**Ninjastic**

Felhasználói dokumentáció

# Nyilatkozat

Jancsik Balázs Zoltán, Erdős Attila és Csendes Marcell nyilatkozunk arról, hogy a szakdolgozatot mi hárman csináltuk, nem másoltuk, a hivatkozásokat megfelelően használtuk, máshol még nem adtuk be és meg nem engedett segédanyagot nem használtunk.

Tartalomjegyzék

[Nyilatkozat 2](#_Toc163586142)

[Bevezetés, témafelvetés 5](#_Toc163586143)

[Szakirodalmi áttekintés 5](#_Toc163586144)

[Apache 5](#_Toc163586145)

[MySQL 5](#_Toc163586146)

[PHP 6](#_Toc163586147)

[Fórum jelentősége 7](#_Toc163586148)

[Anyag és módszer 7](#_Toc163586149)

[Adatbázis tervezés elmélete 7](#_Toc163586150)

[Programozás elmélete 8](#_Toc163586151)

[Adatbázis tervezése 8](#_Toc163586152)

[Admin felület 12](#_Toc163586153)

[Bejelentkezés 12](#_Toc163586154)

[Dashboard 13](#_Toc163586155)

[Felhasználók 13](#_Toc163586156)

[Listázás 13](#_Toc163586157)

[Létrehozás/Szerkesztés 14](#_Toc163586158)

[Törlés 14](#_Toc163586159)

[Topikok 15](#_Toc163586160)

[Listázás 15](#_Toc163586161)

[Létrehozás/Szerkesztés 15](#_Toc163586162)

[Törlés 16](#_Toc163586163)

[Hozzászólások 16](#_Toc163586164)

[Listázás 16](#_Toc163586165)

[Létrehozás / Szerkesztés 17](#_Toc163586166)

[Törlés 17](#_Toc163586167)

[Csúnya szavak 18](#_Toc163586168)

[Listázás 18](#_Toc163586169)

[Létrehozás / Szerkesztés 18](#_Toc163586170)

[Törlés 19](#_Toc163586171)

[Adminok 19](#_Toc163586172)

[Listázás 19](#_Toc163586173)

[Létrehozás / Szerkesztés 20](#_Toc163586174)

[Törlés 20](#_Toc163586175)

# Bevezetés, témafelvetés

A veszprémi egyetem hallgatójaként a számítástechnika és a programozói világ számos ágával volt lehetőségünk találkozni. Bővül a lexikális tudásunk és szerencséseknek mondhatjuk magunkat, hogy tanáraink rengeteg időt fordítottak gyarkorlati képzésünkre is. A szakdolgozat elkészítése során elsősorban az így megszerzett ismereteket, a magyar és angol szakirodalmat, valamint internetes szakanyagokat használtam fel. Dolgozatunk témájának egy fórum elkészítését választottuk.

Napjainkban a világ egyik legelterjebb kommunikációs csatornája az interneten a fórum, ahol mindenkinek lehetősége van akár segítség kérésre vagy nyújtásra. Például visszanézhetőek korábbi beszélgetések, amely a mai fiatal generációnak segítséget nyújthat a pályakezdésükben és a felmerülő kezdetleges problémákban.

Rengeteg fórumot láttunk már, de magunktól még sosem készítettünk el egyet sem. Úgy gondoltuk, ezzel a feladattal érdemes foglalkozni. Mindemelett HTML, PHP, MySql, JavaScript, Jquery, Ajax, Symfony, Docker, Vue Js, Twig és sok egyéb technológia felhasználását bevettük.

A jövőben fórum, webshop, blog, ERP és egyéb típuső web és konozolos alkalmazások fejlesztésével szeretnénk foglalkozni. Ezen belül a PHP frameworkok, JavaScript frameworkok, Microservicek és egyéb kommunikációs megoldások megvalósításával.

# Szakirodalmi áttekintés

## Apache

Az Apache HTTP Server (röviden Apache) nyílt forráskódú webkiszolgáló alkalmazás. Az

Apache program jellemzőivel párhuzamba hozható az amerikai indián törzsek találékonysága

és alkalmazkodási képessége. Ez egy ingyenes program, amely fontos szerepet játszott az

Internet elterjedésében. A projekt célja, olyan webszerver szoftver megalkotása és fejlesztése,

amely eleget tesz a rohamosan fejlődő világ és az Internet igényeinek, biztonságos, üzleti-,

vállalati felhasználásra is alkalmas és emellett szabadon használható. Az Apache a régi

NCSA HTTPd szerverre épül és az Apache Szoftver Licenc feltételeivel terjesztik. Az első

webszervert egy angol fizikus, Tim Berners-Lee, alkotta meg. Az Apache sok szabványt

támogat, melyeknek nagy része lefordított modulok formájában áll rendelkezésre. Ezek a

modulok sok területet lefednek, a kiszolgáló oldali programnyelv támogatástól kezdve a

hitelesítési sémákig. Az ismertebb, támogatott programnyelv modulok a Perl, Python, Java,

Tcl és a Php. A legnépszerűbbek modulok a mod\_access, mod\_auth és a mod\_digest. Statikus

és dinamikus weboldalak közzétételére egyaránt használják. Sok webalkalmazást az Apache által nyújtott környezethez és szolgáltatásokhoz terveznek. Az Apache alkotja a webszerver

komponenst a népszerű LAMP alkalmazáscsomagban, melynek további komponensei a

MySQL adatbázisszerver és a PHP/Perl/Python programozási nyelvek.

## MySQL

A MySQL egy több felhasználós, többszálú, SQL-alapú relációs adatbázis-kezelő szerver.

A szoftver eredeti fejlesztője a svéd MySQL AB cég, amely kettős licencezéssel tette

elérhetővé a MySQL-t. Választható módon vagy a GPL, vagy egy kereskedelmi licenc

érvényes a felhasználásra. 2008 januárjában a Sun felvásárolta 800 millió dollárért a

céget. 2010. január 27-én a Sun az Oracle Corporation tulajdonába került. A MySQL az

egyik legelterjedtebb adatbázis-kezelő, aminek egyik oka lehet, hogy a teljesen nyílt

forráskódú LAMP összeállítás részeként költséghatékony és egyszerűen beállítható

megoldást ad dinamikus webhelyek szolgáltatására.

## PHP

A PHP általános szerver oldali szkript nyelv, melyet dinamikus weblapok megalkotására

hoztak létre. Az első szkript nyelvek egyike, amely külső fájl használata helyett HTML

oldalba ágyazható. A kódot a webszerver PHP processzor modulja értelmezi, ezzel dinamikus

weboldalakat hoz létre. Rasmus Lerdorf 1995-ben indította útjára. Ma a The PHP Group tartja

fenn és fejleszti. A PHP szabad szoftver, de licence nem csereszabatos a GNU licensszel,

mivel megkötéseket tartalmaz a PHP név használatára. A PHP nyelv túlnőtt eredeti

jelentőségén. Születésekor csupán egy makrókészlet volt, amely személyes honlapok

karbantartására készült. Innen ered a neve is: Personal Home PageTools. Később a PHP

képességei kibővültek, így egy önállóan használható programozási nyelv alakult ki, amely

képes nagyméretű webes adatbázis alapú alkalmazások működtetésére is.

**A böngésző és a szerver közötti kommunikáció folyamata**

A szerver-kliens kapcsolat kétirányú. A böngészőbe a felhasználó beírhat egy URL-t, rákattint

egy linkre, ekkor egy úgynevezett GET metódusú kérést küld el a szervernek. Ezen felül az

űrlap kitöltése után az elküldés gombra kattintás lehetőséget ad arra, hogy POST (vagy GET)

metódusú kérést küldjünk a szervernek illetve onnan választ is kapjunk a böngészőn

keresztül. A kérés tulajdonképpen egy adatcsomag, ami több lépésből áll. Először a kérésben

elmegy az URL, a metódus fajtája POST vagy GET, a használandó protokoll HTTP 1.0 vagy

HTTP 1.1. Ezt követően POST metódus esetén a kérésben elmennek az űrlapban kitöltött

adatmezők adatai, majd a böngésző neve, és speciális tulajdonságai is. Ezután a böngésző

által lekezelni képes speciális adatfajták szabványos nevei kerülnek továbbításra. Majd a böngészőben tárolt és az adott domainhez tartozó COOKIE-k adatai következnek. Ha az URL

a szerveren .html, .htm ,.gif, .mpeg, stb... file-t jelöl meg, akkor a webszerver kikeresi a saját

háttértárán a megfelelő fájlt és elküldi a böngészőnek különböző plusz, úgynevezett fejléc

adatok (HEADER) kíséretében. A HEADER-ben lévő adatok a következők lehetnek. A

kódolás nyelve, a cache-ek működésére vonatkozó parancsok, az esetlegesen elküldendő

COOKIE-k, a SESSION-ökhöz tartozó úgy nevezett SESSION COOKIE-k. Ha az URL .php

kiterjesztésű file-t jelöl, akkor a webszerver átadja az URL-t a PHP motornak. Itt legelőször a

PHP motor megkeresi a háttértáron a megadott file-t. Ezután betölti, szintaktikailag ellenőrzi

a betöltött php file tartalmát, majd elkezdi értelmezni és végrehajtani azt. Ha szükséges elindít

a PHP-hez konfigurált egyéb modulokat, kommunikál velük, mint egy SQL szerver. A PHP

kód segítségével a kimenetre lehet küldeni HEADER parancsokat, COOKIE-k és SESSION-

ök adatait. Ez a folyamat a PHP motor HTML kód küldése előtt történik. A programírás

során, a PHP oldalakon keverni lehet a HTML és a PHP kódot. Ha az értelmező HTML kód

részt talál, akkor ezt a kódot küldi a kimenetre változtatás nélkül. A PHP motor működése

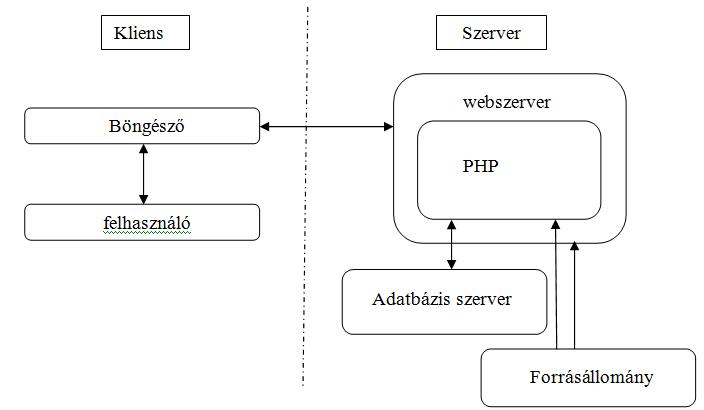
során kiíró utasításokkal előállítja a HTML forráskódot és elküldi a kimenetre. Ezt az

információt a webszerver kapja meg. A webszerver a tartalmat tovább juttatja a böngészőnek.

