy listázásra kerülnek a találatok és nyeremény összegek, amennyiben pedig nem, úgy a „Nincs nyeremény adat” felirat jelenik meg.

### 3.1.6 Bejelentkezés:



A bejelentkezés menüpontra kattintva jelentkezhetünk be az előzetesen regisztrált felhasználó fiókunkra, illetve itt van lehetőségünk új fiókot létrehozni is, amennyiben még nem rendelkezünk vele.

### 3.1.7 Regisztráció:



A regisztrációs oldalon az adatok kitöltését követően, létrehozhatjuk felhasználói fiókunkat.

### 3.1.8 Az oldal lábléce:



Az oldal láblécében található pár hasznos link ami egyéb projektetekre, avagy a 6-os lottó hivatalos közösségi felületeire mutat.

## *Reszponzív megjelenés*

A reszponzív megjelenés kialakításához a projekt során a Spring Boot által támogatott modern technológiákat és egyedi CSS-t alkalmaztunk. A sablon alapú megközelítés lehetővé tette, hogy az oldal elrendezése könnyen adaptálható legyen különböző képernyőméretekhez, legyen szó asztali számítógépekről, tabletekről vagy mobiltelefonokról. A CSS segítségével az oldal stílusát teljes mértékben testre szabtuk, beleértve a színek, betűtípusok és margók finomhangolását. A reszponzivitás érdekében rugalmas elrendezést alakítottunk ki, amely biztosítja a tartalom átláthatóságát és könnyű kezelhetőségét minden eszközön. A Spring Boot integrációjával egyszerűen valósítottuk meg a dinamikusan generált tartalmak adaptív megjelenítését, így a felhasználók minden környezetben esztétikus és funkcionális élményben részesülhetnek.

# **Fejlesztési keret: Frontend, Backend és Adatkezelés**

## *Frontend*

**Feladata:** A weboldal elsődleges célja és feladata, hogy bemutassa a projektmunkánkat, valamint információkat nyújtson a feladathoz kiadott adatbázisból lekérdezhető adatokról.

**Felépítése:** A weboldal struktúrája a Spring Boot által támogatott sablonrendszeren és egyedi CSS megoldásokon alapul, amelyek lehetővé teszik az oldalak dinamikus és reszponzív elrendezését. Az oldalak kialakításánál rugalmas elrendezést alkalmaztunk, amely a tartalmak és a vizuális elemek hatékony kezelését biztosítja különböző képernyőméreteken. A dizájn elemek és az oldal specifikus igényeinek megfelelő megjelenés az egyedi CSS fájlok segítségével valósultak meg.

**Működése:** Az oldal működését a felhasználói interakciók és a vizuális elemek harmonikus összehangolására terveztük, különös figyelmet fordítva a felhasználói felület intuitív és felhasználóbarát kialakítására. A cél az volt, hogy a látogatók könnyen tájékozódjanak, miközben az oldal megjelenése esztétikailag is vonzó maradjon. Az egyedi stílusjegyekkel gazdagított dizájn célja, hogy egyedi vizuális élményt nyújtson, amely hosszabb időtöltésre és interakcióra ösztönzi a látogatókat.

Az oldalt úgy alakítottuk ki, hogy különböző képernyőméretekhez kiválóan alkalmazkodjon, a Spring Boot sablonmotorja és az egyedi CSS megoldások révén. Mobiltelefonokon és kisebb képernyőkön a navigációs menü egyszerű, kattintható ikonná zsugorodik, amelyet a felhasználók könnyedén kibonthatnak. Az oldal elemei – beleértve a szövegeket és képeket – rugalmasan egymás alá rendeződnek, biztosítva a könnyű olvashatóságot és egyszerű navigációt.

A képek mérete automatikusan igazodik a képernyőhöz, így mindig optimális megjelenést biztosítanak, miközben megőrzik a vizuális minőséget. Ez a reszponzív működés garantálja, hogy az oldal minden eszközön informatív, felhasználóbarát és esztétikus maradjon.

## *Backend*

**Feladata:** A backend a weboldal szerveroldali logikájának kivitelezéséért felelős, és a felhasználók számára közvetlenül nem látható. Ez a réteg felel az adatkezelésért, az üzleti logika megvalósításáért és a kliensoldali kérések feldolgozásáért. A projekt során a **Spring Boot** keretrendszert használtuk, amely a Java nyelvre épül. Ez a technológia lehetővé teszi a robusztus és bővíthető szerveroldali alkalmazások fejlesztését, modern objektumorientált programozási megközelítéssel.

A Spring Boot egyszerűsíti az alkalmazás konfigurálását és működtetését, automatikusan kezeli a függőségeket és a beépített szolgáltatásokat, például a REST API-k kezelését. Az adatbázissal való kommunikációhoz a Hibernate ORM-et és a Spring Data JPA-t használtuk, amelyek hatékonyan biztosítják a relációs adatbázisok kezelését és a Java objektumokkal való integrációt. Ez a megközelítés nemcsak megbízható adatkezelést tesz lehetővé, hanem hozzájárul a kód átláthatóságához és újrafelhasználhatóságához is.

**Felépítése:** A backend rendszerünk egy Spring Boot-alapú MVC (Model-View-Controller) architektúrára épül, amely tiszta szerkezetet és logikus rétegződést biztosít a kód számára. A **FrontendController** a központi vezérlő elem, amely a kliens oldali kéréseket fogadja, feldolgozza, és megfelelő modellekkel és nézetekkel dolgozik együtt a válasz előállításához.

