UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Budal

Avtomatsko performančno testiranje spletnih aplikacij

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: viš. pred. dr. Damjan Vavpotič

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00075/2011
Datum: 03.03.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MARKO BUDAL**

Naslov: **AVTOMATSKO PERFORMANČNO TESTIRAJE SPLETNIH APLIKACIJ**
AUTOMATED PERFORMANCE TESTING OF WEB APPLICATIONS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V okviru diplomskega dela najprej predstavite širše področje performančnega testiranja in njegovo umestitev med ostala testiranja. Predstavite pomen performančnega testiranja za preverjanje pravilnega delovanja informacijskih sistemov ter okoliščine v katerih je performančno testiranje še posebej pomembno. V nadaljevanju naloge prestavite načine in orodja za avtomatizacijo performančnega testiranja spletnih aplikacij ter prednosti in slabosti, ki jih takšna avtomatizacija prinaša. V okviru praktičnega dela naloge demonstrirajte avtomatizirano performančno testiranje na konkretnem primeru.

Mentor:

viš. pred. dr. Damjan Vavpotič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Marko Budal,

z vpisno številko 63070415,

sem avtor/-ica diplomskega dela z naslovom:

Avtomatsko performančno testiranje spletnih aplikacij.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek) viš. pred. dr. Damjan Vavpotič.
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Kopru, dne 10.6.2011 Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorju viš. pred. dr. Damjanu Vavpotiču za vodenje in strokovno pomoč pri izdelavi diplomske naloge.

Posebna zahvala gre moji družini, saj so me ves čas študija spodbujali in mi stali ob strani.

Zahvaljujem se Anji Rožac, Svetlani Šaljić in razvojni ekipi podjetja SETCCE, od katerih sem se v zadnjem letu veliko naučil in zaradi katerih je tudi nastala tema diplomske naloge.

Zahvaljujem se tudi Nini Ceglar in Barbari Škrbini za lektoriranje diplomske naloge.

Kazalo

POVZETEK **1**

ABSTRACT **2**

1. UVOD **3**

2. NAČIN TESTIRANJA SPLETNIH APLIKACIJ **5**

2.1 Splošno o spletnih aplikacijah **5**

2.2 Testiranje spletnih aplikacij **6**

2.3 Namen testiranja in življenjski model uspešnega testiranja spletne aplikacije **7**

2.3.1 Prepoznavanje okolja 8

2.3.2 Prepoznavanje zmogljivostnih meril 9

2.3.3 Načrtovanje zmogljivostnih testov 10

2.3.4 Priprava testnega okolja 12

2.3.5 Razvijanje oblike za zmogljivostne teste 12

2.3.6 Izvajanje zmogljivostnih testov 13

2.3.7 Analiza rezultatov, poročilo in ponovno testiranje 14

2.4 Zakaj izvajati zmogljivostne teste **16**

2.5 Zmogljivostno, bremensko in stresno testiranje **17**

2.5.1 Test zmogljivosti 17

2.5.2 Bremenski testi 17

2.5.3 Stresni testi 17

2.5.4 Pridobitve pri posameznih testnih kategorijah 18

2.6 Prednosti, slabosti in smernice pri izvajanju avtomatskih testov **19**

3. PREGLED ORODIJ ZA IZVAJANJE ZMOGLIVOSTNIH TESTOV **22**

3.1 Predstavitev orodij za avtomatsko testiranje **23**

3.1.1 Forecast 23

3.1.2 LoadTracer 23

3.1.3 OpenLoad 24

3.1.4 JMeter 24

3.1.5 vTest 25

3.1.6 QEngine 25

3.1.7 WAPT 25

3.1.8 Scapa Test and Performance Platform 25

3.1.9 OpenSTA 26

3.1.10 Siege 26

3.1.11 Pylot 26

4. PREDSTAVITEV PROBLEMA **27**

4.1 SETCCE proXSign **27**

4.2 SETCCE ePero **27**

4.3 Predstavitev spletne aplikacije ePero **29**

4.3.1 Arhitektura spletne aplikacije ePero in uporaba spletnih storitev 30

4.3.1.1 ePero odjemalec 30
4.3.1.2 Podpisni strežnik 31
4.3.2 Predstavitev tokov dogodkov 32

5. IZBIRA ORODIJ ZA IZVAJANJE AVTOMATSKIH ZMOGLJIVOSTNIH TESTOV IN IZVAJANJE MERITEV 36
5.1 Priprava in izvajanje zmogljivostnih testov z orodjem JMeter 36
5.2 Priprava in izvajanje zmogljivostnih testov z orodjem WAPT Pro 40

6. ANALIZA IZVAJANIH TESTOV 45
6.1 Rezultati testov izvajanja z orodjem WAMP 45
6.2 Rezultati testiranja z orodjem JMeter 46

7. ZAKLJUČEK 50

8. KAZALO SLIK 51

9. KAZALO TABEL 52

10. VIRI 53

Slovarček:

Apache Tomcat Strežnik – Apache Tomcat je odprtokodna programska oprema, ki se pogosto uporablja za gostovanje spletnih aplikacij.

