Multidiszciplináris tudományok, 10. kötet. (2020) 2 sz. pp. 514-518 **https://doi.org/10.35925/j.multi.2020.2.58**

TESZTSEGÉD ALKALMAZÁS

Szűcs Miklós
mesteroktató, Miskolci Egyetem, Általános Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: szucs@iit.uni-miskolc.hu

Fejti Martin
egyetemi hallgató, Miskolci Egyetem, Általános Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: fejtimartin98@gmail.com

Absztrakt

A szoftverfejlesztés folyamatának egyik igen fontos lépése a tesztelés. A teszteléskor felmerülő problémák nem feltétlenül a tudás, a szakértelem hiányából erednek. Sok esetben a tesztelés eredményét nehéz értelmezni, feldolgozni. A tesztelési információk programmal történő nyilvántartása, elemzése megoldást ad a tesztelés sikeres elvégzésére. Ezen cikk célja egy ilyen alkalmazás megtervezése, a fejlesztéséhez szükséges technológiák ismertetése, a fejlesztés kivitelezése, és a tapasztalatok leírása. Az elkészülő programnak képesnek kell lennie teszteseteket létrehoznia, szükség esetén ezeken módosításokat végeznie, törölnie. A véglegesnek tekinthető teszteseteket a programnak képesnek kell lennie futtatni, vagyis elvégezni a tényleges tesztelést. A teszt eredményeit tárolnia kell, ezeket az adatokat meg kell jelenítenie, és egyéb funkciókkal is segítenie kell a tesztelési folyamatot, pl. hibaüzenetek naplózása, vagy statisztikai adatok diagrammal történő megjelenítése.

Kulcsszavak: *tesztelés, teszteset, naplózás*

Abstract

One very important step in the software development process is testing. Problems that arise during testing do not necessarily stem from a lack of knowledge or expertise. In many cases, the results of testing are difficult to interpret and process. Recording and analyzing the test information with a program provides a solution for the successful completion of testing. The purpose of this article is to design an application, describe the technologies required for its development, carry out the development, and describe the experience. The completed program must be able to create test cases, make changes and delete them if necessary. The program must be able to run the test cases that can be considered final, and perform the actual testing. You should store the test results, display this data, and use other functions to assist the testing process, e.g. logging error messages or charting statistics.

Keywords: *testing, test case, logging*

1. Bevezető

A szoftverfejlesztés folyamatának egyik legfontosabb lépése a tesztelés. Ezzel, vagy hasonló tartalmú kijelentéssel olyan sokszor találkozunk az informatika világában, hogy sokszor már csak közhelyként tekintünk rá, elcsépeltnek tartjuk. Egy program fejlesztése során viszont előbb-utóbb eljutunk arra a pontra, amikor megtapasztaljuk ennek a kijelentésnek a súlyát. Különösen abban az esetben, ha nem csupán egy egyszerű programot írunk otthon, hanem akár egy egyetemi beadandó feladatot készítünk,

vagy akár gyakornoki teendőinket végezzük egy vállalatnál. Nagyon kellemetlen lehet, ha éles helyzetben hasal el az alkalmazásunk, mert nem volt megfelelően tesztelve. Nyilvánvaló tehát, hogy a tesztelés valóban kulcsfontosságú. Ám a tesztelést sem végezhetjük el csak úgy vaktában. Szükség van valamilyen rendszerezésre, hogy tesztelés során keletkezett információk értelmet nyerjenek.

Komolyabb programok írásakor többször szembesülhettünk azzal a problémával, hogy amint tesztelésre került a sor, problémák léptek fel. Ennek oka nem feltétlen a tudás hiánya, hanem inkább egy olyan rendszer hiánya, amelyben értelmet nyernek a tesztadatok. Ezért felmerült bennünk egy olyan tesztelési információkat nyilvántartó alkalmazás ötlete, amely pontosan ezt a problémát küszöbölik ki. A cikkben egy ilyen alkalmazás megtervezését, a fejlesztéséhez szükséges technológiák ismertetését, a fejlesztés kivitelezését, és a tapasztalatainkat írjuk le.