**Működése:** A működés központjában a ***FrontendController*** áll, amely a kérések útválasztásáért felelős.

***FrontendController:***

* **Kérések fogadása:** Az oldal URL-jeire érkező kérések a megfelelő végpontokhoz irányulnak (pl. /contact, /messages).
* **Adatok feldolgozása:** A vezérlő a szolgáltatásrétegtől kapott adatokat feldolgozza és továbbítja a nézetnek.
* **Nézet generálása:** A nézetek HTML sablonok formájában készülnek, amelyekhez a vezérlő átadja a szükséges adatokat.
* **Válasz visszaküldése:** A generált HTML oldalak a felhasználók böngészőjében jelennek meg.

***HTML Sablonok:***

**Weboldal fő HTML sablonja (layout.html)**

Ez a sablon a weboldal alapvető struktúráját és elrendezését határozza meg, amelyet a Spring Boot és Thymeleaf segítségével dinamikusan töltünk meg tartalommal. A sablon reszponzív, modern megjelenést biztosít, és tartalmazza az oldal navigációját, láblécét, valamint a dinamikus tartalmi területet.

**Fejléc**

A ***<head>*** részben található:

* Az oldal dinamikus címe, amely a ${title} változóból kerül betöltésre.
* A metaadatok, beleértve a karakterkódolást és a reszponzivitást biztosító viewport beállítást.
* A main.css fájl hivatkozása, amely az oldal alapvető megjelenését szabályozza.
* A ${meta} blokkon keresztül további specifikus metaadatok helyezhetők el.

**Header (fejléc rész)**

A fejléc tartalmazza az oldal logóját és navigációs menüjét:

* A logó a hatoslotto.png képet használja, amely a főoldalra mutató linkként is szolgál.
* A navigációs menü dinamikusan jeleníti meg az opciókat:
  + A Főoldal, Kapcsolat és Üzenetek minden felhasználó számára elérhető.
  + A Lekérdezés menüpont csak adminisztrátori jogosultsággal látható a sec:authorize="hasAuthority('admin')" feltétel alapján.
  + Az anonim felhasználók a Bejelentkezés opciót látják, míg az autentikált felhasználók üdvözlést és kijelentkezési lehetőséget kapnak.

**Fő tartalom**

A fő tartalmi rész a <main> elemben található, amely dinamikusan cserélődik a ${content} blokk helyére. Ez a Thymeleaf segítségével integrálja az egyes oldalak specifikus tartalmát.

**Lábléc**

A lábléc az oldal kiegészítő információit tartalmazza:

* **Hasznos linkek:** Példák a lottóhoz kapcsolódó külső oldalakra.
* **Kapcsolat:** A közösségi média linkjei és a vállalat elérhetőségei.
* **Szerzői jog:** Alapértelmezett sablon a HTML5 UP dizájn hivatkozásával.

**Reszponzív design és reszponzív menü**

A sablon teljesen reszponzív, a CSS stílusok és a JavaScript segítségével minden képernyőméreten optimalizált megjelenést biztosít:

* A menü automatikusan adaptálódik kisebb eszközökhöz a dropotron.min.js és a main.js segítségével.
* A tartalmi elemek mérete és elrendezése igazodik a képernyőhöz, biztosítva az esztétikus és felhasználóbarát élményt.

**Dinamikus adatok és funkciók**

A sablon Thymeleaf attribútumokat használ a dinamikus tartalom megjelenítésére:

* A navigáció és a tartalom megjelenése a felhasználó jogosultságaitól és hitelesítési állapotától függ.
* Az adminisztrátori jogosultságok esetén további opciók jelennek meg, például a lekérdezések.

Ez a sablon biztosítja az oldal egységes megjelenését és működését, miközben lehetőséget ad az egyedi tartalmak dinamikus betöltésére a különböző végpontok alapján.

**Főoldali HTML sablon (index.html)**

Ez a sablon a Hatos Lottó weboldalának főoldalát határozza meg. Célja, hogy vizuálisan vonzó és informatív belépési pontot biztosítson a látogatók számára, miközben hangsúlyozza a játék élményét és lehetőségeit.

**Tartalom elrendezése**

Az oldal elrendezése két fő oszlopra van bontva, amelyeket a container és a row osztályok kezelnek.

* + **Oldalsáv:** A bal oldalon található, a col-4 col-12-medium osztályokkal definiálva.
    - Az oldalsáv egy thumbnails widgetet tartalmaz, amely vizuálisan vonzó képeket jelenít meg, például pénzzel kapcsolatos grafikákat, amelyek a lottózás nagy nyereményeire utalnak.
    - A képek reszponzív elrendezésűek, és az image fit osztály biztosítja, hogy a képek minden eszközön jól nézzenek ki.
  + **Fő tartalom:** A jobb oldalon található, a col-8 col-12-medium osztályokkal, amely az oldalsáv melletti fő területet foglalja el.
    - Ez a rész egy last szekcióval rendelkezik, amely egy üdvözlő szöveget és egy leírást tartalmaz a Hatos Lottó izgalmas játékélményéről.
    - A szöveg célja, hogy ösztönözze a látogatókat a lottózásra, miközben kiemeli a játék közösségi és anyagi előnyeit.

**Reszponzivitás**

* Az elrendezés a gtr-200 és gtr-50 osztályok segítségével alkalmazkodik a különböző képernyőméretekhez.
* Az oszlopok (oldalsáv és fő tartalom) kisebb képernyőkön egymás alá rendeződnek, biztosítva a könnyű olvashatóságot és navigációt.