A böngésző a megkapott információt betölti és értelmezi. Ha a kódban jó JavaScript, Flash,

Java vagy egyéb nem kifejezetten HTML kód található, akkor a böngésző elindítja a kliens

számítógépre telepített megfelelő feldolgozó modult.



1. **ábra: Szerver-kliens kapcsolata (forrás: saját ábra)**

## Fórum jelentősége

A fórumok lehetőséget biztosítanak a felhasználók számára arra, hogy szöveges párbeszédet folytassanak egymással. Itt az üzeneteket témák szerint csoportosítják, ami a jobb átláthatóságot segíti elő. A „bulletin board” az Internet elterjedését megelőző korszak legnépszerűbb elektronikus adatcsere rendszere volt, amelyben egy adott téma iránt érdeklődő felhasználók cserélhettek üzeneteket és állományokat egymással. Képernyőn való megjelenítése kizárólag szöveges jellegű volt, elérésére közvetlenül a kiszolgálóra történő telefonos betárcsázás útján nyílt lehetőség. Manapság az angol "bulletin board" kifejezést a magyar nyelvben "fórum" név helyettesíti, amit általában webes alapú üzenetcserét lehetővé tevő rendszerek megnevezésére használnak. A fórumok többféleképpen épülhetnek fel. Többnyire minimálisan két szintből állnak. Ezekben általában az a közös vonás, hogy először különböző kulcsszavak, címek alapján bontják szét a tartalmat, majd ezeken belül találhatjuk meg a hozzászólásokat.

# Anyag és módszer

## Adatbázis tervezés elmélete

Az adatbázisok tervezését és kivitelezését egy összetett rendszerszervezési és elemzési folyamat előzi meg. Nagyon lényeges, hogy a tervező megismerje az adatbázisban eltárolandó adatok természetét. A normalizálás eredményét általában három lépéssel érhetjük el. A normalizálás lényegében táblázatszétbontó relációs műveletek sorozata, amelyek eredményeként egymással kapcsolatban álló, az eredetinél kisebb tárolási igényű relációkat kapunk. A szétbontás haszna nagyjából így foglalható össze. Csökken a tárolási igény (bizonyos adatokat nem tárolunk feleslegesen többször is), megszűnnek a törlési, módosítási és beszúrási problémák (anomáliák) és logikailag áttekinthetőbb lesz a táblázat. A normalizálás folyamata egzakt matematikai elveken alapuló művelet. Fontos megismerkedni néhány matematikai fogalommal, mely a normalizálás folyamán előfordulnak. Funkcionális függőség P-attribútum részhalmaz funkcionálisan meghatározza Q-attribútum halmazt, ha minden P-hez tartozó értékhez (a P attribútumainak oszlopaiban álló értékekhez) pontosan egy Q-hoz tartozó értéket tudunk hozzárendelni. Tehát P értékei egyértelműen maghatározzák Q értékeit. Teljes funkcionális függőség: Azt mondjuk, hogy Q teljes funkcionális függőségben van P-vel, ha P-ből képzett attribútum részhalmazoktól nem függ a Q attribútum halmaz. Ellenkező esetben részleges függőségről beszélünk. Tranzitív funkcionális függés "A" attribútum részhalmaztól tranzitív funkcionális függőségben van "C" attribútum részhalmaz, ha létezik egy"B" részhalmaz, amely "A"-tól függ, a "C" pedig "B" től. Tehát mondhatjuk, hogy a két funkcionálisan függő attribútum halmaz mellett található egy harmadik részhalmaz, mely a két halmaz közötti funkcionális függőséget átviszi (tehát direkt módon is függnek egymástól és áttételesen is). A normalizálás végrehajtása során egymásra épülő, egyre kevesebb redundanciát tartalmazó normálformák keletkeznek. 6 egymásra épülő normálformát ismerünk: 1NF, 2NF, 3NF, BCNF, 4NF, 5NF azaz első, második, harmadik, Boyce-codd normálforma, negyedik és ötödik normálforma. Az első három normálforma a funkcionális függőségekben található redundanciák, míg a negyedik és ötödik a többértékű függőségekből eredő redundanciák megszüntetésére szolgál. Gyakorlatilag a 3. normálformában lévő adatbázis redundancia mentesnek tekinthető. Az esetek többségében elegendő a normalizálást a 3. normálformáig elvégezni. 0NF-ben az adatbázis nem normalizált alakját szokás 0.-dik normálformának nevezni. A tervezett adatbázisunk akkor van első normálformában, ha minden sora különböző, az oszlopok száma, sorrendje minden sorban azonos, minden oszlop csak egy attribútum értéket vesz fel és minden sorhoz egy egyedi kulcs tartozik, amitől az összes többi attribútum funkcionálisan függ. Második normálformában akkor van, ha első normálformában van (előfeltétel) és a nem kulcs attribútumok funkcionálisan teljesen függnek az elsődleges kulcstól. Harmadik normálformában akkor van, ha második normálformában van és funkcionális függés csak az elsődleges kulcsból indul ki; vagyis megszüntettük a közvetett (tranzitív) függéseket. A normalizálási folyamat során a funkcionális függőségek már csak a kulcstól valósulnak meg. A redundancia csökken, a relációk száma nő.

## Programozás elmélete

Programtervezésen a leendő program szerkezetének megalkotását értjük. A program elemeinek megfelelő sorrendbe rakása alapvető. A szerkezetében változók, műveletek, ciklusok, utasítás-sorok, függvények, eljárások, alprogramok találhatók. Programtervezés során az első dolgunk, hogy a problémát meghatározzuk. Ehhez fel kell mérnünk a feladatot, mit szeretnénk megvalósítani és mik lehetnek az akadályok. Ha tudjuk a feladatunkat, akkor a következő teendőnk a rendszertervezés és modellalkotás. A program struktúráját itt készítjük el vagy képzeljük el. Ha a struktúra jó, akkor azt követi a harmadik lépés, ahol a kódolás történik, azaz a programírás. A programírást követően a tesztelés következik, ahol próbáljuk megtalálni a hibákat olyan értékekkel, amik elképzelhetetlenek, de azokra is működnie kell a programnak. Ha hibákat is sikerült kijavítani, akkor az utolsó lépés a dokumentáció elkészítése. A programunknak mindenféleképen megbízhatónak kell lennie, megfelelően kell működnie. A jó terv lecsökkenti az idő- és ez által költségigényes tesztelési fázist. Egyébként a jól megtervezett program módosítása is egyszerűbb.

## Adatbázis tervezése

A weblapomhoz készítettem egy adatbázist, ehhez a symfonynak a migrációját használtam. Ebben van lehetőség adatbázisok létrehozására és mód van a hozzá szükséges táblák, valamint az azokat tartalmazó mezők generálására. Adatokkal fel tudjuk tölteni az adatbázisunkat. Az adatbázisomat „app” néven lehet elérni. Öt darab táblája van, melyek a következők: „admin”, „comment”, „dirty\_word”, „topic”, „user”. Minden egyes tábla és mező nevében használtam azt a szabályt, hogy a neve nem tartalmazhat csak angol karaktereket, számokat és pár egyéb karaktert. Mivel igyekeztem beszédes neveket választani, ezért ezek nem okoznak értelmezési nehézségeket. Így a későbbiekben - akár más által is - könnyen áttekinthető és ez által továbbfejleszthető szerkezetet készítettünk.

Az „admin” tábla azért szükséges, hogy itt tudjunk új adminisztrátorokat rögzíteni. Akik segíteni tudják a weboldal működését. Az admin felhasználók tudnak csak belépni az admin oldalra és csak azok akik aktívak. Az „admin” táblánk oszlopai a következőek:

1. **id:** Ez egy azonosító, ami egy int típusú érték (egész szám). Az értékét automatikusan növeli minden egyes új rekordnál. Elsődleges kulcs. Erre lehet hivatkozni minden egyes lekérdezésnél, amikor a „admin” táblának egy adott rekordjából információt szeretnénk kinyerni.
2. **name**: Ez szöveg mező, 255 karaktert tartalmazhat. Ez egy adott adminisztrátornak a nevét tárolja. Ez azért szükséges, hogy megtudjuk mutatni neki, hogy ő van éppen belépve az admin felületre.
3. **email**: Ez szöveg mező, 160 karaktert tartalmaz, mert hivatalosan ennél hosszabb nem lehet. Ez a adminisztrátor email címét tárolja, amivel be tud jelentkezni az admin felületre.
4. **status**: Ennek a mezőnek a típusa „int”. Ez határozza meg, hogy adminisztrátor beléphet ez az admin felületre vagy sem. Ha az értéke 0 akkor nem léphet be, amennyiben 1 akkor engedélyezzük a belépést.
5. **created\_at**: Ennek a mezőnek a típusa „datetime”. Egy adminisztrátor mikor került be a táblába.
6. **updated\_at**: Ennek a mezőnek a típusa „datetime”. Egy adminisztrátor mikor frissült az adata utoljára táblába.
7. **password**: Ennek a mezőnek a típusa „password”. Amikor ebbe a mezőbe írunk, nem látjuk, hogy milyen karaktert írtunk be, csillagok jelzik a karakterek számát. A jelszó hossza 255 karakter lehet maximum. A felhasználó jelszavát sha256 kódolással tároljuk. Ez biztosítja az admin felület védelmét.

