# Mi a Bugzilla/Testopia?

Az önkormányzati ASP projektben a konzorciumi tagok döntse alapján a tesztelések tervezésére, végrehajtásának támogatására, és a tesztelés eredményeinek dokumentálására a Bugzilla terméket használja a Testopia modullal.

A Bugzilla lényegében nem más, mint egy hibajegy-kezelő rendszer, hasonló a PMSZ kapcsán már sokak számára ismert Mantis-hoz. A lényege, hogy hibajegyeket lehet benne rögzíteni, ezeket a hibajegyeket hozzá lehet rendelni előre definiált termékekhez, komponensekhez, rendszerverziókhoz és környezetekhez, ki lehet osztani felelősöknek, meg lehet határozni a súlyosságukat az előre definiált értékek kiválasztásával, illetve meg lehet adni a javítás sürgősségét.

Különösen a komplex informatikai rendszerek esetében, illetve a finanszírozás elszámolásához nagyon fontos azonban a leszállítandó projekttermékek – rendszerek – tervezett és jól dokumentált vizsgálata nem csak funkcionális megfelelőség, hanem sok egyéb szempontból is. Kiemelten fontos a részletes megvalósíthatósági tanulmányban, a pályázat során, a támogatási szerződésben, az ajánlati felhívásokban, illetve a szállítói szerződésekben rögzített követelményeknek való megfelelés igazolása, valamint sok esetben a vagyonkezelői jogok átadása kapcsán, illetve az üzemeltetésre átadás során a megfelelőségek igazolása. Éppen ezért a teszteléseket ezeknek a szempontoknak a figyelembe vételével is tervezni kell.

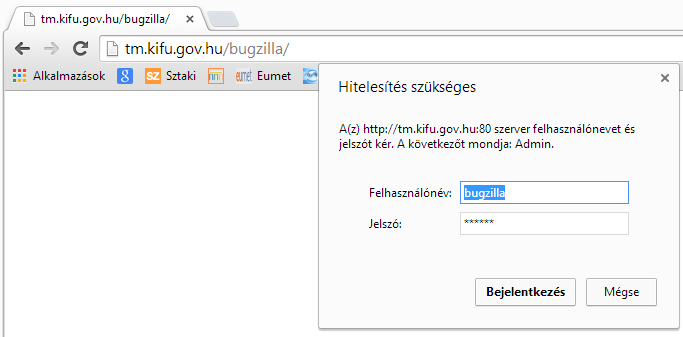
A Bugzilla hibajegy-kezelő rendszere ezért kiegészítésre került a Testopia előtéttel. A Testopia teszi lehetővé a rögzített projekttermékekhez teszttervek, tesztesetek, esetfuttatások rögzítését egy sor kiegészítő információval együtt. Ezek a lehetőségek biztosítják, hogy a teszteléseket a követelmények figyelembe vételével akár elemi lépések szintjén megtervezzük, az egyes lépések futtatását különböző bemeneti adatokkal különböző tesztelőkre bízzuk, és a futtatások eredményeit rögzítsük – sikeresként, vagy egy hibajegyben. A projektben a megfelelő szakemberek részvételével tehát előre megtervezzük a teljes tesztelési folyamatot és tartalmat, és ezt feladatkiosztás szintjéig előre rögzítjük a Testopia rendszerben, ahol aztán az eredmények is rögzítésre kerülnek. A rögzített eredményekből természetesen mindenféle lekérdezést, statisztikát lehet készíteni.

A KIFÜ Bugzilla/Testopia rendszerét úgy állítottuk be az ASP teszteléshez, hogy az leképezze az ASP projekt tesztelési stratégiáját.