CSS (ang. *Cascading style sheets*) – slogovna predloga, ki lahko določa zgled spletnih strani

DNS (ang. *Domain name system*) – Kretica predstavlja tehnologijo za upravljanje y imeni spletnih strani ali drugih področji interneta.

FTP (ang. *File transfer protocol*) – Omrežni protokol za prenos datotek med dvema računalnikoma.

GUI (angl. *Grafical user interface*) – grafični uporabniški vmesnik

HTML zahteva – (ang. *Hypertext transfer protocol request*) računalnik pošlje zahtevo za datoteke na spletni strežnik

HTTP (ang. *Hyper text transfer protocol*) – protokol za izmenjavo nadbesedil ter grafičnih, zvočnih in drugih večpredstavnostnih vsebin na spletu

HTTPS (ang. *Hyper text transfer protocol Secure*) – protokol je podoben HTTP in opravlja šifrirano komunikacijo in varno identifikacijo omrežnega strežnika

IMAP (ang. Internet message access protokol) – eden od pogostejših internetnih protokolnih standardov za iskanje e-mailov

IP (ang. *Internet protocol*) – glavni komunikacijski protokol

MTBF (ang. *Mean time between failure*) – povprečni čas med dvema odpovedma

MTTF (ang. *Mean time to failure*) – povprečen čas do odpovedi

PDF (ang. *Portable document format*) – standard ya izmenjavo dokumentov

PDF-A (ang. *Portable document format*) – standard za dolgoročno arhiviranje elektronskih dokumentov

URL (ang. *Uniform resource locator*) – je oblikovano besedilo, ki prepozna omrežne vire na internetu. Omrežni viri so datoteke, ki vsebujejo vse podatke o spletni strani, aplikaciji itd.

XML (ang. *Extensible markup language*) – niz pravil za kodiranje dokumentov v strojno berljivi obliki

POVZETEK

Proces testiranja spletnih aplikacij je eden od ključnih procesov pri razvoju kakovostne spletne programske opreme. Pogoj za kakovostno programsko opremo je odzivnost, prepustnost, zanesljivost, varnost in razširljivost aplikacije, četudi je izpostavljena visokim obremenitvam. V diplomski nalogi sem bil osredotočen na izvajanje avtomatskih zmogljivostnih testov. Ti so namenjeni prepoznavanju ozkih grl v sistemu, zagotavljanju izhodišča za prihodnja testiranja ter ugotavljanju skladnosti z zmogljivostnimi cilji in zahtevami.

Namen diplomske naloge je predstaviti pristop k testiranju zmogljivosti spletnih aplikacij in izdelati pregled orodij, ki jih imamo na voljo pri testiranju. Tako sem naprej predstavil življenjski model uspešnega testiranja spletne aplikacije. Ob modelu sem podrobneje opisal, kaj nam prinaša zmogljivostno testiranje, nato pa sem predstavil postopke, orodja in pripravo za izvajanje avtomatskih zmogljivostnih testov. V nadaljevanju sem primerjal med seboj pridobljene rezultate posameznih testih scenarijev in se odločil, katero orodje je po mojem mnenju najprimernejše za uporabo pri izvajanju avtomatskih zmogljivostnih testov.

Kljub temu, da se v diplomskem delu osredotočam na izvajanje avtomatskih zmogljivostnih testiranj spletnih aplikacij, bi se podoben pristop lahko uporabilo pri drugih vrstah testiranj spletnih aplikacij oz. druge programske opreme.

Ključne besede:

avtomatsko testiranje, zmogljivostno testiranje, spletne aplikacije, življenjski model testiranja, testni scenarij.

ABSTRACT

The process of testing web applications is one of the key processes when it comes to development of high quality web-based software. High quality web-based software is adorned with quick responsiveness, bandwith, reliability, safety and a possibility to widen the application under a certain workload. In my diploma I focused on the implementation of automatic performance tests. Performance tests are designed for recognizing bottlenecks in the system, for assuring a starting-point for future testing, and for determining accordance with performance goals and demands.

This diploma tries to present a life model of successful web application testing and the implementation of performance tests on the web application ePero with different tools for automatic performance testing. It presents procedures, tools, and preparations for the implementation of automatic performance tests. Furthermore, it compares the results given by particular test scenarios. Based on those comparisons, I decided which tool is in my opinion the most appropriate for the implementation of automatic performance tests.