A „comment” táblára azért van szükségünk. Itt gyűjtük a hozzászólásokat az egyes topicokhoz. Az „comment” táblánk oszlopai a következőek:

1. **id:** Ez egy azonosító, ami egy int típusú érték (egész szám). Az értékét automatikusan növeli minden egyes új rekordnál. Elsődleges kulcs. Erre lehet hivatkozni minden egyes lekérdezésnél, amikor a „admin” táblának egy adott rekordjából információt szeretnénk kinyerni.
2. **user\_id:** Egy foreign\_key, ami int típusú (egész szám), a User táblához. Ezzel van összekapcsolva a két tábla egymással. Tudni szeretnénk melyik User-nek volt ez a hozzászólása.
3. **message**: Ez szöveg mező, amely tartalmazza azt a szöveget, amely már nem tartalmaz ofcén szavakat. Az ofcén szavak csillaggal vannak jelölve. A rész egyezést is csillagozzuk.
4. **status**: Egy adott üzenet megjelenhet-e, a felületen. Moderálási lehetőség, amennyiben 1 akkor megjelenhet egyébként meg nem.
5. **created\_at**: Ennek a mezőnek a típusa „datetime”. Egy comment mikor került be a táblába.
6. **updated\_at**: Ennek a mezőnek a típusa „datetime”. Egy comment mikor frissült az adata utoljára táblába.
7. **topic\_id**: Ez egy foreign\_key, amely egy topicot azonosít. Ez mutatja meg számunkra, hogy ez az üzenet melyik topic-hoz tartozik.
8. **original**: Az eredeti szöveg, amely az ofcén szavakat is tartalmaz.

A „dirty\_word” tábla. Ebben a táblában az ofcén szavakat használjuk, amelyeket letöltöttünk xml formában és feltöltöttünk az adatbázisban. Ebben az xml-ben 2 érték szerepelt egy típus és maga a szó. A „dirty\_word” tábla oszlopai a következők:

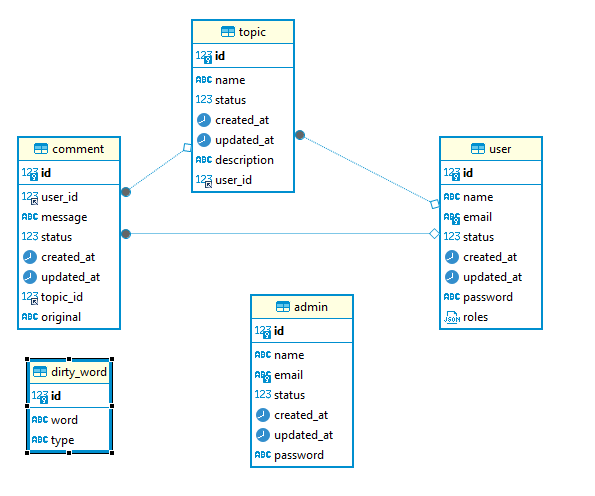
1. **id**: Ez egy azonosító, ami egy int típusú érték (egész szám). Az értékét automatikusan növeli minden egyes új rekordnál. Elsődleges kulcs. Erre lehet hivatkozni minden egyes lekérdezésnél, amikor a „admin” táblának egy adott rekordjából információt szeretnénk kinyerni.
2. **word**: varchar típusú oszlop, amelynek a hossza 255 karakter lehet. Ez tartalmazza konkrétan a trágár szót. Ezeket a szavakat kell kiszűrni a megjelenésből.
3. **type**: varchar típusú oszlop, amely „f” és „m” betűket tartalmazhat kizárólag.

A „topic” táblára azért van szükségünk. Itt gyűjtük a topic-kokat, amelyekhez hozzászólhatnak a felhasználók. Az „topic” táblánk oszlopai a következőek:

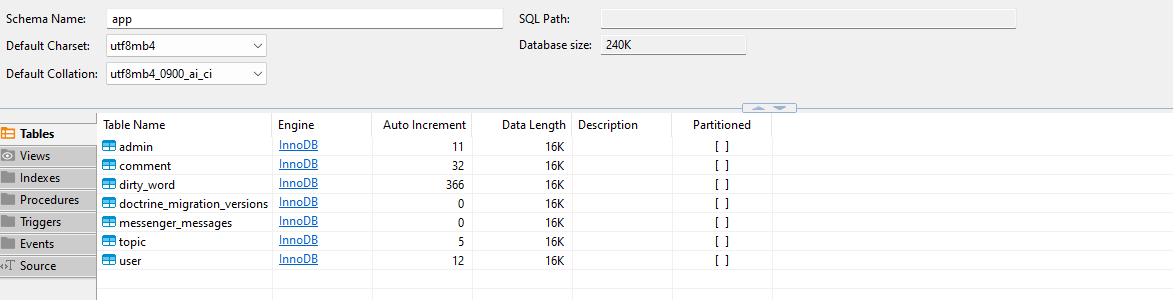
1. **id:** Ez egy azonosító, ami egy int típusú érték (egész szám). Az értékét automatikusan növeli minden egyes új rekordnál. Elsődleges kulcs. Erre lehet hivatkozni minden egyes lekérdezésnél, amikor a „admin” táblának egy adott rekordjából információt szeretnénk kinyerni.
2. **user\_id:** Egy foreign\_key, ami int típusú (egész szám), a User táblához. Ezzel van összekapcsolva a két tábla egymással. Tudni szeretnénk melyik User hozta létre.
3. **name**: Ez szöveg mező, amely tartalmazza azt a szöveget, amely meghatározza a topic témáját.
4. **status**: Egy adott topic megjelenhet-e, a felületen. Moderálási lehetőség, amennyiben 1 akkor megjelenhet egyébként meg nem.
5. **created\_at**: Ennek a mezőnek a típusa „datetime”. Egy topic mikor került be a táblába.
6. **updated\_at**: Ennek a mezőnek a típusa „datetime”. Egy topic mikor frissült az adata utoljára táblába.
7. **description**: A topic leírása, hogy miről kell szólnia, ha név (cím) nem lenne egyértelmű.

Az „user” tábla azért szükséges, hogy itt tudjunk új felhasználókat rögzíteni. Akik ténylegesen fogják felhasználni a weboldalt. Ők tudnak új topicokat létrehozni és a topicokon belül kommunikálni másokkal. Az „user” táblánk oszlopai a következőek:

1. **id:** Ez egy azonosító, ami egy int típusú érték (egész szám). Az értékét automatikusan növeli minden egyes új rekordnál. Elsődleges kulcs. Erre lehet hivatkozni minden egyes lekérdezésnél, amikor a „admin” táblának egy adott rekordjából információt szeretnénk kinyerni.
2. **name**: Ez szöveg mező, 255 karaktert tartalmazhat. Ez egy adott felhasználó a nevét tárolja. Ez azért szükséges, hogy megtudjuk mutatni neki, hogy ő van éppen belépve a weboldalon.
3. **email**: Ez szöveg mező, 160 karaktert tartalmaz, mert hivatalosan ennél hosszabb nem lehet. Ez a felhasználó email címét tárolja, amivel be tud jelentkezni az admin felületre.
4. **status**: Ennek a mezőnek a típusa „int”. Ez határozza meg, hogy felhasználó beléphet-e a weboldalra vagy sem. Ha az értéke 0 akkor nem léphet be, amennyiben 1 akkor engedélyezzük a belépést.
5. **created\_at**: Ennek a mezőnek a típusa „datetime”. Egy felhasználó mikor került be a táblába.
6. **updated\_at**: Ennek a mezőnek a típusa „datetime”. Egy felhasználó mikor frissült az adata utoljára táblába.
7. **password**: Ennek a mezőnek a típusa „password”. Amikor ebbe a mezőbe írunk, nem látjuk, hogy milyen karaktert írtunk be, csillagok jelzik a karakterek számát. A jelszó hossza 255 karakter lehet maximum. A felhasználó jelszavát sha256 kódolással tároljuk. Ez biztosítja az weboldal felület védelmét.



1. **ábra: Harmadik normálforma (forrás: saját ábra)**

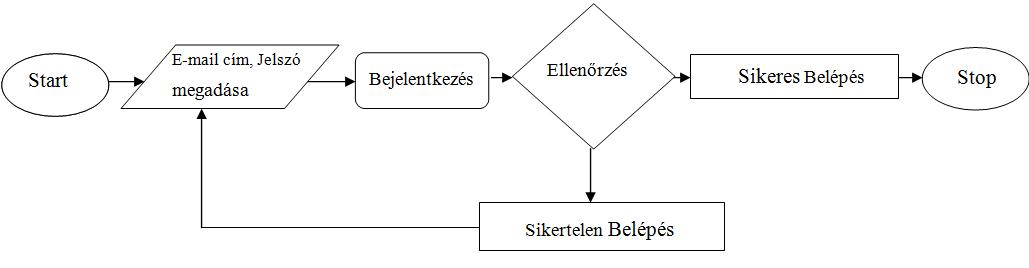


1. **ábra: Adatbázistábla (forrás: saját ábra)**

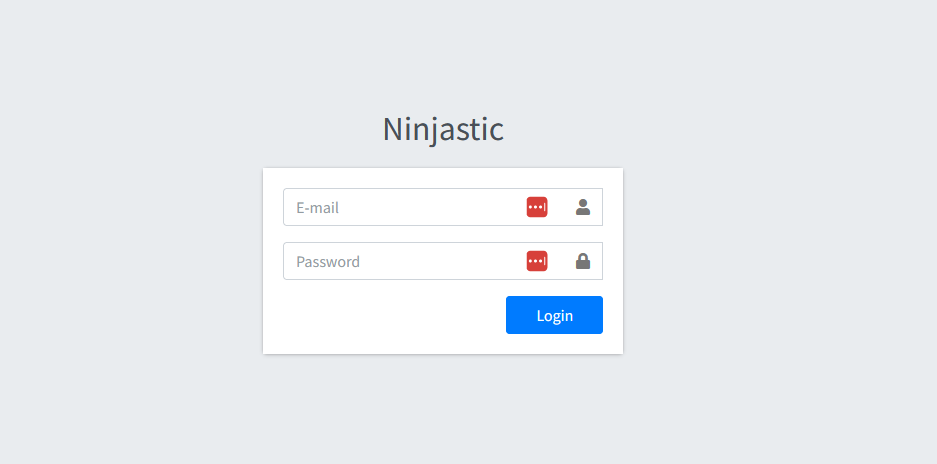
# Admin felület

# Bejelentkezés

A felületre domain után a /admin utvonalon érhető el. Ahol az admin email cím és jelszó párossal van lehetőségünk bejelentkezni, amelyet ha sikeresen megadunk akkor egy új felületen találjuk magunkat, a dashboardon. Ha helytelenül akkor hiba üzeneteket tapasztalhatunk.