# Elérés, belépés

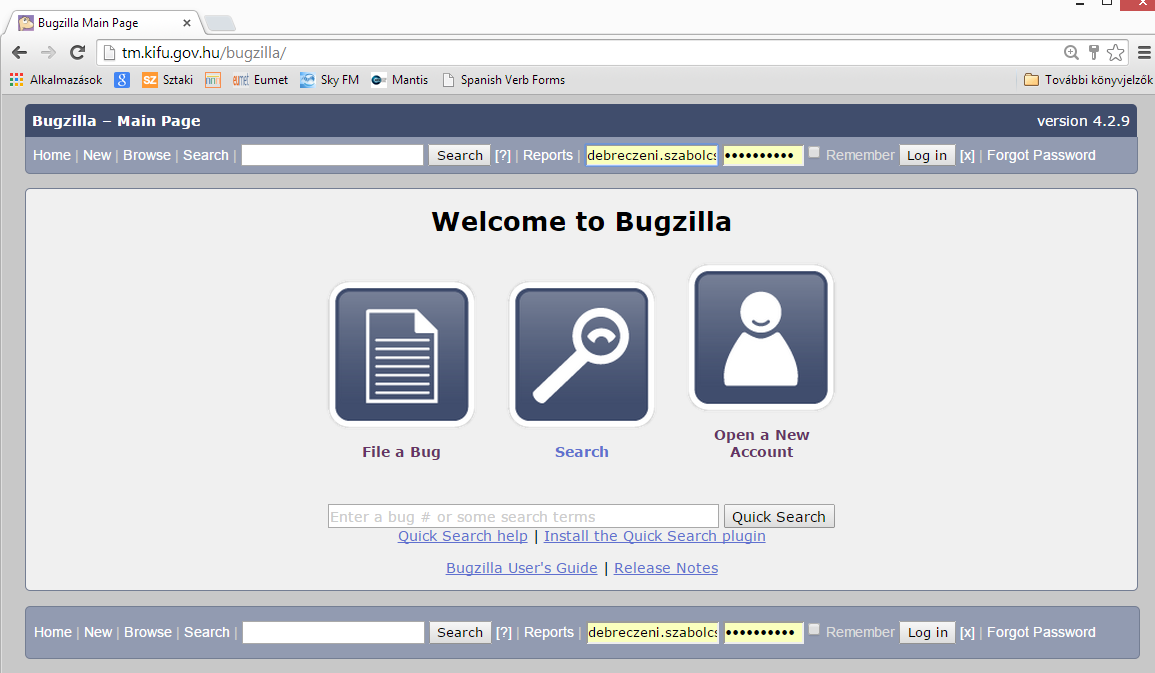
Az ASP projekttermékek tesztelésének tervezésére és dokumentálására felállított Testopia/Bugzilla az Interneten keresztül elérhető. Egy böngészőben (lehetőleg a Chrome vagy a Firefox) kell ide navigálni: <http://tm.kifu.gov.hu/bugzilla>. Az elérés kettős védelemmel ellátott; egyrészt be kell lépni a megfelelő kiszolgálóra, majd ezt követően magába a Testopiába.

Belépés a kiszolgálóra:



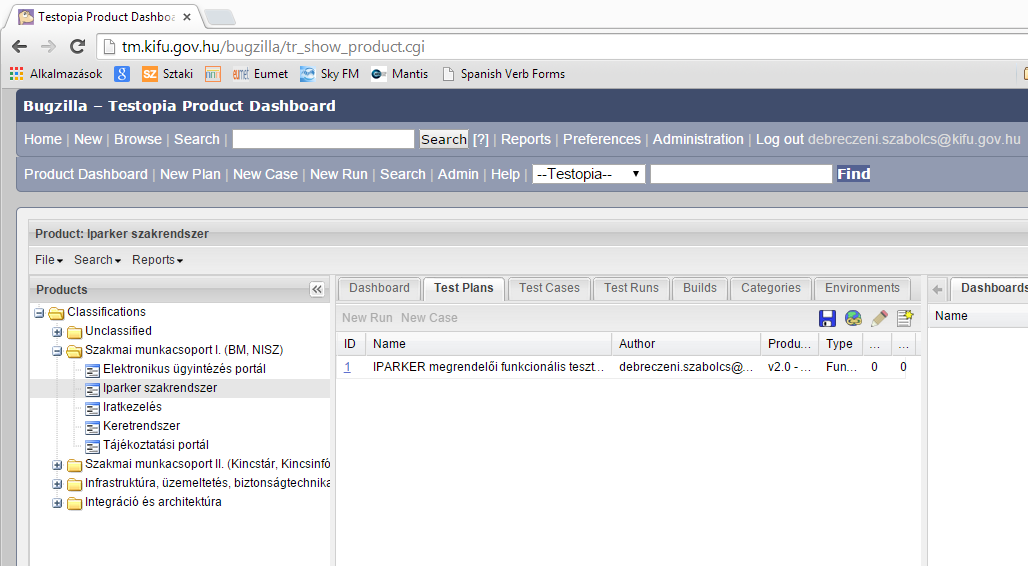
A felhasználónevet és jelszót mindenkinek külön megküldjük. Ez a felhasználónév-jelszó páros mindenki számára ugyanaz.