**Funkcionalitás és vizuális elemek**

* A vizuális elemek, mint például a pénz témájú képek, az oldal hangulatát és témáját erősítik, miközben figyelemfelkeltő hatást keltenek.
* A központi tartalom egy barátságos és ösztönző üdvözlő üzenetet jelenít meg, amely az oldalt látogatók számára egyértelművé teszi, hogy a Hatos Lottó játék lehetőségeiről szól.

**Felhasználói élmény**

* Az egyszerű és áttekinthető elrendezés segíti a látogatókat az oldal felfedezésében.
* A reszponzív kialakítás biztosítja, hogy az oldal minden eszközön jól használható legyen, legyen szó asztali gépről, tabletről vagy mobilról.

Ez a sablon hatékonyan ötvözi az esztétikai vonzerőt és a funkcionalitást, így ideális belépési pontként szolgál a weboldal többi részére.

**Kapcsolatfelvételi űrlap HTML sablonja (contact.html)**

Ez a sablon a kapcsolatfelvételi funkció megvalósítására szolgál, amely lehetővé teszi a látogatók számára, hogy üzenetet küldjenek az oldal üzemeltetőinek. A sablon a Thymeleaf keretrendszert használja, ami dinamikus adatmegjelenítést és egyszerű integrációt biztosít a Spring Boot backendhez.

A metaadatok és a külső stíluslapok a <head> szakaszban kerülnek betöltésre. A contact-style.css biztosítja az oldal egyedi és reszponzív megjelenését.

A sablon központi eleme a kapcsolatfelvételi űrlap, amely a contact-form azonosítójú <form> elemben található. Az űrlap POST metódussal továbbítja az adatokat a /contact/submit végpontra, amit a backend dolgoz fel.

A hibaüzenetek és a sikeres visszajelzések dinamikusan jelennek meg a Thymeleaf segítségével. Ha egy hibaüzenet van definiálva, az az ***<div th:if="${error}">*** elemben jelenik meg. A sikeres üzenetküldés esetén hasonlóan a ***${success}*** változó tartalmát jeleníti meg.

A felhasználók jogosultságát a Spring Security biztosítja. A ***sec:authorize="isAnonymous()"*** attribútum ellenőrzi, hogy az űrlap elemei csak anonim felhasználók számára jelenjenek meg, például az email cím mező, amely regisztrált felhasználók esetén nem szükséges.

Az űrlap két fő mezőt tartalmaz:

* **Email cím:** Kötelező mező anonim felhasználók számára, amely validálja a helyes email formátumot.
* **Üzenet:** Kötelező mező, amelyben a látogató szabad szöveges üzenetet írhat, legfeljebb 1000 karakter hosszúságban.

A küldés gomb ***<button>*** elemmel van megvalósítva, amely beküldi az űrlap adatait a megadott végpontra.

Az oldal működését egy külső JavaScript fájl, a contact.js biztosítja, amely a felhasználói élmény javításához és az interakciók kezeléséhez nyújt támogatást.

**Bejelentkezési HTML sablon (login.html)**

Ez a sablon a Hatos Lottó weboldalának bejelentkezési felületét valósítja meg. Célja, hogy egyszerű és biztonságos módot biztosítson a felhasználók számára az autentikációhoz, miközben esztétikailag is kellemes élményt nyújt.

**Metaadatok és stílus**

* A ***<head>*** rész a login-style.css stíluslapot tölti be, amely a bejelentkezési oldal kinézetét és elrendezését szabályozza.
* A sablon reszponzív kialakítású, a is-preload no-sidebar osztályokat használja a testreszabott megjelenés érdekében.

**Fő tartalom**

* Az oldal fő elemei a ***<div id="main-wrapper">*** és ***<div id="content">*** részeken belül találhatók, amelyek biztosítják az oldal struktúráját.
* A bejelentkezési űrlap az ***<article>*** elemben található.

**Bejelentkezési űrlap**

* A login-form azonosítójú ***<form>*** elem POST metódussal küldi az adatokat a /login végpontra.
* Az űrlap két mezőt tartalmaz:
  + **Felhasználónév (username):** Kötelező mező, amely a felhasználó bejelentkezési nevét fogadja el.
  + **Jelszó (password):** Kötelező mező, amely titkosított formában küldi el a jelszót a backend számára.
* Az id="submit-button" azonosítójú gomb szolgál az űrlap elküldésére.

**Fiók regisztráció**

* Ha a felhasználónak nincs fiókja, a rendszer egy regisztrációs linket biztosít az űrlap alatt. Ez a link a /register végpontra irányítja a látogatót.

**Felhasználói élmény**

* Az egyszerű és letisztult elrendezés biztosítja, hogy az oldal könnyen kezelhető legyen.
* A kötelező mezők validálása már az űrlapon történik, minimalizálva a hibázási lehetőségeket.

**Biztonság**

* Az autentikáció folyamata a backendben történik, biztosítva a felhasználói adatok biztonságát.
* A jelszó mező type="password" attribútuma biztosítja, hogy a jelszó rejtett formában jelenjen meg az űrlap kitöltése közben.

Ez a sablon egyszerű, átlátható és funkcionális, ideális választás a bejelentkezési folyamat gyors és biztonságos lebonyolításához.

**Regisztrációs HTML sablon (register.html)**

Ez a sablon a Hatos Lottó weboldalának regisztrációs felületét valósítja meg, amely lehetővé teszi az új felhasználók számára, hogy fiókot hozzanak létre. A cél egy könnyen kezelhető, biztonságos és esztétikus regisztrációs folyamat biztosítása.