****

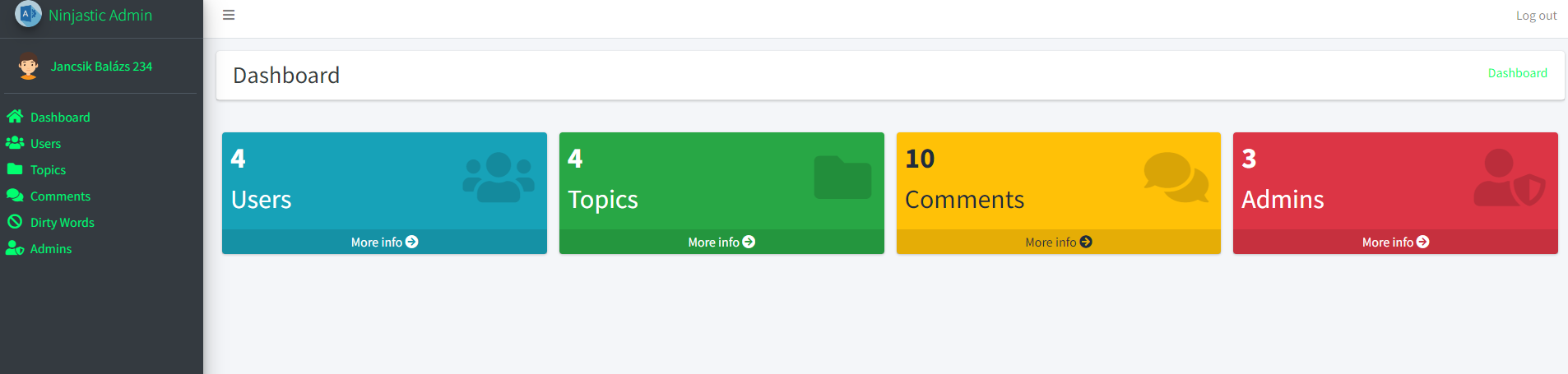
1. **ábra: Belépési folyamatábra (forrás: saját ábra)**



1. **ábra: Bejelentkezési lap (forrás: saját ábra)**

Sikeres bejelentkezés esetén a felhasználó adatait SESSION-ben tároljuk le, amíg ott megtalálható addig az admin bejelentkezve marad.

## Dashboard



1. **ábra: Dashboard (forrás: saját ábra)**

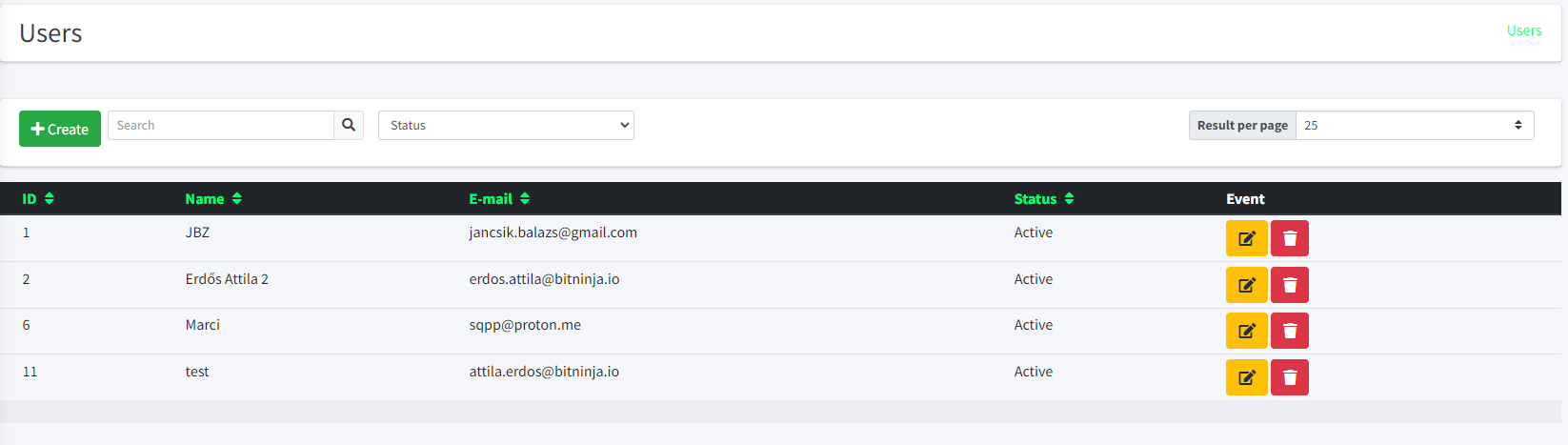
Ez egy rövid át tekintő, egyből az adminisztrátor számára, ahol egyből láthatja, hogy hány regisztrált felhasználó van éppen, hány topikot hoztak már létre, hány hozzászólás történt már összesen és hány admin felhasználó van regisztrálva.

Baloldalon található egy menü, amelyen menü pontokat találunk.

A jobb felső sarkoban található a „Log out” 🡪 Kijelentkezés. Amelyre kattintást követően kijelentkezünk, a SESSION-t töröljük és visszatérünk a Bejelentkezési felületre.

## Felhasználók

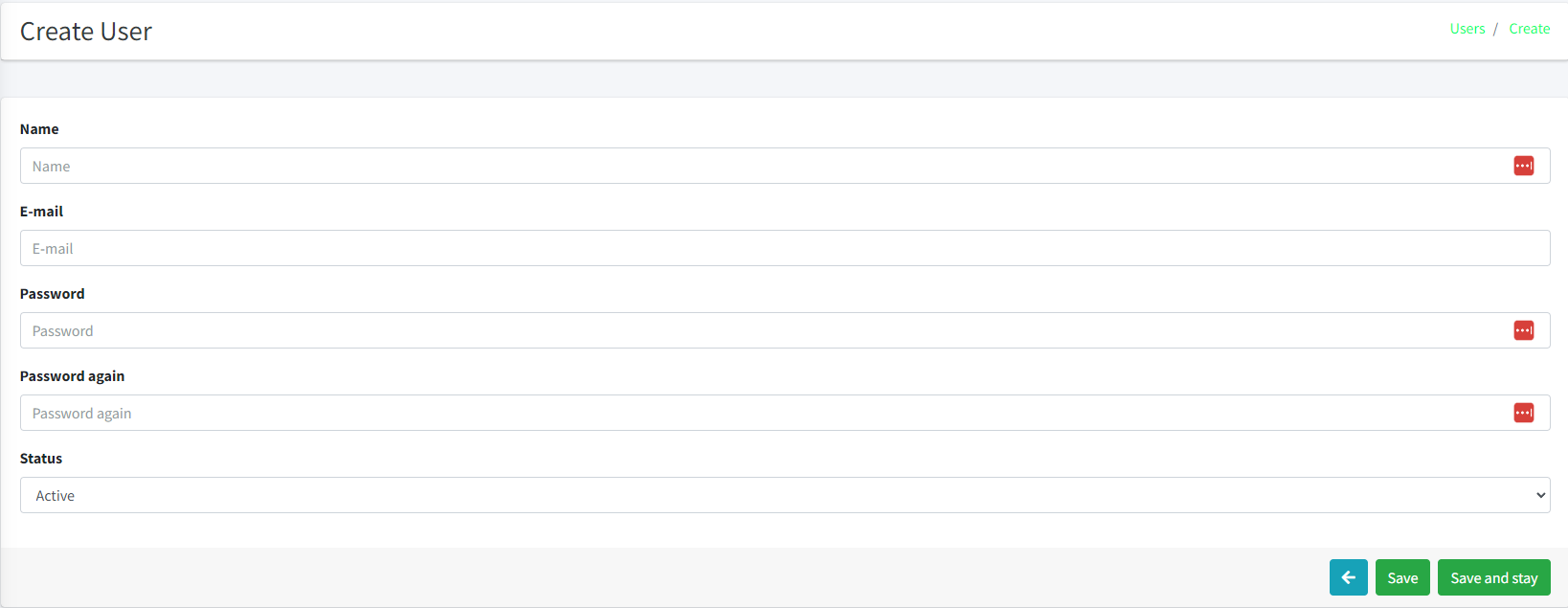
### Listázás



1. **ábra: Users lista oldal (forrás: saját ábra)**

A felhasználókat találjuk ezen a felületen. A create gomb megnyomása esetén tudunk új felhasználót létrehozni. Keresésre van lehetőség amely a névben és az e-mail címben tud keresni. A status választó meg egy filter. Ezen kívül a táblázat jobb oldalán van lehetőségünk állítani, hogy hány elem jelenhet meg egyszerre a felületen. Users-zel egysorban található zöld betűvel kisebben szintén ez a breadcrumb. Amely, ha mélybere megyünk létrehozás vagy szerkesztésre akkor lehet róla vissza navigálni.