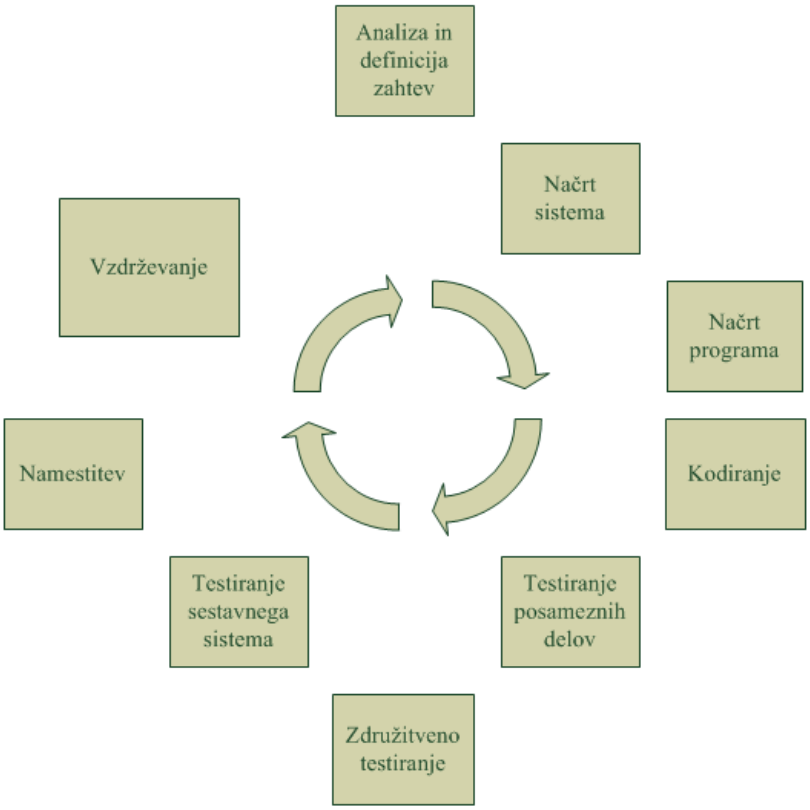
Despite the fact that, in my diploma, I focus on the implementation of automatic performance testing of web applications, a similar approach could also be used for other types of web applications testing or for the testing of other types of software.

Key words: automatic testing, performance testing, web applications, life model of testing, test scenario

1. UVOD

Tako od samega začetka razvoja programske opreme kot pri kasnejšem razvoju internetnih storitev je bilo treba za kakovostne izdelke vpeljati pristope, ki zagotavljajo visoko stopnjo kakovosti razvoja programske opreme in spletne programske opreme. Danes se uveljavljajo spletne tehnologije kot ene ključnih tehnologij za elektronsko poslovanje, saj omogočajo povezljivost med informacijskimi sistemi ne glede na platformo, programski jezik ali operacijski sistem. Za opis pristopa k razvoju spletene programske opreme uporabljamo devet faz (analiza in definicija zahtev, načrt sistema, načrt programa, kodiranje, testiranje posameznih delov, združitveno testiranje, testiranje sestavnega sistema, namestitev, vzdrževanje) [Slika 1], med katerimi se bom v diplomski nalogi osredotočil predvsem na testiranje, natančneje avtomatsko testiranje spletnih aplikacij.

Testiranje spletnih aplikacij postaja iz dneva v dan težje, saj spletne aplikacije postajajo vedno bolj kompleksne in posledično zahtevnejše za testiranje.



Slika 1: Življenjski model razvoja spletne/programske opreme [6].

Avtomatski testi so se pojavili zaradi svoje kompleksnosti in obsežnosti. Ti lahko ob spremembah na spletni aplikaciji hitro preverijo pravilnost delovanja. Po navadi se avtomatski

testi izvajajo zato, da preverijo funkcionalnost, združljivost, zmogljivost[1] in varnost. Testiranje grafičnega vmesnika, uporabnosti in tudi varnosti pa po navadi ne opravljamo z avtomatskimi testi.

Alberto Savoia[2] je mnenja, da je testiranje delovanja spletnih aplikacij »preprost proces«, to pa zato, ker ljudje, ki se ukvarjajo s testiranjem spletnih aplikacij, oblikujejo nerealne scenarije. Z napačnimi scenariji pridobijo ogromno neuporabnih podatkov, ki jih z napačnimi metodami in analizami obdelajo ter napačno predstavijo dobljene rezultate. Po mnenju Savoie je treba oblikovati testne scenarije načrtno in razumno. Ne bodo nam dali želenih rezultatov, temveč nam bodo dobljeni rezultati prikazali realno podobo zmogljivosti testirane spletne aplikacije. Pri tem pa se je vseeno treba držati reka Alberta Einsteina: »Make everything as simple as possible, but not simpler.«