**Metaadatok és stílus**

* A <head> rész a register-style.css stíluslapot tölti be, amely a regisztrációs oldal egyedi megjelenéséért felelős.
* A sablon tartalmaz egy oldalspecifikus címet, amely a "Register" feliratot jeleníti meg.

**Tartalom**

* Az oldal tartalma a <div id="content"> elemben helyezkedik el, amely biztosítja a központi elrendezést és struktúrát.
* A fejléc (<h1>) egyértelműen jelzi az oldal célját: "Regisztráció".

**Regisztrációs űrlap**

* Az űrlap a register-form azonosítót kapja, és a POST metódussal küldi az adatokat a /register végpontra.
* A Thymeleaf th:object és th:field attribútumai a backendben lévő user objektum adatait kötik az űrlap mezőihez.
* **Mezők:**
  + **Felhasználónév:** Egyedi név, amely az azonosítást szolgálja. Kötelező mező.
  + **Email cím:** Az email formátumot követő kötelező mező, amely a kapcsolattartásért és hitelesítésért felelős.
  + **Jelszó:** Titkosított formában továbbított kötelező mező.
  + **Jelszó megerősítése:** A felhasználói hibák elkerülése érdekében a jelszót újra meg kell adni.
* A "Register" gomb küldi el az űrlap adatokat.

**Hibaüzenetek**

* Ha regisztráció közben hiba lép fel (pl. nem egyező jelszavak, már létező felhasználónév), a hibaüzenet piros színnel jelenik meg a ${error} változó tartalma alapján.

**Felhasználói élmény**

* Az egyszerű és átlátható elrendezés segíti a felhasználókat az űrlap gyors és helyes kitöltésében.
* A kötelező mezők biztosítják, hogy minden szükséges adatot megadjanak.
* A reszponzív kialakítás biztosítja, hogy a regisztrációs felület minden eszközön megfelelően jelenjen meg.

**Biztonság**

* Az adatok továbbítása HTTPS-en keresztül történik, biztosítva a felhasználói adatok védelmét.
* A backend validálja az adatokat, így elkerülhetők a nem megfelelő regisztrációs próbálkozások.

**Üzenetek megtekintése HTML sablon (messages.html)**

Ez a sablon a Hatos Lottó weboldalán az elküldött üzenetek listázására szolgál. A célja, hogy az adminisztrátorok vagy a felhasználók áttekinthető táblázatos formában láthassák a korábban elküldött üzeneteket.

**Metaadatok és stílus**

* A <head> szakaszban nincsenek specifikus stíluslapok betöltve, azonban az oldal örökli a fő sablonban meghatározott alapstílusokat.
* A is-preload no-sidebar osztályok biztosítják a letisztult és sidebar nélküli megjelenést.

**Tartalom**

* Az oldal központi eleme a <div id="content"> rész, amely tartalmazza az üzenetek megjelenítésére szolgáló táblázatot.

**Üzenetek táblázat**

* A <table> elem az üzenetek megjelenítéséhez használt táblázatot határozza meg.
* **Fejléc:**
  + Három oszlopot tartalmaz: Név, Üzenet és Időpont.
* **Dinamikus tartalom:**
  + A Thymeleaf th:each attribútuma segítségével a ${messages} lista elemei kerülnek betöltésre.
  + Minden üzenet egy <tr> sorban jelenik meg:
    - **Név:** Az üzenetet küldő felhasználó neve.
    - **Üzenet:** Az üzenet tartalma.
    - **Időpont:** Az elküldés dátuma és időpontja, a ${message.formattedDate} mező alapján.

**Felhasználói élmény**

* A táblázat kialakítása egyszerű és átlátható, amely biztosítja a könnyű áttekinthetőséget.
* A dinamikusan betöltött tartalom garantálja, hogy az aktuális adatok jelennek meg a felhasználók számára.

**Dinamika és funkcionalitás**

* Az üzenetek listáját a backend biztosítja a ${messages} objektum formájában.
* Az adatok formázása és előkészítése a backendben történik, például a dátumot a formattedDate mező tartalmazza, amely könnyen olvasható formátumban kerül megjelenítésre.

**Reszponzivitás**

* Az alapstílusok és táblázati elemek automatikusan alkalmazkodnak a képernyőmérethez, biztosítva, hogy a tartalom minden eszközön jól jelenjen meg.

**Felhasználási terület**

* Ez a sablon különösen hasznos adminisztrációs felületeken, ahol szükséges az üzenetek gyors és egyszerű áttekintése.
* A megjelenítési logika és az adatok kezelése könnyen bővíthető, például további oszlopok vagy funkciók (pl. törlés vagy válaszadás) hozzáadásával.

Ez a sablon áttekinthető, funkcionális, és a felhasználók számára egyszerű hozzáférést biztosít az elküldött üzenetekhez.

**Lottó Adatok Lekérdezési HTML sablon (lotto\_query.html)**

Ez a sablon a lottó adatok év és hét alapján történő lekérdezésére szolgál. A célja, hogy a felhasználók egyszerűen kiválaszthassák az érdeklődésüknek megfelelő időszakot, és gyorsan megjeleníthessék az adott hét lottóhúzási adatait.

**Metaadatok és stílus**

* A <head> szakaszban a lotto-query-style.css stíluslapot tölti be, amely az oldal egyedi megjelenéséért felelős.
* A sablon reszponzív kialakítású, a is-preload no-sidebar osztályokat használja a tiszta és sidebar nélküli megjelenés érdekében.