### Létrehozás/Szerkesztés



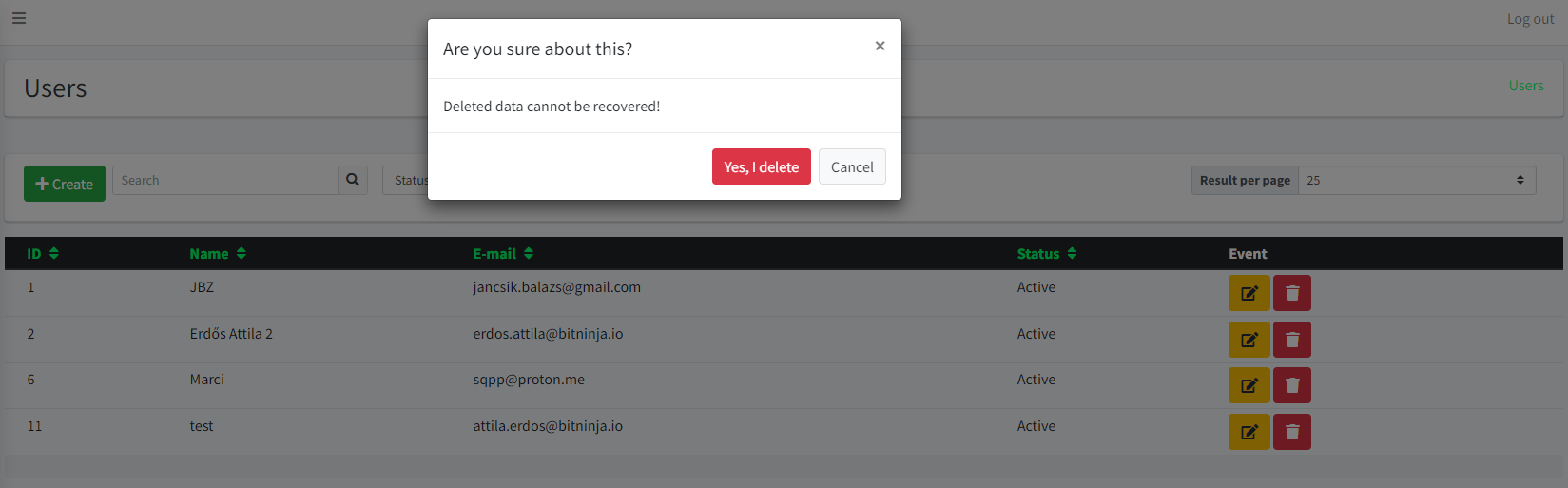
1. **ábra: Users létrehozás/szerkesztés oldal (forrás: saját ábra)**

Ezen felületen lehet létrehozni vagy szerkeszteni egy felhasználót. A felhasználó mentése esetén amely a „save” vagy „save and stay” gomb megnyomásával történik. A „save” gomb használata esetén visszanavigálunk a listázási oldalra. A „save and stay” megnyomása esetén maradunk ezen a felület, amennyiben szeretnénk folytatni a felhasználó adatainak a módosítását. A vissza nyilacska gomb mentés nélkül visszanavigál a listázási oldalra. Az adatok mielőtt a mentés megtörténne ellenőrzés történik, amelyek a következők.

1. „Name”: Nem lehet üres mező
2. „Email”: E-mail formátumnak megfelelőnek kell lennie.
3. „Password”: Minimum 8 karakter hosszúnak kell lennie és password again input mezővel megegyező tartalommal kell bírnia.

Amennyiben egy email cím már használva van, arról hiba üzenetet kapunk.

### Törlés

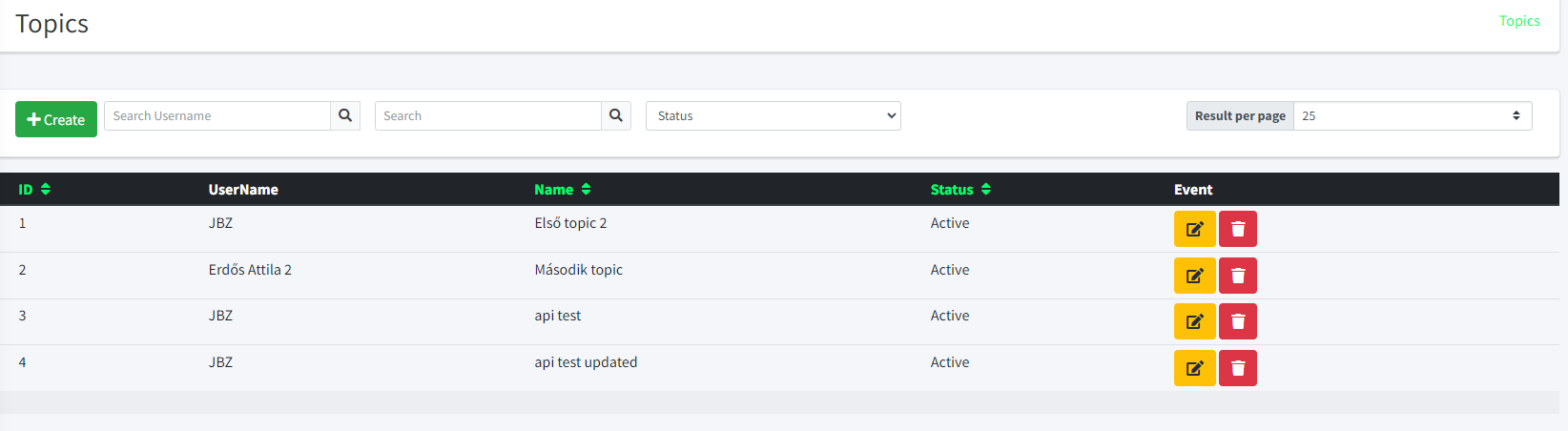


1. **ábra: Users törlés popup (forrás: saját ábra)**

Törlés esetén mindig egy felugró ablak jelenik meg, amely egy megerősítés ellenében végrehajta a műveletet.

## Topikok

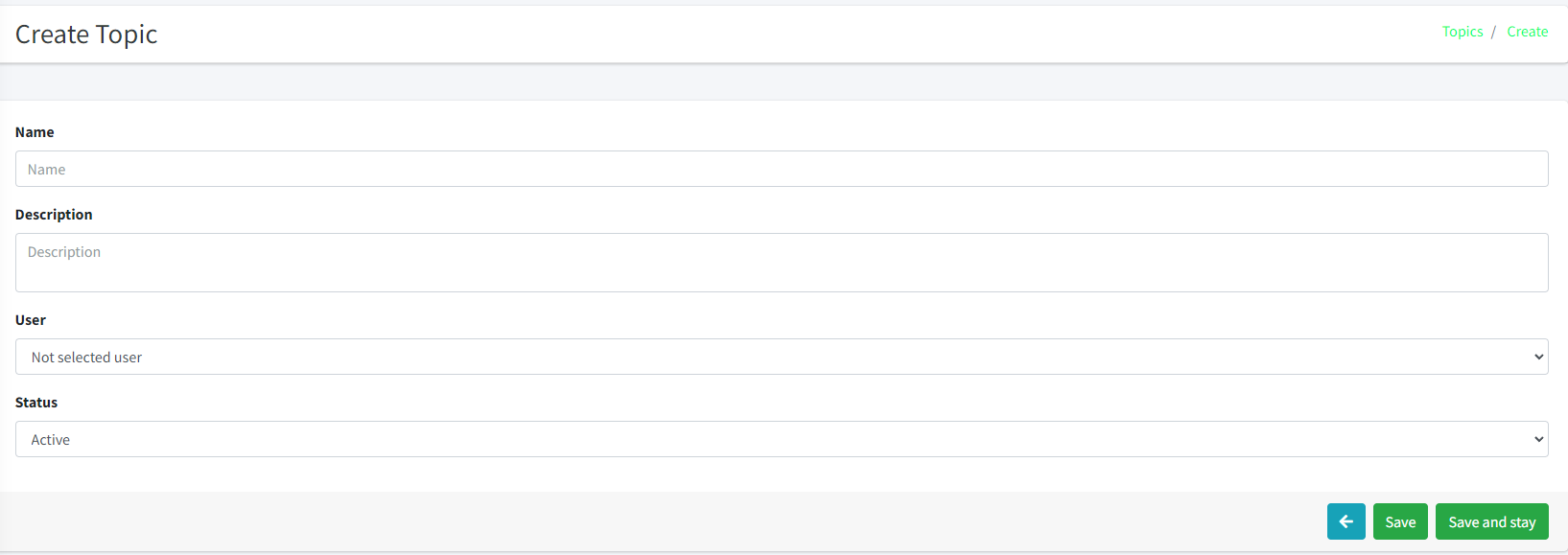
### Listázás



1. **ábra: Topics lista oldal (forrás: saját ábra)**

A topikokat találjuk ezen a felületen. A create gomb megnyomása esetén tudunk új topikot létrehozni. „Search Username” placeholder mutatja nekünk azt, hogy abban az input mezőben kizárólag a felhasználó nevére lehet keresni.  
„Search” placeholder mutatja nekünk azt, hogy keresni tudunk a topik nevére.  
A „Status” választó meg egy filter. Ezen kívül a táblázat jobb oldalán van lehetőségünk állítani, hogy hány elem jelenhet meg egyszerre a felületen.