V diplomski nalogi sem podrobno raziskoval področje avtomatskega zmogljivostnega testiranja spletnih aplikacij. Za to področje sem se odločil, ker sem v podjetju, za katerega delam, veliko časa porabil za iskanje ustreznega orodja za izvajanje zmogljivostnih testov in za testiranje novega produkta ePero. Z izvajanjem zmogljivostnih testov na spletni aplikaciji ePero sem odkril napake in aplikacijo tako optimiziral, da je delovala gladko in neprekinjeno, brez kakršnih koli zakasnitev. Z zmogljivostnimi testi sem ugotovil, kako se bosta aplikacija in strežnik obnašala, ko se bo nanj povezalo določeno število uporabnikov. Diplomsko nalogo sem v grobem razdelil na dva dela. V prvem delu je zajeto teoretično znanje s področja avtomatskih testov, ki sem ga pridobil na predavanjih, ob branju gradiv na internetu in na praksi. Prav tako sem se dotaknil področja testnih orodij. Predstavil sem poznana orodja in razlike, ki sem jih opazil pri uporabi. V drugem delu sem se lotil testiranja spletne aplikacije ePero. Ob začetku drugega dela sem opisal spletno aplikacijo, t. i. »SETCCE ePero« in vtičnik »proXSign«. Pod opis spletne aplikacije sem dodal tri scenarije, ki sem jih testiral z različnimi orodji za avtomatsko zmogljivostno testiranje spletnih aplikacij. Uporabljena orodja za testiranje spletnih aplikacij sem opisal v petem poglavju. Poleg opisa orodij za izvajanje avtomatskih zmogljivostnih testov sem v tem poglavju predstavil dobljene rezultate.

---

[1] Performanse (ang.).

[2] Alberto Savoia je med leti 1985-1998 delal na »Sun microsystems«. Začel je kot član testne ekipe in napredoval do glavnega odgovornega za razvoj. Med leti 2001-2002 je delal na »Googlu« kjer je bil višji direktor inženirjev. Leta 2002 je postal soustanovitelj podjetja »Agitar«.

2. NAČIN TESTIRANJA SPLETNIH APLIKACIJ

2.1 Splošno o spletnih aplikacijah

Največji in najpopularnejši medij v današnjem času je Internet. Uporablja ga vedno več ljudi. Namen uporabe je različen. Nekateri ga uporabljajo za zabavo, drugi prenašajo različne multimedijske datoteke, tretji le preverjajo informacije, a na poslovnem področju je Internet že dolgo nepogrešljiv način povezovanja in komunikacije.

Dnevno nastajajo podjetja, ki razvijajo spletne aplikacije. Spletna aplikacija (ang. web application) je aplikacija, ki jo uporabnik uporablja s pomočjo spletnega brskalnika. Preko internetnega ali lokalnega omrežja se s pomočjo spletnega strežnika (več o tem v poglavju 3.1.1) aplikacija prenese in jo prikaže na računalniškem zaslonu.

Prednosti spletnih aplikacij [4]:

- *ni jih treba nameščati ali posodabljati*

  S pomočjo brskalnikov zaganjamo in upravljamo spletno aplikacijo. Ta nikoli ne zahteva namestitvenih postopkov. Aplikacija »živi« na strežniku, kar nam omogoča, da ob vsakem zagonu spletne aplikacije uporabljamo najnovejšo različico z zadnjimi popravki. Možnost opravljanja posodobljene aplikacije imajo vsi uporabniki, saj se popravki ne nalagajo pri posameznem uporabniku, temveč se novi popravki namestijo na strežnik. Pri poganjanju spletne aplikacije nam Ajax pomaga narediti spletno aplikacijo odzivnejšo.

- *ne potrebuje administratorskih pravic za nameščanje*

  Spletnih aplikacij ni treba nameščati na delovno postajo, saj se te izvajajo preko brskalnikov, zato ne potrebujemo administratorja, ki nam omogoči delovanje aplikacije. Pri tem velja omeniti, da določene spletne aplikacije potrebujejo za svoje delovanje vtičnike (npr. SETCCE proXSign, več o tem v poglavju 3.1).

- *Performančna neodvisnost*

  Platformna neodvisnost je ena od velikih prednosti spletnih aplikacij. Proizvajalcem programske opreme so pred leti spletne aplikacije odprle nove možnosti na tržišču. Brskalnik postane platforma, ta pa omogoči uporabo istih spletnih aplikacij na različnih sistemih (Mac, Windows, Linux itd.).

- *manj okoljskih sporov*

  Verjetnost napak, ki so posledica nastavitev strojne opreme, se izniči.