**Tartalom**

* A <div id="content"> szakasz tartalmazza az oldal fő funkcióit:
  + **Cím és leírás:** A <h2> elem jelzi az oldal célját, és a <p> leírás segíti a felhasználót a használatban.
  + **Űrlap:** Az év és hét kiválasztására szolgáló lenyíló menük és egy lekérdezési gomb található az űrlapban.

**Űrlap felépítése**

* Az űrlap az id="lotto-form" azonosítót kapta.
* **Lenyíló menük:**
  + **Év (year):** Egy dinamikusan feltöltött lenyíló lista, amely az elérhető évek közül engedi a választást.
  + **Hét (week):** Egy másik lenyíló lista, amely az adott évben elérhető hetek alapján frissül.
* **Lekérdezés gomb:**
  + A query-button azonosítójú gomb indítja el a lekérdezést. A gomb JavaScript segítségével hívja meg a backend szolgáltatást.

**Eredmények**

* Az id="results" szakasz dinamikusan tölti be a lekérdezés eredményeit. Az eredmények formátuma (pl. táblázat vagy lista) a backend által visszaadott adatoktól és a frontend megjelenítési logikájától függ.

**Működés**

* A lenyíló listák tartalma és az eredmények megjelenítése JavaScript segítségével valósul meg. Az oldal betöltésekor a lotto\_query.js fájl betölti az elérhető éveket, majd a felhasználó által kiválasztott év alapján frissíti a hét mező tartalmát.
* A lekérdezési gomb megnyomása a backend szolgáltatáshoz küld egy kérést (például egy REST API-t használva), és a kapott adatokat a results szakaszba jeleníti meg.

**Felhasználói élmény**

* Az egyszerű és intuitív elrendezés segíti a felhasználókat az adatok gyors és pontos lekérdezésében.
* A form-validáció gondoskodik arról, hogy mindkét mezőt kötelezően ki kelljen tölteni a lekérdezés elindítása előtt.

**Reszponzivitás**

* A form és az eredmények szakasz CSS segítségével biztosítja a megfelelő megjelenést minden képernyőméreten, legyen szó asztali gépről vagy mobilról.

***Java Scriptek:***

Ez a rész kizárólag azokat a scripteket taglalja, amelyeket mi fejlesztettünk a projekthez, nem foglalkozik az ingyenes sablon scripjeivel.

**JavaScript az év, hét, és lottó adatok dinamikus betöltéséhez (lotto\_query.js)**

Ez a JavaScript kód a lottó lekérdezési funkcióhoz kapcsolódik, és biztosítja az év, hét, valamint a lottó adatok dinamikus betöltését és megjelenítését. A kód reszponzív és felhasználóbarát megoldást kínál az adatok kezelésére és megjelenítésére.

**Fő funkciók**

**1. Alapvető eseménykezelés**

* **DOMContentLoaded esemény:** Az eseménykezelő biztosítja, hogy a JavaScript futtatása az oldal teljes betöltődése után kezdődjön.
* **Év kiválasztása:** Az év legördülő menü változásának figyelése és az adott évhez tartozó hetek betöltése.
* **Lekérdezési gomb működése:** Az év és hét kiválasztása után a lottó adatok lekérése. Ha hiányzik a szükséges információ, figyelmeztetést jelenít meg.

**2. Évek betöltése (**loadYears**)**

* Az /api/huzas/years API végpontról kéri le az elérhető éveket.
* A kapott évadatokat hozzáadja az év legördülő menü opcióihoz.
* Dinamikusan inicializálja az alapértelmezett opciókat.

**3. Hetek betöltése (**loadWeeks**)**

* Az adott év alapján tölti be a hozzá tartozó heteket az /api/huzas/weeks API végpontról.
* Az alapértelmezett opciók mellett hozzáadja a lekért adatokat a week legördülő menühöz.

**4. Lottó adatok lekérdezése (**fetchLottoResults**)**

* Az /api/huzas/results API végpontról kéri le a megadott év és hét lottó adatait.
* Az adatok feldolgozása és megjelenítése a következőképpen történik:
  + **Alap információk megjelenítése:** Az év, hét és a húzott számok listázása.
  + **Nyeremények feldolgozása és formázása:**
    - Az elérhető nyeremények ellenőrzése és megjelenítése, beleértve a találatok, darabszám és érték formázott kiírását.
    - A találatok speciális formázása (pl. 7 találat -> 5+1).
    - 3 számjegyenként pontozott formátum a darabszám és az értékek megjelenítéséhez.
  + **Hibaüzenetek:** Amennyiben nincs adat vagy valamilyen hiba történik, megfelelő üzenetek jelennek meg.

**Felhasználói élmény**

* **Interaktív legördülő menük:** Az év és hét kiválasztása automatikusan frissíti a hozzájuk kapcsolódó adatokat.
* **Értesítések és figyelmeztetések:** Ha az év és hét nem kerül kiválasztásra, figyelmeztető üzenet jelenik meg.
* **Dinamizmus:** Az eredmények az oldal újratöltése nélkül jelennek meg, fokozva a felhasználói élményt.

**Hibakezelés**

* **API-hibák:** A kód kezeli az API hívások során fellépő hibákat, és felhasználóbarát hibaüzeneteket jelenít meg a konzolon és az oldalon.
* **Adatok hiánya:** Ha az adott évhez vagy héthez nem érhető el adat, figyelmeztető szöveg jelenik meg a felhasználónak.

**Funkciók bővíthetősége**

* **API-végpontok:** A meglévő logika könnyen bővíthető további API végpontokkal, például havi statisztikák vagy több éves adatok megjelenítéséhez.
* **Megjelenítési funkciók:** Az adatokhoz kapcsolódó további információk (pl. statisztikák, diagramok) egyszerűen hozzáadhatók.