### Létrehozás/Szerkesztés

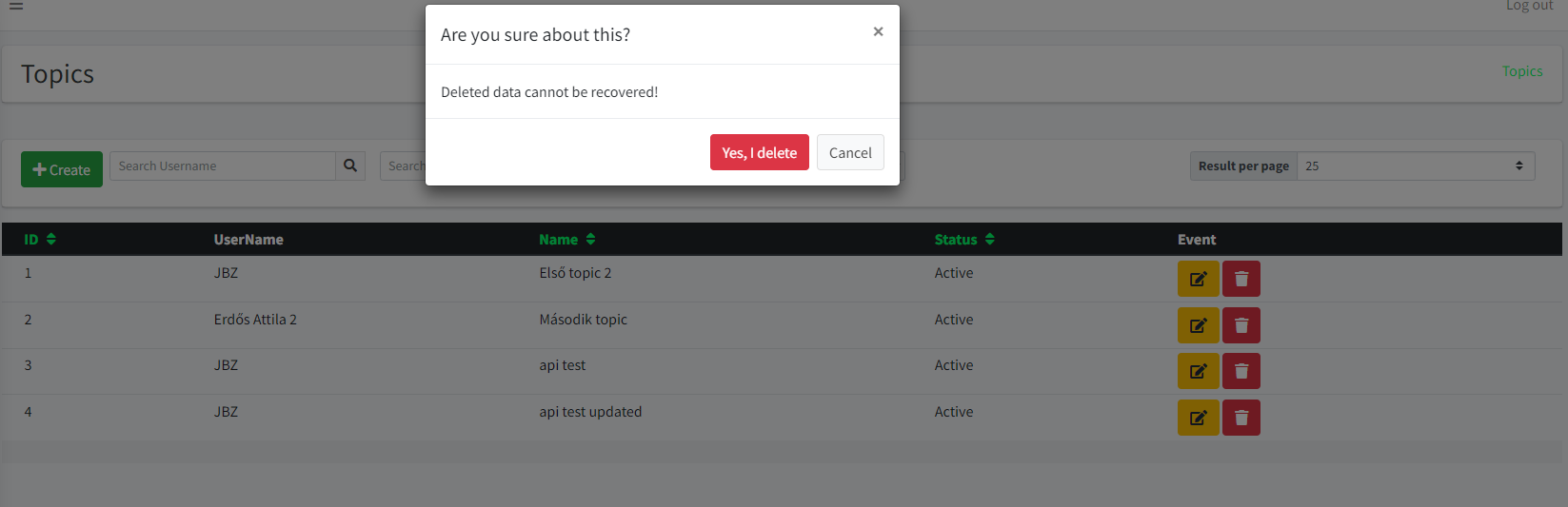


1. **ábra: Topics létrehozás/szerkesztés oldal (forrás: saját ábra)**

Ezen felületen lehet létrehozni vagy szerkeszteni egy topikot. A topik mentése esetén amely a „save” vagy „save and stay” gomb megnyomásával történik. A „save” gomb használata esetén visszanavigálunk a listázási oldalra. A „save and stay” megnyomása esetén maradunk ezen a felület, amennyiben szeretnénk folytatni a topik adatainak a módosítását. A vissza nyilacska gomb mentés nélkül visszanavigál a listázási oldalra. Az adatok mielőtt a mentés megtörténne ellenőrzés történik, amelyek a következők.

1. „Name”: Nem lehet üres mező
2. „Description”: Nem lehet üres mező
3. „User”: Kötelező felhasználót hozzárendelni egy topikhoz

### Törlés

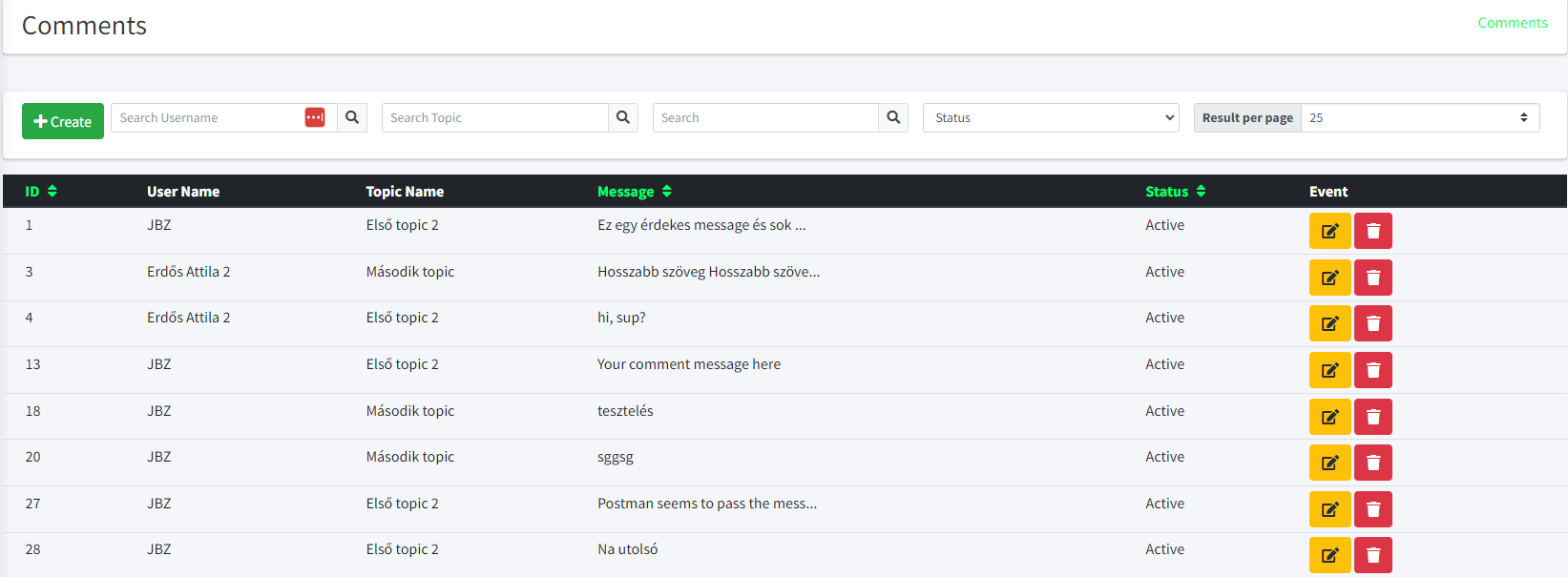


1. **ábra: Topics törlési oldal (forrás: saját ábra)**

Törlés esetén mindig egy felugró ablak jelenik meg, amely egy megerősítés ellenében végrehajta a műveletet.

## Hozzászólások

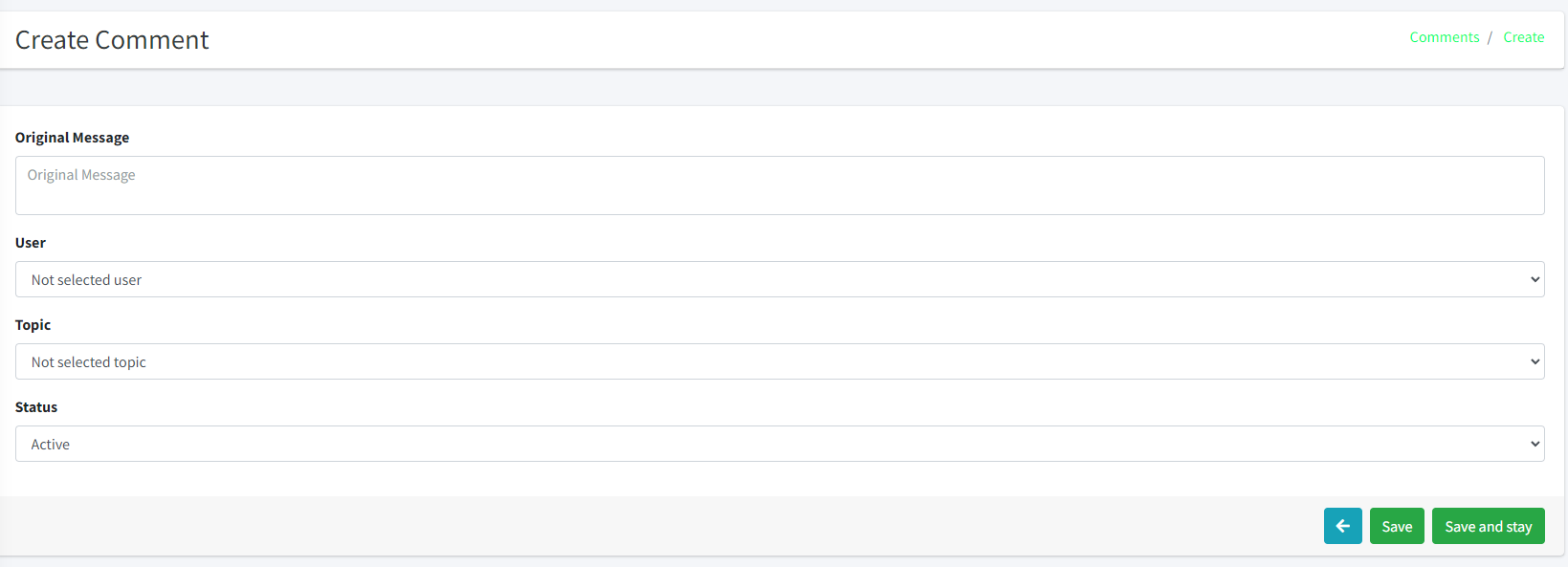
### Listázás



1. **ábra: Comments lista oldal (forrás: saját ábra)**

A hozzászólasokat találjuk ezen a felületen. A create gomb megnyomása esetén tudunk új hosszászólást létrehozni.   
„Search Username” placeholder mutatja nekünk azt, hogy abban az input mezőben kizárólag a felhasználó nevére lehet keresni.  
„Search Topic” placeholder mutatja nekünk, hogy a topic nevére tudunk keresni.  
„Search” placeholder mutatja nekünk azt, hogy keresni tudunk az üzenet szövegében.  
A „Status” választó meg egy filter. Ezen kívül a táblázat jobb oldalán van lehetőségünk állítani, hogy hány elem jelenhet meg egyszerre a felületen.