- *omogoča socialne stike*

  Ogromno spletnih aplikaciji omogoča klepet[3] in deljenje dela z ostalimi sodelavci v realnem času. Ta funkcionalnost odpravlja predhodno obdobje namiznih aplikacij, kjer je velik del programskega razvoja opravljala le ena oseba. Socialno omrežje je danes že močno razvito. Ljudje se želijo povezovati z drugimi in opravljati različne aktivnosti. Spletne aplikacije nam to omogočajo brez večjih težav.

- *nižji stroški prodaje*

  Nižji stroški prodaje se pokažejo, ker spletne aplikacije »živijo« na spletnem strežniku. Aplikacij ni treba shranjevati na zgoščenke, ne potrebujemo embalaže za pakiranje produkta, ni treba natisniti priročnika, ne ustvarimo stroškov prevoza, ne potrebujemo distribucijskih posrednikov itd. Ostajajo pa še vedno stroški reklamiranja.

- *nizki stroški podpore in vzdrževanja*

  Glede na to, da je brskalnik platforma, so potrebe po dragi strojni opremi odveč. Uporabniku ni treba skrbeti za podporo, saj se vse to dogaja na strežniku.

- *mobilnost je tukaj*

  Namizne aplikacije bo treba prilagoditi za uporabo na sistemih mobilnih telefonov. Spletne aplikacije se preko mobilnih brskalnikov brez težav poganja na mobilnih telefonih, večina pa jih ima prilagoditev, da na mobilnih napravah delujejo drugače kot na računalniku, tako da so na mobilnikih hitrejše.

Večinoma sem opisoval le prednosti spletnih aplikacij. Kot vsaka stvar imajo tudi spletne aplikacije svoje slabosti. Zaradi svoje razširjenosti in delovanja preko spleta so občutljivejše na napade in zlorabe. Če želimo zagotoviti, da bodo novonastale spletne aplikacije čim bolj varne, moramo poskrbeti za njihovo brezhibno delovanje. To lahko zagotovimo z orodji za razvoj spletnih aplikacij, ki zmanjšujejo možnost za napake, ter s testiranjem delovanja spletnih aplikacij. Zaradi obsežnosti testov so prišli v uporabo avtomatski testi, o katerih bom več povedal v nadaljevanju.

2.2 Testiranje spletnih aplikacij

Spletno testiranje je testiranje programske opreme, ki se osredotoča na spletne aplikacije. Spletno testiranje je zbiranje informacij, ki so povezane s cilji, kot so: odkrivanje napak v vsebini spletnih aplikacij, delovanje funkcij, uporabnosti, učinkovitosti, zmogljivosti in varnosti. Za dosego visoke kakovosti spletne aplikacije je nujno, da je v proces spletnega razvoja vključena vsaj ena oseba z izkušnjami z vodenjem projektne ekipe.

Ljudje, ki vodijo projektno ekipo in testirajo spletne aplikacije, morajo odpraviti čim več napak, preden se spletno aplikacijo postavi v produkcijsko okolje. Testira se vsebino, vmesnike, navigacijo, komponente, konfiguracijo, delovanje in varnost. Pridobljene

[3] Ang. Chat

informacije in zbirko testnih primerov hranimo v arhivu za poznejšo uporabo. Testerji pri testiranju uporabijo različne tehnologije in orodja, kar pomeni, da imajo pri izvajanju testnih scenarijev veliko različnih možnosti.

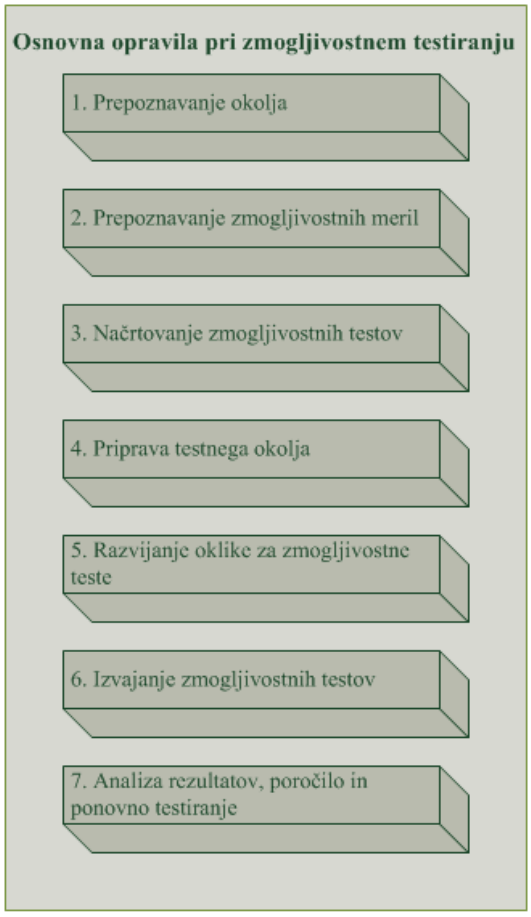
Pri razvoju dobrih spletnih aplikaciji je posredno vključena tudi njena kakovost. Kakovostna spletna aplikacija bo tista, ki bo stranko zadovoljila in ji dala občutek varnega poslovanja s spletno aplikacijo, zato je kakovost zelo pomembna. Pri testiranju spletne aplikacije testiramo njeno vsebino na sintaktični in semantični ravni. Pregledovanje sintakse pomeni, da preverimo, ali je besedilo pravopisno in slovnično pravilno. Na semantični ravni preverimo ustreznost vsebine besedila. Za ostale dele spletne aplikacije (funkcionalnost, združljivost, zmogljivost in varnost) pri testiranju uporabimo avtomatske teste. Z avtomatskimi testi preverimo navigacijo skozi spletno aplikacijo. Ob tem, ko testiramo navigacijo, se nehote testirajo tudi posamezne komponente, ki so vključene v spletni aplikaciji. Testiranje pravilnega delovanja aplikacije je poleg testiranja varnosti eno od ključnih testiranj. Scenarij testiranja varnosti in delovanja morata biti vnaprej dobro premišljena in implementirana. Dobro pripravljeni scenariji nam omogočajo hitro odkrivanje napak, preden se postavi spletno aplikacijo v produkcijo [10].