Ez a JavaScript kód hatékonyan kezeli a lottó lekérdezési funkció működését, miközben rugalmas és bővíthető alapot biztosít a jövőbeli fejlesztésekhez.

**Karakterek számlálása JavaScript-tel (contact.js)**

Ez a kód egy egyszerű és hatékony megoldást nyújt arra, hogy valós időben nyomon kövessük egy szövegmező (textarea) kitöltése során a hátralévő karakterek számát. A megvalósítás különösen hasznos lehet olyan űrlapok esetében, ahol a felhasználóknak maximális karakterlimitet kell betartaniuk, például üzenetküldési vagy megjegyzésmezők esetében.

**Kód működése**

1. **Eseményfigyelő (Event Listener):**
   * A kód egy input eseményfigyelőt ad hozzá az id="message" azonosítójú szövegmezőhöz.
   * Az input esemény minden alkalommal lefut, amikor a felhasználó szöveget ír, töröl, vagy más módon módosítja a szövegmező tartalmát.
2. **Karakterek számlálása:**
   * **Maximális hossz:** A maxLength változóban van definiálva, jelen esetben 1000 karakter.
   * **Jelenlegi hossz:** Az e.target.value.length segítségével lekérdezi a mező aktuális tartalmának hosszát.
   * **Hátralévő karakterek:** A maximális hosszúságból kivonja az aktuális hosszúságot, és az eredményt a remaining változóban tárolja.
3. **Visszajelzés megjelenítése:**
   * A remaining változó értékét valós időben frissíti a remaining-chars azonosítójú HTML elem szöveges tartalmában (textContent).

**Felhasználói élmény**

* **Valós idejű visszajelzés:** A felhasználók folyamatosan látják, hány karaktert használhatnak még.
* **Egyszerűség és hatékonyság:** A karakterlimit vizuális iránymutatást ad, segítve a felhasználókat abban, hogy ne lépjék túl a megengedett hosszúságot.

**Kiterjeszthetőség**

* **Testreszabás:** A maximális hosszúság (maxLength) könnyen módosítható a konkrét felhasználási esettől függően.
* **Több mező támogatása:** A kód egyszerűen kibővíthető több szövegmező kezelésére, ha az eseményfigyelőt dinamikusan adjuk hozzá minden releváns mezőhöz.

Ez a kód egyszerű, könnyen érthető, és hatékonyan javítja a felhasználói élményt az űrlapok kitöltése során.

***Java osztályok:***

**Kapcsolatfelvételi funkció Java osztályai**

A kapcsolatfelvételi funkciót a Spring Boot keretrendszerben több összekapcsolt Java osztály valósítja meg. Ezek az osztályok biztosítják a felhasználói adatok kezelését, az üzleti logika végrehajtását és az üzenetek adatbázisba mentését.

**ContactController**  
Ez az osztály felelős a kapcsolatfelvételi űrlap beküldési műveleteinek kezeléséért. Az űrlap adatai POST kérésként érkeznek, amelyeket a vezérlő feldolgoz és továbbít a ContactService számára. A vezérlő a szerveroldalon ellenőrzi az adatokat, például hogy az üzenet mező nem üres, és nem haladja meg az 1000 karakteres limitet. Bejelentkezett felhasználók esetén az azonosítás a Principal objektummal történik, amelynek segítségével a felhasználó azonosítóját lekérdezi az adatbázisból. Vendég felhasználók esetén csak az email cím kerül mentésre. Sikeres mentés után a vezérlő sikerüzenetet küld vissza, míg hiba esetén hibaüzenetet jelenít meg.

**ContactEntity**  
Ez az osztály a kapcsolatfelvételi üzenetek adatbázisban történő tárolásához készült. Az @Entity annotáció jelzi, hogy ez egy adatbázis táblát reprezentáló osztály. Az osztály mezői, mint például az id, userId, message, email, és createdAt, a messages tábla oszlopainak felelnek meg. A createdAt mező automatikusan rögzíti az üzenet létrehozásának időpontját.

**ContactRepository**  
Ez egy Spring Data JPA alapú interfész, amely a kapcsolatfelvételi üzenetek adatbázisban történő műveleteit kezeli. Az interfész automatikusan biztosít alapvető műveleteket, mint például a mentés és a lekérdezés.

**ContactService**  
Ez az osztály az üzleti logikát valósítja meg. A saveMessage metódus egy új ContactEntity példányt hoz létre az üzenet adataival, majd az adatokat az adatbázisba menti a ContactRepository segítségével. A metódus hibakezelést is tartalmaz, amely biztosítja, hogy bármilyen hiba esetén megfelelő visszajelzés történjen.

Ezek az osztályok együttműködve biztosítják, hogy a kapcsolatfelvételi funkció biztonságosan és megbízhatóan működjön, miközben a felhasználói élményt is szem előtt tartják.

**Messages Java Osztályok**

A **Messages** modul az üzenetek kezelésére szolgál a Hatos Lottó weboldalán. Ez a modul biztosítja az üzenetek tárolását, feldolgozását és lekérdezését az adatbázisból, valamint az üzenetek megjelenítését a felhasználók számára.

**MessageDto**  
A MessageDto egy adatátviteli objektum (DTO), amely az üzenetek felhasználói felületen történő megjelenítéséhez használatos. Az osztály tartalmazza a felhasználó nevét, az üzenet szövegét és a létrehozás dátumát formázott formában. A dátum formázását a DateTimeFormatter végzi, amely a "yyyy-MM-dd HH:mm:ss" formátumot biztosítja. Az osztály getter és setter metódusokat biztosít az adatok kezeléséhez.