Az elkészülendő programnak képesnek kell lennie teszteseteket létrehoznia, szükség esetén ezeken módosításokat végeznie, törölnie. A program a véglegesnek tekinthető teszteseteket képes legyen futtatni, mely gyakorlatilag a tényleges tesztelést jelenti, és melynek eredménye a tesztelés során gyűjtött tapasztalatok. Ezeket az eredményeket tárolni kell, és igény esetén megjelenítenie. Ezen alapfunkciókon kívül szeretnénk, ha az eredményeket statisztikai diagramokkal szemléltné, a hibaüzeneteket naplózná, valamint képes lenne a már korábban meglévő, Excel dokumentumban található teszteseteket is beolvasni. Szeretnénk, ha az eredményekből hasonló dokumentumokat tudna generálni.

Rendkívül fontos elvárásunk, hogy a fent említett funkciókat úgy tudja a program megvalósítani, hogy az alkalmazás használata továbbra is érthető legyen mind egy fejlesztő, mind egy tesztelő számára. A programom elsődlegesen tesztelők számára készül, az ő munkájukat könnyíti meg. Egy olyan környezetet szeretnék megteremteni, amelyben egységesen történik a tesztelés, ezáltal hatékonyabbá válik a munkavégzés. Ez az alkalmazás reményeink szerint csökkentheti a fejlesztők és a tesztelők közötti távolságot. Sok esetben azért lassul le a fejlesztés menete, mert a fejlesztők és a tesztelők is másképp gondolkodnak, másképp képzelik el egy feladat megvalósítását. Az előbb említett egységes környezet ezt a problémát küszöbölné ki, ezzel rövidtávon a tesztelést, hosszútávón az egész fejlesztést gyorsítva. Ennek érdekében az alkalmazás bizonyos funkcióinak használatát a fejlesztőknek is ajánljuk.

2. Irodalmi háttér

A modern világban egyre több információt halmazunk fel. Ebből az információhalmazból időnként szükségünk van kisebb-nagyobb mennyiségű adatra ellenőrzéshez, módosításhoz, vagy újabb információk megszerzéséhez. Egyértelmű tehát, hogy az adatokat olyan módon kell tárolnunk, hogy szükség esetén gyorsan visszakereshessük őket. A mai nyilvántartó rendszerek megítélése az olyan fogalmaktól függ, mint a gyorsaság, a rendelkezésre állás, a robusztusság, a hibamentesség és nem utolsó sorban, az átláthatóság, egyszerűség.

Egy olyan alkalmazás, amelynek megfelelő használata nem jelent komolyabb problémát a felhasználónak, már fél sikert jelent. Legyen szó akár egy nyilvántartó rendszerről, vagy akár egy internetes ételrendelő alkalmazásról, az egyik legfontosabb cél, hogy a felhasználó könnyen tudja használni azt. Ennek hiányában a hatékonyság igencsak csökkenhet. Nem meglepő tehát, hogy az elkészülendő alkalmazás egyik alapelvárása az egyszerűségre való törekvés. Egy nagyon hasonló feladatokat megvalósító program a TestLink [1], amely egy nyílt-forráskódú tesztmenedzselő alkalmazás. Képes a tesztesetek, tesztlépések létrehozására, a tesztelés levezénylésére a megadatott követelmények alapján. Lehetőségünk van többszintű felhasználói jogosultságokat megadni. A teszt lefutása után részletes beszámoló és statisztika készül az eredményekről.

3. Az alkalmazás funkciói

A program általános feladata a tesztadatok kezelése, nyilvántartása. Ehhez szükség van egy olyan funkcióra, amellyel ilyen adatokat létre tudunk hozni. Ezt megtehetjük majd az alkalmazáson belül is, valamint importálhatunk megfelelő formátumú tesztdokumentumokat is az adatbázisba. Ha már vannak tesztadataink (struktúrák, komponensek, tesztesetek), akkor ezeket szükség esetén módosíthatjuk. Meg tudjuk változtatni egy egység nevét, leírását, a lépések sorrendjét és az elvárt működést is. Minden egységre igaz, hogy csak azelőtt van lehetőség módosítani, mielőtt lefuttatjuk a tesztet. A teszt lefutása után az eredményekbe természetesen már nem tudunk belenyúlni, csak a tesztesetek módosítása után egy újrafuttatással történhet ez meg. A tesztadatainkat törölhetjük is, ám ez a művelet komolyabb megfontolást igényel, mivel törlés esetén minden információt véglegesen elvesztünk, nem lesz lehetőség visszaállítani a későbbiekben. Ha törlünk egy tesztegységet, akkor a hozzátartozó teszteredmények is törlésre kerülnek.