2.3 Namen testiranja in življenjski model uspešnega testiranja spletne aplikacije

Zakaj torej izvajati celoten postopek testiranja, če že vnajprej vemo, da ne bomo uspeli odkriti vseh napak? Osnovni namen testiranja je odkrivanje ozkih grl v sistemu, odkrivanje napak v programski kodi, preverjanje izvajanja spletne aplikacije, preverjanje shranjevanja podatkov, postavljanje izhodišč za nadaljnje testiranje in ugotavljanje, kakšna so odstopanja od zahtevanih zmogljivosti.

Podatki, pridobljeni pri testiranju, nam služijo za ugotavljanje kakovosti spletne aplikacije, hkrati pa nam služijo tudi za ugotavljanje ustreznosti strojne opreme, ki pomembno vpliva na delovanje spletne aplikacije.

V naslednjih točkah bom predstavil sedem aktivnosti [Slika 2], ki se uporabljajo v praksi, pri temeljitem testiranju spletnih aplikacij. Ključ za dobro testiranje ni le »sprehod« skozi te aktivnosti, ampak temeljito poznavanje aplikacije. Spletno aplikacijo je nujno treba razumeti, da se pravilno odločimo, na kakšen način se bo testiralo spletno aplikacijo.



Slika 2: Osnovna pravila pri zmogljivostnem testiranju [6].

2.3.1 Prepoznavanje okolja

Okolje, na katerem so bo izvajala naša spletna aplikacija , skupaj z orodji in s tem povezano strojno opremo, ter vsi pripomočki, potrebni za izvedbo preizkusov, predstavljajo testno okolje. Pri testnem okolju je treba stremeti k temu, da je okolje v čim večji meri podobno produkcijskemu okolju. To je v večini primerov neizvedljivo, saj so stroški za postavitev takega testnega okolja previsoki. Z nižjimi stroški pa še vedno lahko naredimo tako testno okolje, ki bo zelo podobno produkcijskemu okolju.

Stopnja podobnosti med strojno opremo, programsko opremo in omrežno infrastrukturo igra pomembno vlogo pri določanju, kakšni zmogljivostni testi se bodo izvajali. V tej fazi je pomembno upoštevati in vedeti, da na rezultate testov ne vpliva le strojna in programska oprema, ampak predvsem namembnost samega testa. Pogosto se pri ponavljajočem se testiranju aplikacijskih zmogljivosti uporabljajo nove infrastrukturne rešitve, da se lahko izvajajo isti zmogljivostni testi na različno zmogljivi strojni opremi. Tako testiranje nam zagotavlja verodostojne podatke.

Ključni dejavnik pri določanju testnega okolja je popolno razumevanje podrobnosti spletne aplikacije, razumevanje testnega in produkcijskega okolja.