**MessageEntity**  
A MessageEntity az adatbázis messages tábláját reprezentálja. Az osztály tartalmazza az üzenetekhez tartozó adatokat, például az egyedi azonosítót, a felhasználó azonosítóját, az üzenet szövegét, az email címet és a létrehozás dátumát. Az osztály annotációkkal jelzi, hogy melyik mezőhöz melyik adatbázisoszlop tartozik. A @GeneratedValue annotáció automatikusan generált azonosítót biztosít az id mező számára.

**MessageRepository**  
A MessageRepository a Spring Data JPA interfésze, amely a MessageEntity entitás adatbázisműveleteit végzi. A findAllByOrderByCreatedAtDesc(Pageable pageable) metódus lehetővé teszi, hogy a legfrissebb üzeneteket csökkenő sorrendben lehessen lekérdezni, például az utolsó 10 üzenetet.

**MessageService**  
A MessageService az üzleti logikát valósítja meg az üzenetek kezeléséhez. A getLatestMessages metódus a legfrissebb 10 üzenetet lekéri az adatbázisból, és azokat MessageDto formátumba alakítja. A metódus a felhasználói azonosító alapján a UserRepository segítségével betölti a felhasználó nevét. Ha az üzenet vendég felhasználótól származik, a felhasználó neve "Vendég" lesz. A metódus a JPA által visszaadott entitásokat stream-re alakítja, feldolgozza, majd listává gyűjti.

**Működés**  
Az üzenetek lekérdezésekor a MessageRepository a legfrissebb 10 üzenetet adja vissza a létrehozási dátum alapján csökkenő sorrendben. Az adatok feldolgozása során a MessageService minden üzenetet MessageDto formátumba alakít, amely tartalmazza az üzenet szövegét, a felhasználó nevét és a formázott dátumot. A feldolgozott adatokat a vezérlő továbbítja a felhasználói felületnek, amely ezeket a megfelelő formátumban jeleníti meg. A rendszer biztosítja, hogy az adatok átláthatóan és könnyen kezelhető módon jelenjenek meg a felhasználók számára.

**Login Java Osztályok**

A login modul a felhasználók biztonságos bejelentkezését és regisztrációját biztosítja a Spring Security és a Spring Boot keretrendszer segítségével. A modul magába foglalja az adatbázissal való interakciót, az adatok validációját, valamint a felhasználói jogosultságok kezelését.

**CustomUserDetailsService**  
Ez az osztály implementálja a Spring Security által meghatározott UserDetailsService interfészt, amely a felhasználók hitelesítéséhez szükséges adatokat biztosítja. A loadUserByUsername metódus felel a felhasználó betöltéséért az adatbázisból a felhasználónév alapján. Ha a felhasználó nem található, UsernameNotFoundException kivételt dob. A metódus a felhasználóhoz tartozó szerepeket is betölti, és azokat jogosultságokként adja vissza.

**User**  
A User osztály az adatbázis users tábláját reprezentálja. Az osztály azonosítót, felhasználónevet, email címet, jelszót, jelszómegerősítést (amely nem kerül mentésre az adatbázisba), valamint a felhasználó szerepét tartalmazza. Az osztály a felhasználói adatok kezeléséhez getter és setter metódusokat biztosít.

**UserRepository**  
A UserRepository egy Spring Data JPA alapú interfész, amely a User entitáshoz kapcsolódó adatbázisműveleteket végzi. A findByUsername metódus lehetővé teszi, hogy a felhasználót a felhasználónév alapján keressük meg.

**UserService**  
Ez az osztály az üzleti logikát valósítja meg a felhasználói műveletekhez, például a regisztrációhoz. A regisztráció során ellenőrzi, hogy a felhasználónév már létezik-e, valamint hogy a jelszavak egyeznek-e. A jelszót a Spring Security PasswordEncoder segítségével titkosítja, majd menti az új felhasználót az adatbázisba. Az osztály a userRepository segítségével menti az adatokat az adatbázisba.

**Működés**  
Amikor egy felhasználó regisztrál, a vezérlő (LoginController) továbbítja az űrlap adatait a UserService-nek. A UserService validálja az adatokat, titkosítja a jelszót, majd menti az új felhasználót az adatbázisba. Bejelentkezéskor a Spring Security a CustomUserDetailsService segítségével ellenőrzi a felhasználó létezését, és betölti a hitelesítési adatait. A megfelelő szerepek és jogosultságok hozzárendelésével biztosítja a hozzáférést a védett erőforrásokhoz.

**Lottó lekérdezési modul Java osztályai**

A lottó lekérdezési modul lehetővé teszi az adatok, például az évek, hetek és lottószámok részletes lekérdezését az adatbázisból. A modul REST API-kat biztosít, amelyek az adatok kezeléséért és visszaadásáért felelnek a front-end számára.

**LottoController**  
A vezérlő a lottó adatok lekérdezésére szolgáló végpontokat definiálja. A REST API-k az alábbi funkciókat valósítják meg:

* Az elérhető évek lekérdezése (/api/huzas/years).
* Az adott évhez tartozó hetek lekérdezése (/api/huzas/weeks).
* Az adott év és hét részletes adatainak (lottószámok és nyeremények) lekérdezése (/api/huzas/results).

**LottoDto**  
Ez az adatátviteli objektum (DTO) a lottó húzások részletes adatait tartalmazza:

* Az év és hét, amelyre a lekérdezés vonatkozik.
* A húzott számok egy szövegként tárolva.
* A nyeremények listája, amelyeket NyeremenyDto objektumok reprezentálnak.