Az egyik legfontosabb funkció a tesztek végrehajtása. Ennek következményeként jönnek létre a tesztadatok. Egy ilyen eredmény egység (instance – példány) tartalmazni fogja, hogy melyik tesztesetet futtattuk, valamint az általa tartalmazott tesztlépésekre vonatkozóan is megtudhatjuk, hogy teljesült-e az elvárt eredmény, volt-e valamilyen megjegyzés. Az alapfunkciók mellett megtalálható még a már korábban megemlített (Excel) dokumentum importálás, valamint ennek fordítottja is, azaz az alkalmazásban létrehozott tesztadatokat ki tudjuk exportálni egy Excel dokumentumba. Emellett megemlítendő még egy a tesztelési eredmények megkönnyítésére szolgáló statisztikai diagram generáló mechanizmus is.

4. Megoldási algoritmusok és módszerek

Első lépésként az adatbázis megtervezésre került sor. Fontos a táblák közötti kapcsolatok, valamint a tartalmazott rekordok pontos megtervezése, mivel ezek javítása a későbbiekben jelentős idő-, és erőforrásigénnyel járna. A kész adatbázisra épül az alkalmazás backend része, amely kapcsolatot teremt a program felhasználói felülete és az adatbázis között. A backend jól elkülöníthető rétegekből épül fel. Ezeket a rétegeket célszerű külön csomagokban elhelyezni. A legalsó réteg a repository réteg, amelyben olyan metódusok találhatóak, amelyek műveleteket végeznek el az adatbázisban. Efelett található az üzleti logika. A beérkező adatokon itt történik az utolsó módosítás, mielőtt tárolásra kerülnek az adatbázisban, illetve itt történnek az első változtatások az adatbázisból történő lekérés után. A backend legfelső rétege a controller réteg, melyben az ún. endpointok találhatóak, melyekre a frontend ráhív. Ez a réteg szolgál a backend és a frontend közti kapcsolat megvalósítására. A rétegek implementálása során óriási segítséget nyújtanak a Spring Boot [2], és a Hibernate [3] keretrendszer annotációi, melyekkel jelentősen csökkenthető a kód mennyisége, ezáltal gyorsítható a fejlesztés menete.

A program frontend része valósítja meg a felhasználói felületet, valamint kezeli a ki-, és bemenő adatokat is. Az Angular [4] programnyelv moduláris felépítést biztosít eme alkalmazásrésznek. Használatával elkülöníthető a JavaScript alapú TypeScript kód, az oldalak vázát adó HTML kód, valamint a stílust kezelő CSS/SASS kód.

A fejlesztéssel párhuzamosan zajlott egyfajta tesztelés, amely minden elkészült funkciót alaposan ellenőrzött. Amennyiben bármilyen hiba lépett fel, az azonnal helyben javításra került. Ellenkező esetben a fejlesztés folytatódhatott a következő funkció megvalósításával.

Természetesen prioritást élveznek az elsődleges funkciók, így ezek implementálása történt meg első körben. Az alkalmazás felhasználói felülete is úgy formálódott, hogy ezen funkciók tökéletesen

világosak, és könnyen elérhetőek legyenek. A kiegészítő funkciók esetében már nem előnyös olyan megoldások használata, amelyek nagyobb mértékű átalakítást igényelnek akár a felületen, akár a modulok logikájában.

5. A megoldás szoftverkörnyezete

A szoftver adatbázisának kezelésére az SQL adatbázis-kezelő nyelvet, míg az adatbázis-szerver létrehozásához és használatához a XAMPP szoftvercsomagot alkalmaztuk. Utóbbi rendelkezik beépített MySQL/MariaDB [5] adatbázis-kezelővel, valamint a phpMyAdmin nevű grafikus felülettel, amely nagyban segítette a munkánkat. A szoftver kódja jól elkülöníthető ún. backend és frontend részekre. Előbbi fejlesztése Java objektumorientált-programozási nyelven zajlott az Eclipse fejlesztőkörnyezetben.