Kritični dejavniki, na katere je treba biti pozoren:

- Strojna oprema
  - Konfiguracije
  - Posamezne komponente strojne opreme (procesor, RAM, disk itd.)
- Omrežje
  - Arhitektura omrežja
  - Konfiguracija DNS
- Orodja za testiranje
  - Orodja za nalaganje aplikacije
  - Orodja za nadzor sistema
- Programska oprema
  - Ostali programi za izmenjavo podatkov ali virtualni strežniki
  - Licence za programsko opremo
  - Shranjevanje pridobljenih podatkov
- Zunanji dejavniki
  - Obseg in vrsta dodatnega prometa na omrežju
  - Posodobitve, varnostne kopije
  - Medsebojno delovanje z drugimi sistemi

Vhodno-izhodni podatki faze »prepoznavanje testnega okolja«

Faza	**Vhod**	**Izhod**
Prepoznavanje testnega okolja	• Logična in fizična produkcijska arhitektura • Logična in fizična testna arhitektura • Orodja, ki jih imamo na voljo	• Primerjava med tesnim in produkcijskim okoljem • Ugotavljanje, ali potrebujemo dodatna orodja

Tabela 1: Prepoznavanje testnega okolja

Pod drobnogledom:

- Prepoznati je treba kritičnost sistemskih komponent. Preveriti je treba, ali obstajajo kakršne koli integracijske točke, ki so izven nadzora testiranja.
- Dobro je poznati IT-osebje, saj pri testiranju spletnih aplikacij potrebuješ orodja za spremljanje celotnega omrežnega prometa, pa tudi zato, da lahko konfiguriraš različna spletna orodja.
- Preveriti je treba namestitev DNS, saj nam lahko to povzroči zakasnitve pri povezovanju z baznimi podatki.
- Preveriti je treba požarni zidu, da nas ne bi omejeval pri testiranju spletne aplikacije.

2.3.2 Prepoznavanje zmogljivostnih meril

Na splošno je smiselno, da postavimo oceno naše spletne aplikacije že v razvojnem ciklu aplikacijskega življenja. To nam omogoči, da se osredotočimo na zmogljivosti aplikacije.

Oceno uporabimo pri predstavitvi modela spletne aplikacije, kjer stranki, s katero sklepamo nov posel, nazorno predstavimo naš pogled na aplikacijo, in ji na podlagi ocene lažje svetujemo. Razredi z našimi ocenami, ki jih predstavljamo stranki, so:

- odzivni čas
  - Primer: Določen dokument z elektronskim podpisom se mora prikazati v času treh sekund.
- zmogljivost
  - Primer: Sistem mora podpirati vnos več dokumentov hkrati v dokumentni sistem.
- poraba sistema
  - Primer: Za nalaganje dokumentov ne smemo porabiti več kot 75 odstotkov procesorja.
- Drugi pomembni viri, ki jih je treba upoštevati pri določanju ciljev, so pomnilnik, vhodno-izhodne enote ter omrežje.

Vhodno-izhodni podatki faze »prepoznavanja zmogljivostnih meril«

Faza	**Vhod**	**Izhod**
Prepoznavanje zmogljivostnih meril	• Pričakovanja stranke • Ublažitev tveganj • Pogodbene obveznosti	• Merila za uspeh zmogljivostnih testiranj • Zmogljivostni cilji in zahteve • Ključna področja raziskave • Ključni kazalniki uspešnosti • Ključni kazalniki poslovanja

Tabela 2: Prepoznavanje zmogljivostnih meril

Pod drobnogledom:

Pri določanju meril uspešnosti je treba upoštevati:

- uporabnikova pričakovanja,
- pogodbene obveznosti,
- industrijske standarde,
- celoten obseg predvidenih obremenitev,
- testne scenarije,
- ključne kazalnike uspešnosti,
- optimizacijo ciljev,
- varnostne faktorje (prostor za rast, razširjenost),
- seznam osebja, proračun, sredstva, znanje.

2.3.3 Načrtovanje zmogljivostnih testov

Načrtovanje in oblikovanje testov zmogljivosti vključuje določanje temeljih scenarijev uporabe, določanje ustrezne variabilnosti med uporabniki, ugotavljanje in generiranje

poskusnih scenarijev. Te predpostavke bodo ključni temelj pri izvajanju, nadzoru in ugotovitvah, ki jih bomo kasneje pridobili med samim testiranjem.

Pri projektiranju in načrtovanju testnih scenarijev moramo vedeti, da bodo pridobljeni rezultati ključni pri določanju dodatnih funkcionalnosti oz. oceni kakovosti izdelka, zato morajo biti testni scenariji ustvarjeni s tem namenom, da bodo simulirali realno situacijo.

Pogosto se zgodi, da je predhodne scenarije težko napisati, preden se oseba, ki se ukvarja z izvajanjem testov, sooči s samo spletno aplikacijo. Zato je dobro upoštevati osnovna pravila pri pisanju scenarijev.

- Upoštevati pogodbeno določene scenarije
- Opis scenarija za splošno rabo
- Uporaba zmogljivostnih zahtevnega scenarija

Po testiranju vnaprej pripravljenih scenarijev lahko vidimo, kakšni so pridobljeni rezultati testiranj v primerjavi s predhodno oceno aplikacije. To nam pomaga pri ugotavljanju kritičnih točk v aplikaciji, in nam posreduje podatke o tem, na katerih mestih bi lahko izboljšali aplikacijo. Po prvih testih lahko dodamo nove scenarije, če sumimo, da so ti potrebni.