**NyeremenyDto**  
Ez az osztály az egyes nyeremények részleteit tartalmazza:

* A találatok száma.
* A nyertes szelvények darabszáma.
* A nyeremény összege.

**LottoEntity**  
Az osztály az adatbázis huzas tábláját reprezentálja, amely az év, hét és húzott számok adatainak tárolásáért felel. Az @Entity annotáció jelzi, hogy az osztály egy adatbázis táblát reprezentál. Az egyedi azonosító (id), az év (ev), a hét (het), és a húzott számok (szamok) mezőit tartalmazza.

**LottoRepository**  
Ez az interfész a Spring Data JPA segítségével biztosítja az adatbázisműveleteket. A főbb metódusai:

* Az elérhető évek lekérdezése (findDistinctYears).
* Az adott évhez tartozó hetek lekérdezése (findByEv).
* Az adott év és hét részletes adatainak lekérdezése (findHuzasWithDetails), amely natív SQL lekérdezést használ.

**LottoService**  
A szolgáltatás az üzleti logikát valósítja meg. A főbb funkciói:

* Az elérhető évek lekérdezése a findDistinctYears metódus segítségével.
* Az adott év heteihez tartozó adatok lekérdezése a findByEv metódussal.
* Az adott év és hét részletes adatainak feldolgozása a findHuzasWithDetails metódus által visszaadott adatok alapján. Az adatok LottoDto objektumokká alakítása biztosítja, hogy a front-end könnyen kezelhető formátumot kapjon.

**Működés**  
A lottó lekérdezési folyamat a következő lépésekből áll:

1. A vezérlő (LottoController) kezeli az API-hívásokat, és a megfelelő szervizmetódusokat hívja meg.
2. A szerviz (LottoService) a repository-t használja az adatok lekérdezésére az adatbázisból.
3. A nyers adatokat DTO objektumokká alakítja a könnyebb feldolgozás érdekében.
4. Az adatok JSON formátumban kerülnek visszaadásra az API-t hívó front-end számára.

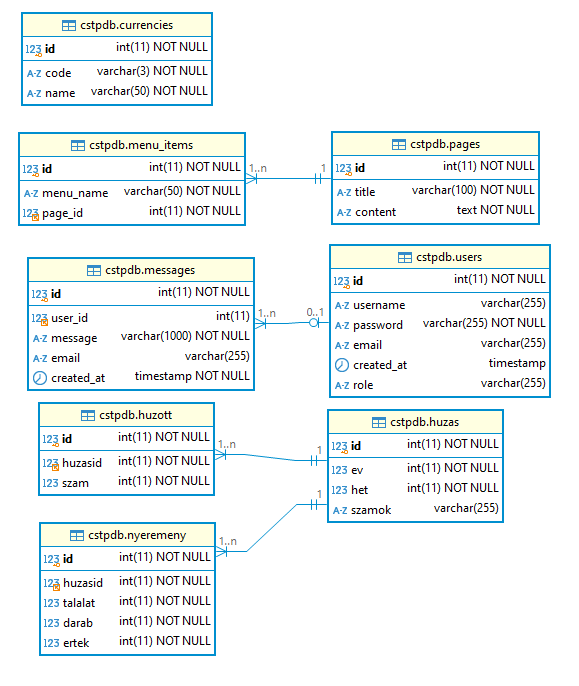
**Kapcsolódás a front-endhez**  
A REST API-k lehetővé teszik a dinamikus lekérdezéseket a front-end által. Az évek és hetek adatait a front-end betölti, amikor a felhasználó kiválasztja az évhez tartozó opciókat. Az adatok JSON formátumban érkeznek a front-endhez, ahol megjelenítésre kerülnek, például táblázatokban vagy grafikonokon. A lekérdezések optimalizáltak, így gyorsan és hatékonyan szolgáltatják az adatokat még nagy mennyiség esetén is.

## *Adatkezelés*

Az adatkezelés során a nethely.hu által szolgáltatott **MySQL** alapú adatbázist használjuk.

**Feladata:** Az adatbázis elsődleges feladata egy olyan megbízható háttérszolgáltatás biztosítása, amely lehetővé teszi a felhasználók regisztrációját és bejelentkezését, valamint azonosításukat az űrlapok kitöltésekor. Emellett támogatja a különböző jogosultságok kezelését.

**Felépítése:** Az adatbázis pontos felépítése az alábbi ER diagrammon tekinthető meg:



Az adatbázismodell magában foglal egy felhasználók táblát, amely az alapvető felhasználói adatokat, mint például felhasználónév és jelszó tárolja, valamint egy külön email táblát az űrlapokkal kapcsolatos adatok kezelésére. Ez a struktúra lehetővé teszi az egyszerű felhasználói műveletek, mint a regisztráció és bejelentkezés, valamint az űrlapokon keresztüli kommunikáció kezelését.

## *Tesztek és kiegészítő információk*

**Postman és cURL RESTful API teszt:**

A weboldal API-ját Postman-el és CURL-el is teszteltük.

Az api/huzas/years végpont tesztje:  
A screenshot of a computer

Description automatically generated

CURL-el:

A screenshot of a computer

Description automatically generated

Az api/huzas/results végpont tesztje:

A screenshot of a computer

Description automatically generated

Curl-el:  
A screen shot of a computer

Description automatically generated

**GitHub projekt elérhetőség:**   
**https://github.com/irkm813/java\_gyak\_beadando**