### Létrehozás / Szerkesztés

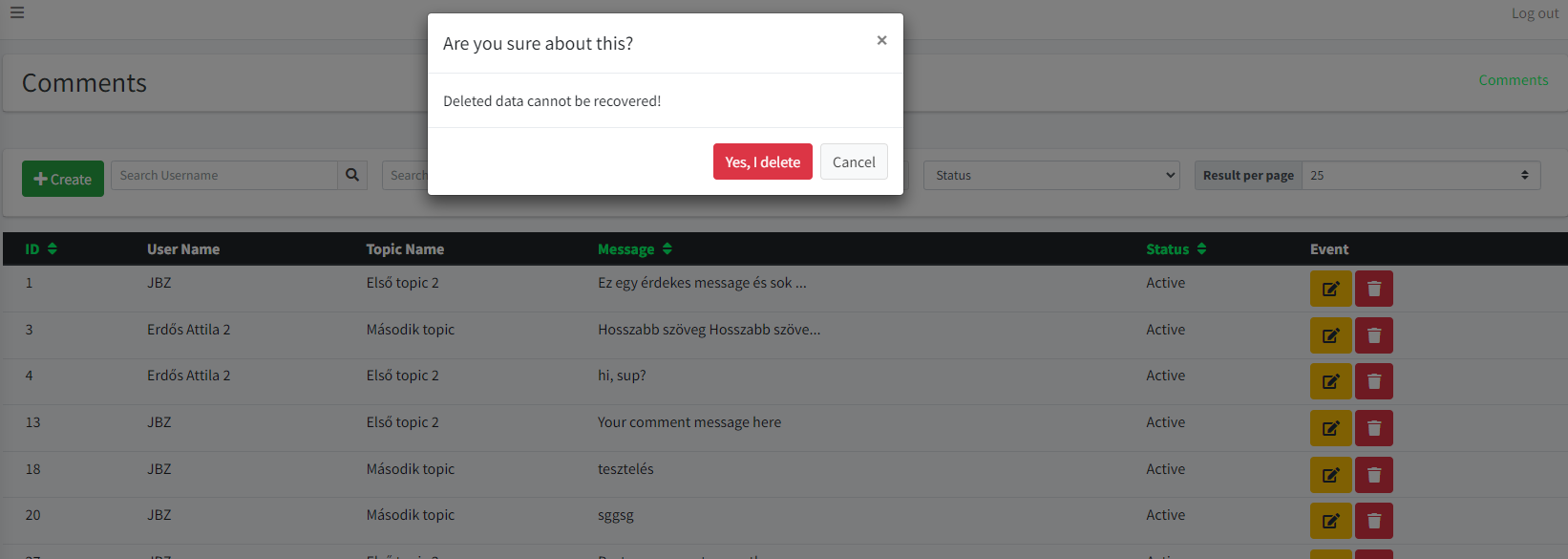


1. **ábra: Comments létrehozás/szerkesztés oldal (forrás: saját ábra)**

Ezen felületen lehet létrehozni vagy szerkeszteni egy hozzászólást. A hozzászólás mentése esetén amely a „save” vagy „save and stay” gomb megnyomásával történik. A „save” gomb használata esetén visszanavigálunk a listázási oldalra. A „save and stay” megnyomása esetén maradunk ezen a felület, amennyiben szeretnénk folytatni a hozzászólás adatainak a módosítását. A vissza nyilacska gomb mentés nélkül visszanavigál a listázási oldalra. Az adatok mielőtt a mentés megtörténne ellenőrzés történik, amelyek a következők.

1. „Original”: Nem lehet üres mező
2. „User”: Kötelező felhasználót hozzárendelni egy hozzászóláshoz
3. „Topic”: Kötelező Topicot hozzárendelni egy hozzászóláshoz

### Törlés

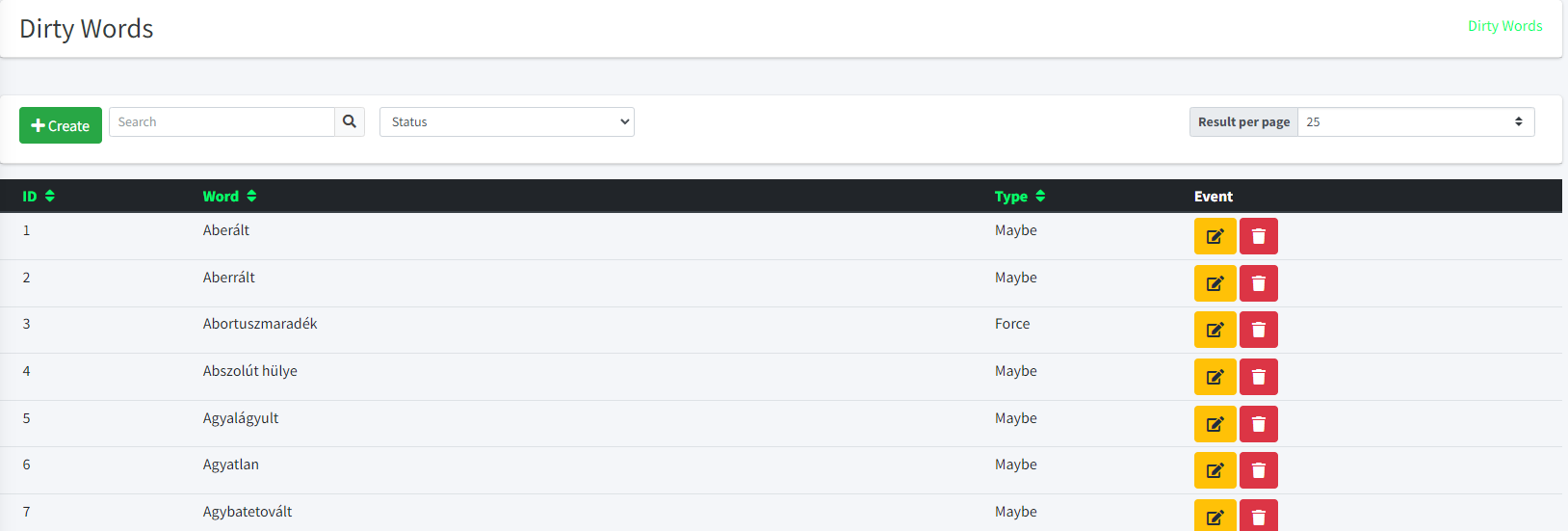


1. **ábra: Comments törlés oldal (forrás: saját ábra)**

Törlés esetén mindig egy felugró ablak jelenik meg, amely egy megerősítés ellenében végrehajta a műveletet.

## Csúnya szavak

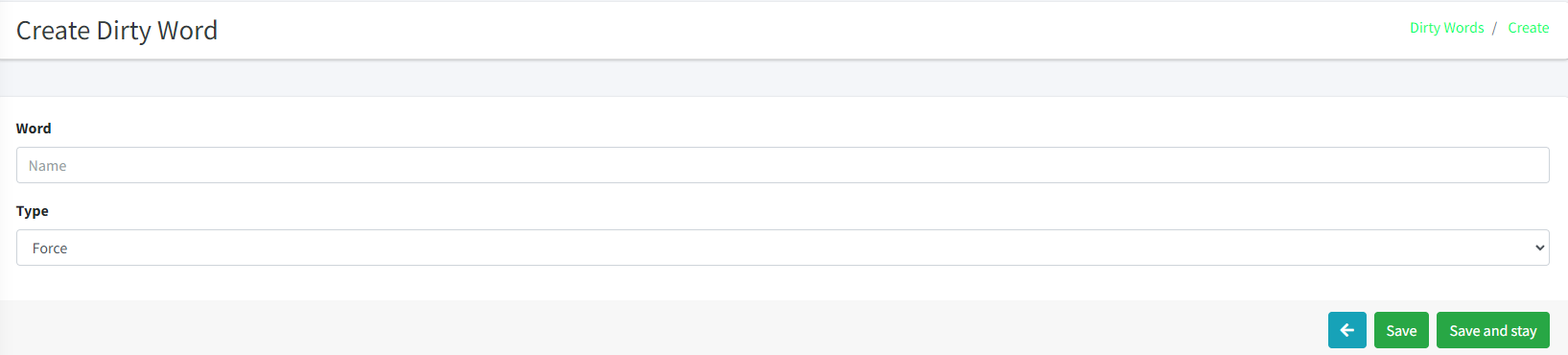
### Listázás



1. **ábra: Dirty Words lista oldal (forrás: saját ábra)**

A ofcén szavak listáját találjuk ezen a felületen. A create gomb megnyomása esetén tudunk új ofcén szót létrehozni.   
„Search” placeholder mutatja nekünk azt, hogy keresni tudunk trágár szóra.  
A „Status” választó meg egy filter. Ezen kívül a táblázat jobb oldalán van lehetőségünk állítani, hogy hány elem jelenhet meg egyszerre a felületen.

### Létrehozás / Szerkesztés

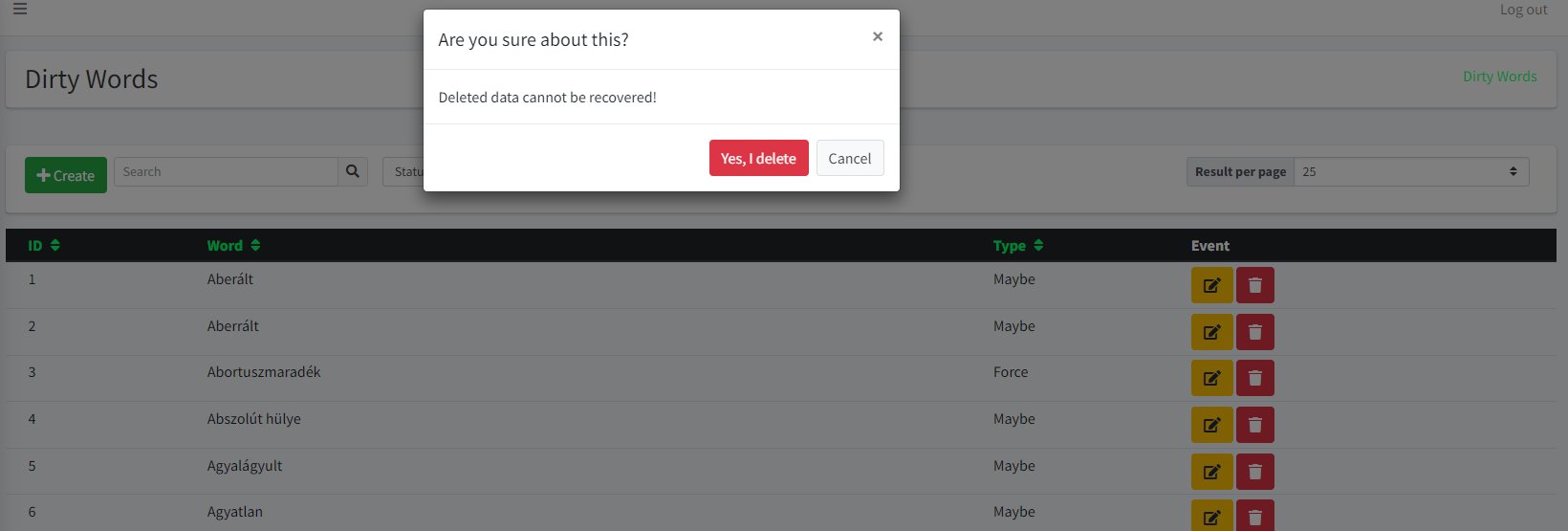


1. **ábra: Dirty Words létrehozás/szerkesztés oldal (forrás: saját ábra)**

Ezen felületen lehet létrehozni vagy szerkeszteni egy csúnya szót. A csúnya szó mentése esetén amely a „save” vagy „save and stay” gomb megnyomásával történik. A „save” gomb használata esetén visszanavigálunk a listázási oldalra. A „save and stay” megnyomása esetén maradunk ezen a felület, amennyiben szeretnénk folytatni a csúnya szó adatainak a módosítását. A vissza nyilacska gomb mentés nélkül visszanavigál a listázási oldalra. Az adatok mielőtt a mentés megtörténne ellenőrzés történik, amelyek a következők.