A kiszolgálóra történő belépést követően megjelenik a Testopia/Bugzilla nyitó képernyője, ahol be kell lépni mindenkinek a saját felhasználónevével és jelszavával:



A felhasználónév mindenkinek a saját, hivatalos, intézményi e-mail címe; a jelszavakat külön megküldjük. A felhasználónevet a Login mezőbe, a jelszót a Password mezőbe kell beírni, majd a Log in gombra kattintani.

Belépés után a nyitóoldalon a Product Dashboard gombbal indítható az az áttekintő felület, amelyen mindenki – a jogosultságainak megfelelően – láthatja a termékeit, végrehajtandó teszteseteit, stb.



# Tesztelés tervezés és végrehajtás, dokumentálás a Testopiában

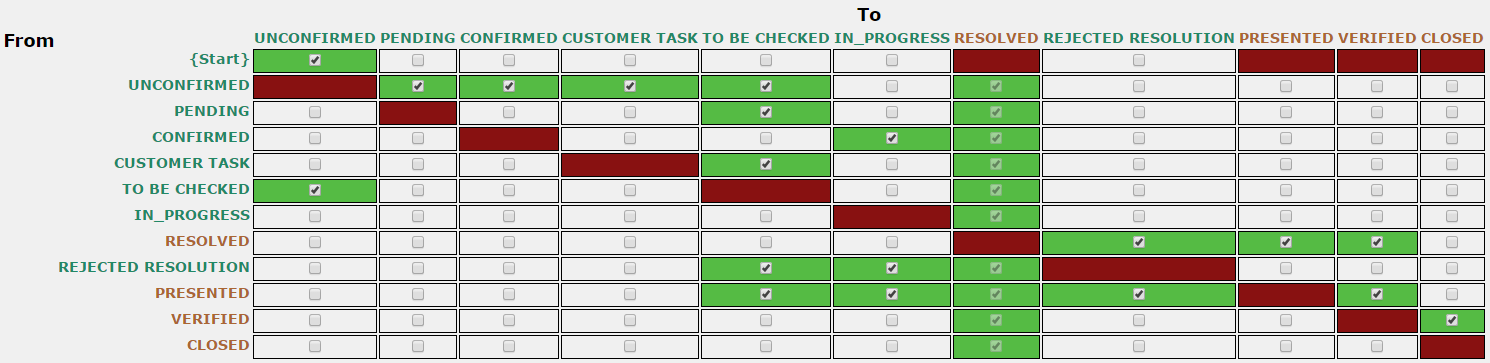
A Testopia tehát arra való, hogy a tesztelések részletes terveit rögzíthessük, nyilvántarthassuk, és a tesztelések lefolytatását – a hibajegyekkel együtt – rögzíthessük. Ez azért hangsúlyos, mert így biztosítható, hogy az egyes projekttermékek tesztelése az elkészült dokumentációk – szállítói teszttervek, tesztesetek és jegyzőkönyvek, rendszertervek, felhasználói kézikönyvek, egyéb követelményjegyzékek – alapján, teljességre törekedve előre megtervezésre kerülhessenek, és a tesztelések végrehajtása, a hibajegyek rögzítése illeszkedhessen ezekhez. A megrendelő oldali tesztek egyértelmű feladata annak vizsgálata, hogy az átadásra kerülő rendszer minden követelménynek és tervnek megfelel-e; ezért van szükség az előzetes tesztelési tervezésre.