Annak érdekében, hogy a backend írását kényelmesebbé és gyorsabbá tegyük, különféle keretrendszerek és technológiák segítségét vettük igénybe. A backend projekthez Spring Boot keretrendszert használtunk, míg az adatbázissal való kommunikációhoz a Hibernate objektum-reláció leképző rendszerét. A frontend megírásához a JavaScript-alapú Angular nyelvet alkalmaztuk. A fejlesztés a Microsoft által fejlesztett Visual Studio Code-ban történt. A fejlesztés során szükség volt még a Postman [6] nevű alkalmazásra, amellyel hatékonyan lehet HTTP kéréseket tesztelni, még a frontend elkészülte előtt.

6. Összefoglalás

A program használatát elsősorban olyan kisebb vállalatok számára ajánlanánk, amelyek nem rendelkeznek még valamilyen egységes tesztelési környezettel. Egy ilyen alkalmazás nagy segítség lenne számukra, mert gyorsíthatnák a tesztelési folyamatokat.

Szintén említettük már, hogy mennyire fontos szempont az egyszerűség és az érthetőség. A rendszer használatához nincs szükség semmilyen magas szintű szakmabeli tudásra. Természetesen előnyt jelent a hasonló programokkal való korábbi ismertség, azonban a használat nem okozhat gondot kezdő, illetve kevés tapasztalattal rendelkező tesztelők, fejlesztők számára. Terveink szerint az alkalmazásban rövidesen lesz egy bármikor elérhető használati útmutató, amely a működés mellett az adatok felépítését, és kapcsolatát is világosan szemlélteti. Noha az alkalmazás elsődleges verziója elkészült, ez nem jelenti azt, hogy még ne lenne vele bármilyen munka a későbbiekben. Igazából a programban még rengeteg potenciál rejtőzik, és ennek kiaknázása csupán idő és erőforrás kérdése.

A fejlesztés ismertetése során többször is szembe kerültünk olyan problémákkal, amelyek miatt az alkalmazás nem teljesít minden olyan funkciót, amit a tervezés során kikötöttünk. Talán a legfontosabb ilyen az Excel dokumentumok importálásának mechanizmusa. Ez a későbbiekben még megoldandó feladat, valamint célszerű lenne a bővítés során az exportálás átalakítása, finomítása is. Rengeteg olyan megoldás is ide tartozik, amelyek az alkalmazás használatát tehetik könnyebbé a felhasználó számára, mint például a részletek oldalon dinamikusan működő „Vissza” gomb beépítése, amihez jelenleg még backend szintű változtatásokra is szükség van. A statisztikai kördiagram fejlesztése is esedékes lehet egy későbbi verzióban, mely sokkal interaktívabbá tehetné az alkalmazást, vagy akár más típusú diagramok integrálása is elképzelhető. Emellett olyan kiegészítő funkciók is szóba jöhetnek, mint egy egyedileg kiválasztható teljes alkalmazás dizájn (színek változtatása).

Egy sokkal nagyobb volumenű bővítési lehetőség egy bejelentkező felület kialakítása lehetne. Ebben az esetben az egyes felhasználók különböző jogosultságokkal rendelkezhetnének, továbbá az ada-

tok birtoklását jelző mezők (author) automatikusan kitöltődnének a felhasználónév alapján. Természetesen egy ilyen mechanizmus megvalósítása hatalmas mértékű újratervezést és átszervezést jelent, ezért került csak megemlítésre.

Irodalom

[1] TestLink - User Manual, 2010. [Online]. https://wiki.openoffice.org/w/images/1/1b/Testlink_user_manual.pdf
[2] Spring Boot Overview, https://spring.io/projects/spring-boot
[3] Hibernate documentation, 2020. [Online]. https://hibernate.org/orm/documentation/5.4/.
[4] Angular documentation, 2020. [Online]. https://angular.io/docs.
[5] MongoDB vs MySQL 2., 2020. [Online]. https://www.mongodb.com/compare/mongodb-mysql.
[6] Postman Development Environment, https://www.postman.com/products/
[7] Fejti Martin: Testing Manager, Szakdolgozat, ME ÁIT, 2020