# Bugzilla Light

## Adatbázisok kötelező feladat

### Feladat

A program feladata hibabejelentések nyomon követése.

A bejelentéseket felhasználók fogják létrehozni. A felhasználó felhasználónév és jelszó párossal képes csak bejelentkezni. Kötelező megadnia az e-mail címét, valamint a telefonszámát, mint elérhetőség. A jelenlegi verzióban az e-mail hitelesítése, valamint az elfelejtett jelszó logikája nincsen implementálva, az adminisztrátor feladata ezeket kezelni.

Bejelentés létrehozása felhasználói jogokkal is elfogadott, azonban minden bejelentést egyedileg kell hogy elbíráljon egy, a bejelentéshez hozzárendelt felhasználónak (továbbiakban: kinevezett). A kinevezéshez megfelelő jogosultságokkal kell rendelkezni.

A kinevezést a felhasználó maga is elvégezheti (megfelelő jogosultságokkal), valamint a nála magasabb jogosultsággal rendelkező személy is hozzárendelheti.

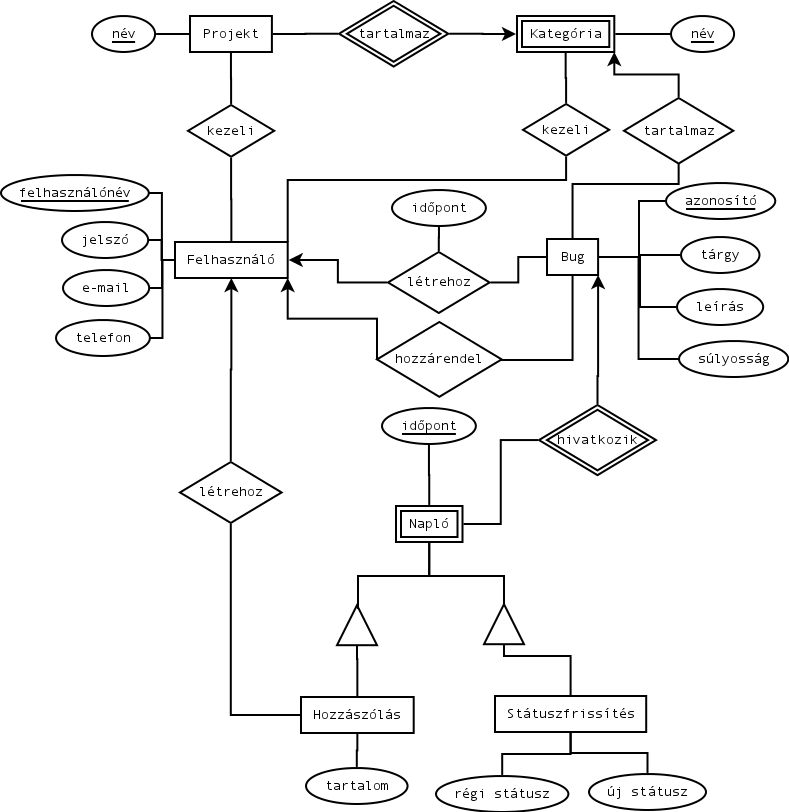
A rendszerben megtalálható egy superuser (admin), aki minden joggal rendelkezik, ezt tőle senki és semmi nem vonhatja meg. A bejelentések először projektek szerint vannak csoportosítva. Ezeket a projekteket a superuser által kinevezett felhasználók kezelik. A következő csoportosítási szint az egyes projekteken belüli kategória, amiknek kezelőit a projektkezelők, valamint a superuser nevezheti ki. A kezelő jogosultság az adott csoportosítási szint alatt elérhető minden beállítást módosíthat.

A bug-ok kategorizálva vannak a rendszerben, minden bug egy kategóriához van rendelve. A kategorizálás a bejelentést létrehozó felhasználó feladata, azonban a kinevezettnek lehetősége van ezen módosítani. A kinevezett feladata a hibabejelentés életciklusának kezelése.

Először a hiba nyitott állapotba kerül. Hozzárendelés után hozzárendelt státuszba lép. Innentől kezdve egy hiba életciklusa egyedi. Amennyiben a kinevezettnek valamilyen kérdése van a felhasználó felé, úgy a hiba ’válaszra vár’ státuszba kerül. Ha a hibabejelentés téves volt, úgy a bug lezárt státuszba kerül. Amennyiben a hibabejelentés jogos volt, úgy a státusz először ’folyamatban’, majd ’megoldott’ státuszba kerül.