Az ASP Testopia rendszerét a működéshez szükséges, projekt specifikus alapadatokkal már feltöltöttük. A Testopia alapvetően a leszállítandó és tesztelendő projekttermékek (Products) köré szerveződik. A projekttermékek lényegében egyes rendszereket, illetve a legmagasabb szintű tesztelendő entitásokat jelölnek. A teszteléshez kapcsolódó terveket konkrét termékhez, Product-hoz kell kötni. A termékeket szabadon csoportosíthatjuk (Classification), ez segíti az eligazodást a nyitóoldalon. A termékekhez további alapvető információkat is kell rögzíteni, amelyek nélkül nem lehet tervdokumentumokat rögzíteni: termékverzió, termék komponens, teszttípus, stb. Ezeknek az alapadatoknak az adminisztrálása a tesztelési munkacsoport vezetésének a feladata.

Az ASP tesztelési stratégiájában a tesztelés tervezéséhez maga a tesztstratégia, a teszttervek, tesztesetek, teszt forgatókönyvek és teszt csomagok tartoznak, mint dokumentumtípusok. A tesztelések végrehajtásának dokumentálására a hibajegyek és a teszt jegyzőkönyvek szolgálnak. A tesztelést támogató eszköznek – esetünkben a Testopiának – nyilván ehhez kell igazodnia. Ennek megfelelően a tesztstratégiából kiindulva a Testopia a következő dokumentációs egységeket tartalmazza:

| **Tesztstratégia alapján** | **Célja, tartalma** | **Testopiában** | **Magyarázat, eltérés** |
| --- | --- | --- | --- |
| Teszt terv | feladat-és ütemterv, felelősök, feltételek, követelmények | - | A Testopiában ezt a tervezési szintet közvetlenül nem rögzítjük. |
| Teszteset | tesztelendő rendszerelemek, funkciók, folyamatok; érintett követelmények, logikai szcenáriók | Test Case | A Testopiában a tesztelés alapvető lépéseit és elvárt eredményeit a Test Case-ben rögzítjük. A Test Case-hez annyi Test Run-t rögzítünk, ahányszor – akár különböző adatokkal – a Test Case végrehajtásra kerül. Adott Test Case adott lefuttatásának konkrét eredménye a Test Run-ban kerül dokumentálásra. Sikertelen Test Run-hoz hibajegyet, Bug-ot rögzítünk. |
| Tesztforgatókönyv | megfelelőség ellenőrzése hogyan, milyen lépések alapján, milyen adatokkal, milyen elvárt eredményekkel | Test Case + Test Run |
| Teszt csomag | mely teszteseteket és tesztforgatókönyveket milyen ütemezés mentén kinek kell végrehajtania | Test Plan | A Testopiában a Test Case-eket csak Test Plan-hoz rendelve lehet rögzíteni. Ezek alkalmasak arra, hogy megtestesítsék a stratégia teszt csomag elemét. |
| Hibajegy | az előforduló hibák pontos, a javításhoz elegendő információinak rögzítése, súlyosság, prioritás | Bug | A Testopiában Bug-ot kizárólag Test Run-hoz kapcsolva lehet létrehozni. A Bug megfogalmazásának és a hozzá rögzítendő információknak megadott szabályai vannak. |
| Teszt jegyzőkönyv | a tesztelés eredményeinek időszakos, illetve tartalmi egységenkénti összefoglalása | Test Run + Bug | Adott időszak vagy tesztelési ciklus eredményeit a Testopiában a hozzá tartozó Test Run-ok eredményeinek lekérdezésével, illetve a kapcsolódó hibajegyek listázásával lehet dokumentálni, jegyzőkönyvezni. Ez a fajta lekérdezés lehet az alapja egy hivatalos, szöveges teszt jegyzőkönyvnek. |

A hibajegyek státuszai külön munkafolyamat-követést, workflow-t képeznek. A lényege, hogy az egyes hibajegyek állapotát leíró Bug Status mező értéke csak megadott szabályok mentén módosítható. Ezeket a szabályokat foglalja össze az alábbi táblázat, illetve alatta a részletes leírás:



UNCONFIRMED -> Ebben a státuszban jelenti be a tesztelő.