1. „Word”: Nem lehet üres mező
2. „Type”: Nem lehet üres mező, kötelező választani valamit

### Törlés

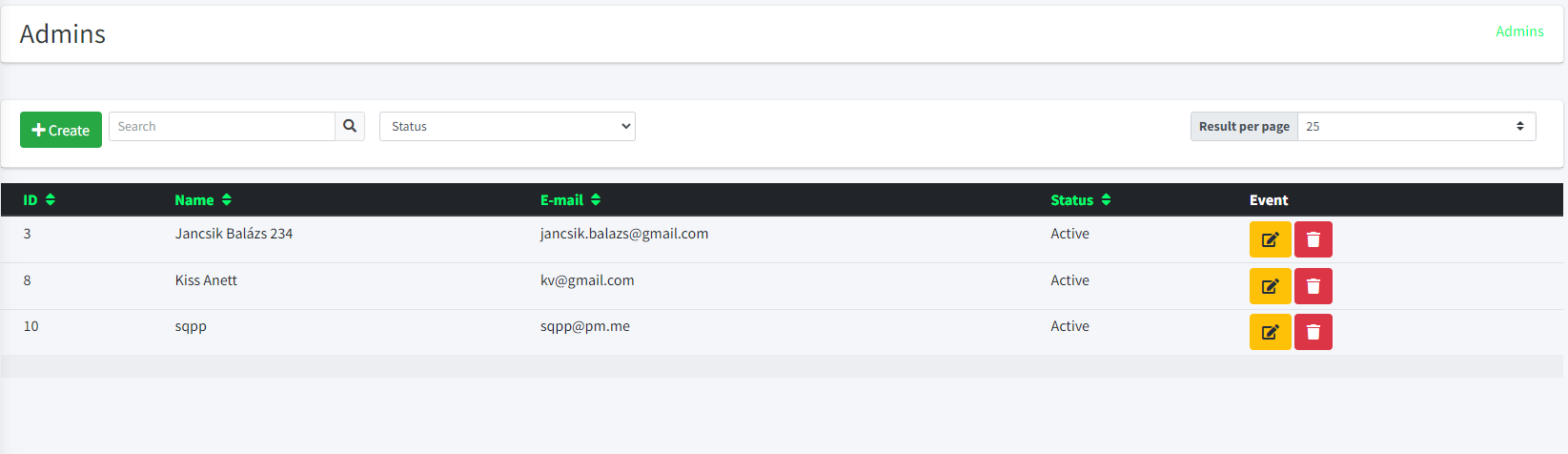


1. **ábra: Dirty Words törlés oldal (forrás: saját ábra)**

Törlés esetén mindig egy felugró ablak jelenik meg, amely egy megerősítés ellenében végrehajta a műveletet.

## Adminok

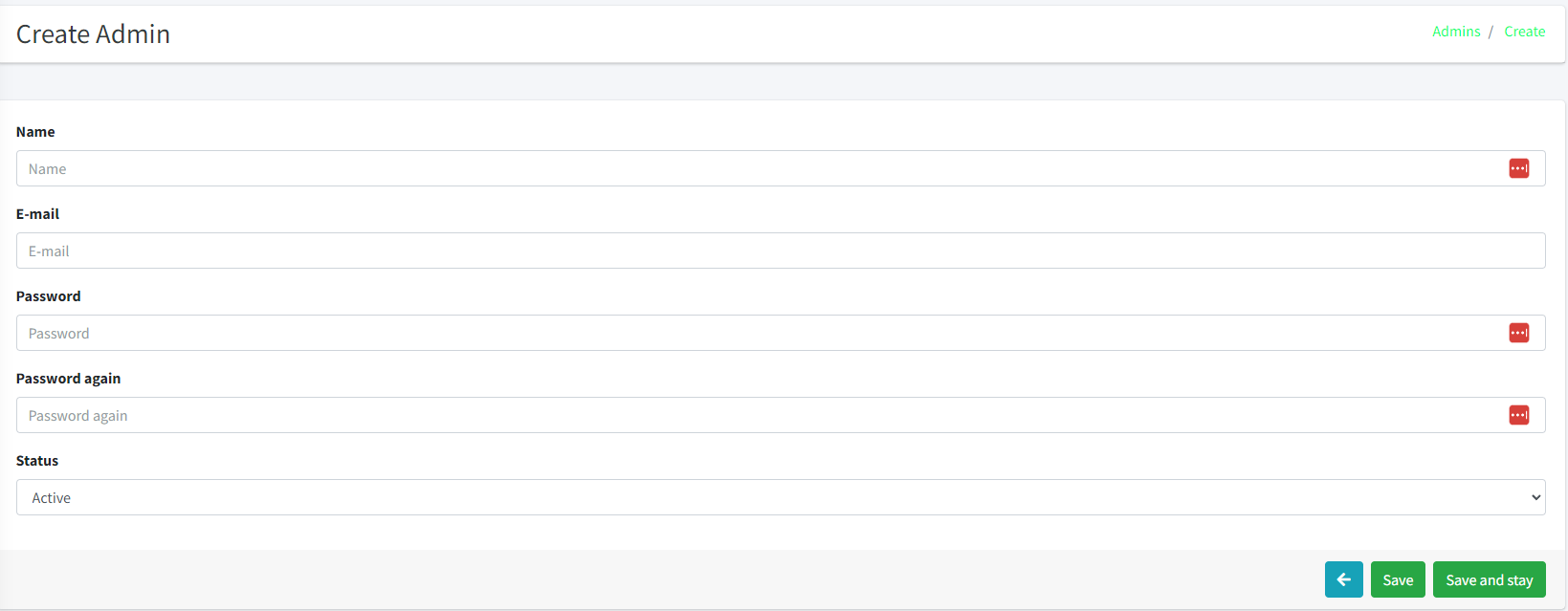
### Listázás



1. **ábra: Admins lista oldal (forrás: saját ábra)**

Az adminok listáját találjuk ezen a felületen. A create gomb megnyomása esetén tudunk új admint létrehozni.   
„Search” placeholder mutatja nekünk azt, hogy keresni név és e-mail cím alapján.  
A „Status” választó meg egy filter. Ezen kívül a táblázat jobb oldalán van lehetőségünk állítani, hogy hány elem jelenhet meg egyszerre a felületen.

### Létrehozás / Szerkesztés



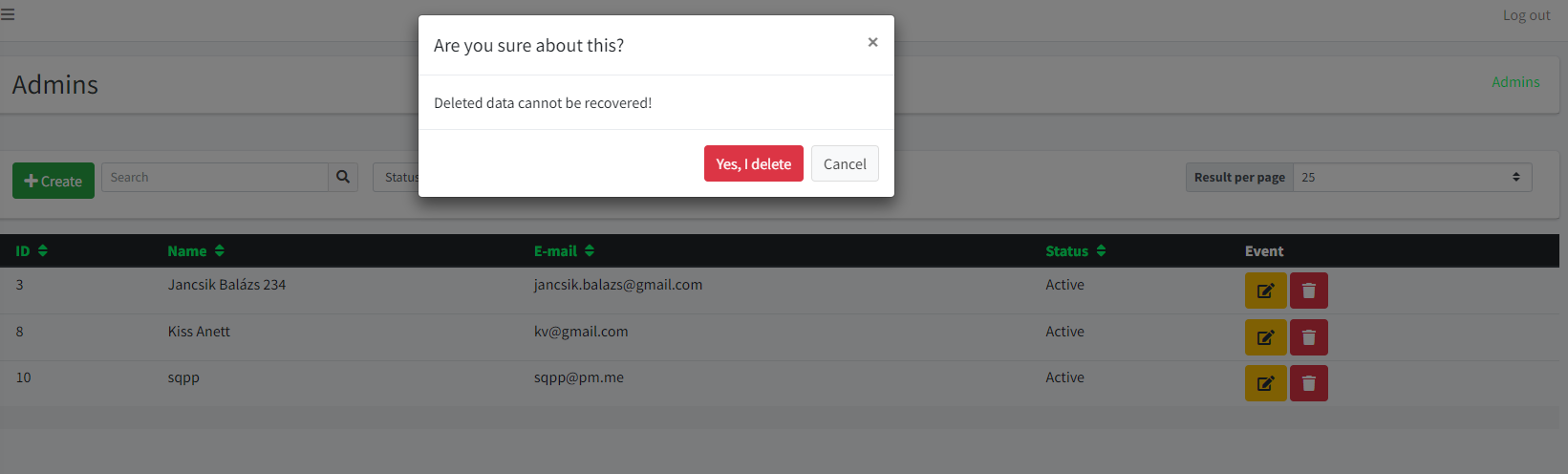
1. **ábra: Admins lista oldal (forrás: saját ábra)**

Ezen felületen lehet létrehozni vagy szerkeszteni egy admint. Az admin mentése esetén amely a „save” vagy „save and stay” gomb megnyomásával történik. A „save” gomb használata esetén visszanavigálunk a listázási oldalra. A „save and stay” megnyomása esetén maradunk ezen a felület, amennyiben szeretnénk folytatni az admin adatainak a módosítását. A vissza nyilacska gomb mentés nélkül visszanavigál a listázási oldalra. Az adatok mielőtt a mentés megtörténne ellenőrzés történik, amelyek a következők.

1. „Name”: Nem lehet üres mező
2. „Email”: E-mail formátumnak megfelelőnek kell lennie.
3. „Password”: Minimum 8 karakter hosszúnak kell lennie és password again input mezővel megegyező tartalommal kell bírnia.

Amennyiben egy email cím már használva van, arról hiba üzenetet kapunk.

### Törlés



1. **ábra: Admins lista oldal (forrás: saját ábra)**

Törlés esetén mindig egy felugró ablak jelenik meg, amely egy megerősítés ellenében végrehajta a műveletet.