Minden hibabejelentés rendelkezik egy létrehozó és egy kinevezett felhasználóval. A hibabejelentés kategóriákba van rendezve, valamint van tárgya és leírása. Ezeken kívül van súlyossága, amit a hozzárendelt felhasználó kezel. 6 súlyossági szint van: kritikus, magas, közepes, alacsony, nagyon alacsony.Minden hibabejelentéshez tartozik egy napló, aminek két specializált változata van. A hozzászólásnak van tartalma, és felhasználója. Minden felhasználó képes hozzászólni a bejelentéshez. A napló egy másik specializált változata a státuszfrissítés, aminek egy régi, valamint egy új státusz mezője van.

### Egyed-kapcsolat modell



### Relációs adatbázisséma

* Projekt (név)
* Kategória (*projekt.név*, név)
* Felhasználó (felhasználónév, jelszó, e-mail, telefon)
* Projektkezelés (*projekt.név*, *felhasználó*.*felhasználónév*)
* Kategóriakezelés (*kategória.név*, *felhasználó*.*felhasználónév*)
* Bug (azonosító, tárgy, leírás, súlyosság, *felhasználó.felhasználónév, kategória*)
* (Napló entitás jelen esetben nem fontos, mivel csak bonyolítaná a lekérdezéseket)
* Hozzászólás (*időpont*, *bug.azonosító*, tartalom, *felhasználó.felhasználónév*)
* Státuszfrissítés (*időpont*, *bug.azonosító*, régi\_státusz, új\_státusz)

### Normalizálás

#### 1NF

A relációséma 1NF-ben van, mert minden eleme atomi.

#### 2NF

A séma 2NF-ben van, mivel a relációkban minden másodlagos attribútum teljesen függ az elsődleges kulcsoktól.

#### 3NF

A séma 3NF-ben van, mert minden másodlagos attribútuma közvetlenül függ bármely kulcstól.

#### Végleges megoldás

* Projekt (név)
* Kategória (*projekt.név*, név)
* Felhasználó (felhasználónév, jelszó, e-mail, telefon)
* Projektkezelés (*projekt.név*, *felhasználó*.*felhasználónév*)
* Kategóriakezelés (*kategória.név*, *felhasználó*.*felhasználónév*)
* Bug (azonosító, tárgy, leírás, súlyosság **f***elhasználó.felhasználónév, kategória*)
* (Napló entitás jelen esetben nem fontos, mivel csak bonyolítaná a lekérdezéseket)
* Hozzászólás (*időpont*, *bug.azonosító*, tartalom, *felhasználó.felhasználónév*)
* Státuszfrissítés (*időpont*, *bug.azonosító*, régi\_státusz, új\_státusz)

### Megvalósítási környezet

A rendszer PHP nyelvet használ (7.2 verzió), adatbáziskapcsolatokhoz PHP Data Objectet használok. Az adatbázis MySQL alapú lesz. A diagramok PHPGraphLib-el lesznek megvalósítva.Frontend téren Bootstrap-el lesz megvalósítva a layout.

### Program szolgáltatásai

#### Alapfunkciók

* Bejelentkezés/Regisztráció
* Bug létrehozása
* Bug állapotának módosítása
* Felhasználói adatok módosítása

#### Extra funkciók

* Mock data PHP Fakerrel
* Felhasználói jogosultságok módosítása
* Statisztikai kimutatások

### Három nemtriviális lekérdezés

1. Hozzászólások és státuszfrissítések összevonása, valamint sorredbe rendezése MySQL UNION paranccsal
   * Eredményül egy időrend szerint sorbarendezett listát kapunk, a bug életciklusának megjelenítéséhez elengedhetetlen.
2. Napi bontásban egy projekt kategória szerinti aktivitása
   * Egy „diagramot” ad eredményül, ami segítségül szolgál például refactor esetében. (látható, hogy melyik kategóriában van a legtöbb bug)
3. Súlyosság alapján egy bug átlagos élettartama (amennyiben a legkorábbi státuszfrissítés nyitott, valamint a legkésőbbi megoldva állapotú)
   * Ez egy súlyosság, időtartam (másodperc) párossal tér vissza. Ezzel a kimutatással meg lehet nézni, hogy a fejlesztőcsapat mennyire veszi komolyan egy hiba fontosságát (a hipotézis az, hogy a súlyosság nagysága fordítottan arányos az időtartam hosszával)