* Ha azért tűnik hibának, mert még nincsenek meg a működés feltételei (nincs kész) -> PENDING
* Ha a fórum megerősíti, hogy hiba -> CONFIRMED
* Ha azért tűnik hibának, mert megrendelői oldalon még tenni kell valamit (pl. adathiány) -> CUSTOMER TASK
* Ha a hiba leírása nem megfelelő, vagy a tesztelés nem megfelelő, ezért újra kell tesztelni -> TO BE CHECKED
* Ha már kijavítva: RESOLVED + FIXED
* Ha korábban már másik hibajegyben bejelentették: RESOLVED + DUPLICATE
* Ha a szállító nem fogja valamiért kijavítani a hibát -> RESOLVED + WONTFIX
* Ha valójában nem hiba -> RESOLVED + INVALID

Ha még nincs kész, valójában nem tesztelhető: PENDING

* Ha a körülmények előálltak és újra kell tesztelni -> TO BE CHECKED
* Ha a körülmények előállása javította -> RESOLVED + FIXED
* Ha korábban már másik hibajegyben bejelentették: RESOLVED + DUPLICATE
* Ha a szállító nem fogja valamiért kijavítani a hibát -> RESOLVED + WONTFIX
* Ha valójában nem hiba -> RESOLVED + INVALID

Normál hibajavítás folyamata: CONFIRMED

* Ha a szállító elkezdett rajta dolgozni -> IN PROGRESS
* Ha elkészült a javítás -> RESOLVED + FIXED
* Ha korábban már másik hibajegyben bejelentették: RESOLVED + DUPLICATE
* Ha a szállító nem fogja valamiért kijavítani a hibát -> RESOLVED + WONTFIX
* Ha valójában nem hiba -> RESOLVED + INVALID

Ha megrendelői feladat: CUSTOMER TASK

* Ha a megrendelő oldal megcsinálta, és újra kell tesztelni -> TO BE CHECKED
* Ha a megrendelő megcsinálta, és a hiba nem áll fenn -> RESOLVED + FIXED
* Ha korábban már másik hibajegyben bejelentették: RESOLVED + DUPLICATE
* Ha a szállító nem fogja valamiért kijavítani a hibát -> RESOLVED + WONTFIX
* Ha valójában nem hiba -> RESOLVED + INVALID

Ha újra kell tesztelni: TO BE CHECKED

* Ha hibás működés -> UNCONFIRMED
* Ha a hiba nem áll fenn -> RESOLVED + FIXED
* Ha korábban már másik hibajegyben bejelentették: RESOLVED + DUPLICATE
* Ha a szállító nem fogja valamiért kijavítani a hibát -> RESOLVED + WONTFIX
* Ha valójában nem hiba -> RESOLVED + INVALID

Ha a szállító megkezdte a hibajavítást: IN PROGRESS

* Ha elkészült a javítás -> RESOLVED + FIXED
* Ha korábban már másik hibajegyben bejelentették: RESOLVED + DUPLICATE
* Ha a szállító nem fogja valamiért kijavítani a hibát -> RESOLVED + WONTFIX
* Ha valójában nem hiba -> RESOLVED + INVALID

Ha a hibára van valamilyen megoldás: RESOLVED

* Ha a megoldást bemutatta a szállító -> PRESENTED
* Ha a megoldást elutasítjuk: REJECTED RESOLUTION
* Ha a megoldást elfogadjuk: VERIFIED + az az alstátusz (FIXED, DUPLICATE, WONTFIX, INVALID), amelyikben a RESOLVED eredetileg is volt

Ha a megoldást elutasítjuk: REJECTED RESOLUTION

* Ha a szállító visszaadja tesztelésre: TO BE CHECKED
* Ha a szállító megkezdi a javítást: IN PROGRESS
* Ha a megrendelő megcsinálta, és a hiba nem áll fenn -> RESOLVED + FIXED
* Ha korábban már másik hibajegyben bejelentették: RESOLVED + DUPLICATE
* Ha a szállító nem fogja valamiért kijavítani a hibát -> RESOLVED + WONTFIX
* Ha valójában nem hiba -> RESOLVED + INVALID

Ha a megoldást a szállító bemutatta: PRESENTED

* Ha a megoldást elutasítjuk: REJECTED RESOLUTION
* Ha a megoldást elfogadjuk: VERIFIED + az az alstátusz (FIXED, DUPLICATE, WONTFIX, INVALID), amelyikben a RESOLVED eredetileg is volt

Ha a megoldást elfogadtuk: VERIFIED

* Ha a továbbiakban már nem kell a hibajeggyel foglalkozni: CLOSED