Vhodno-izhodni podatki faze »načrtovanje zmogljivostnih testov«

Faza	**Vhodni**	**Izhodni**
Načrtovanje zmogljivostnih testov	• Dostopne aplikacije • Uporaba predhodno načrtovanih scenarijev • »Unit testi[4]« • Zmogljivostna merila sprejemljivosti	• Konceptualna strategija • Orodja in viri, ki so na voljo • Model za simulacijo testa • Testni podatki, potrebni za testiranje • Pripravljeni testi za testiranje

Tabela 3: Načrtovanje zmogljivostnih testov

Pod drobnogledom:

- Realistični testni modeli so občutljivi na uporabo ljudi, omrežne aktivnosti in interakcije z drugimi aplikacijami.
- Realistični modeli naj temeljijo na situacijah, ki se dogajajo v realnem svetu, ne pa na idealistični predpostavki.
- Četudi se za pripravo realne simulacije porabi več časa in sredstev, je ta strošek ničen v primerjavi z ugotovitvami kakovosti spletne aplikacije.
- Uporaba podatkov iz slabo pripravljenih scenarijev lahko vodi do slabih odločitev, zadnje pa k slabi kakovosti končnega produkta.

[4] Testiranje enot: testira se le del programske kode (enota).

2.3.4 Priprava testnega okolja

V natančno pripravo testnega okolja sodijo različna orodja in sredstva za izvajanje testov. Programi za testiranje in spremljanje, obnašanje spletnih aplikacij med izvedbo testiranja niso preprosta orodja, zato jih morajo ljudje, ki se ukvarjajo z izvajanjem testiranj, poznati. Pri pripravi testnega okolja lahko nastanejo tudi druge težave, ki niso povezane s programi za testiranje spletne aplikacije. Te težave so lahko z mrežo, požarnim zidom, prostorom na disku itd. Pomembno je, da se morebitnim težavam izognemo še preden se začne testirati novo aplikacijo.

Slabe nastavitve okolja lahko pripeljejo do tega, da se pri večkratnem izvajanju istih testnih scenarijev pojavijo anomalije pri zbiranju rezultatov testov. Zato se je treba pred testiranjem prepričati, ali je vsa strojna oprema ustrezno nameščena. Treba je preveriti tudi programsko opremo, ali je ustrezna in posodobljena za izvajanje simulacijskih testov.

Vhodno-izhodni podatki faze »priprave testnega okolja«

Faza	Vhodni	Izhodni
Priprava testnega okolja	• Konceptualna strategija • Testni programi • Načrt izvajanja testov	• Ustrezno nameščeni programi za testiranje spletne aplikacije • Pripravljeno okolje za testiranje spletne aplikacije

Tabela 4: Priprava testnega okolja

Pod drobnogledom:

- Pomembno je, da se na tem delu ugotovi, koliko podatkov lahko vstavimo, preden obremenitev doseže zgornjo mejo.
- Dobro je preveriti, ali so vse sistemske ure sinhronizirane, da ne bi prišlo do napačnega izvajanja zaradi različnih časovnih obdobij.
- Preveriti je treba točnost obremenitvenega testa, in sicer z uporabo obremenitvenih orodij. Pomembno je, da ta orodja znajo generirati različne IP- naslove, in tako simulirati realno situacijo obremenitve sistema.
- Nadzorovati je treba porabo virov (CPU, RAM, omrežja, diska itd.) na strežniku, da bi ugotovili, koliko aplikacija obremenjuje sistem.

2.3.5 Razvijanje oblike za zmogljivostne teste

Podrobnosti o obliki zmogljivostnih testov vsebuje vsako testno orodje posebej, kar pomeni, da nam boljša orodja omogočajo kreiranje boljših zmogljivostnih testov. S tem pridemo do več uporabnih podatkov. Ne glede na to, kakšen program za testiranje uporabljamo, je treba ustvariti skript, ki opiše potek že prej opisanega scenarija. Več scenarijev nam v končni fazi predstavlja celoten model testiranja.

Razvoj orodij za testiranje spletnih aplikacij se razvija zelo počasi in ne uspe slediti novim tehnološkim odkritjem. Programerji, ki se ukvarjajo z orodji za testiranje, tako finančno kot tehnološko težko sledijo hitremu razvoju. Hitro se lahko zgodi, da postane največji izziv pri testiranju nove funkcionalnosti pridobivanje resničnih podatkov o zmogljivosti spletne aplikacije. Problem nastane, ker test ne poda nobenih legitimnih podatkov o simuliranju izmišljenih uporabnikov in pravih uporabnikov. V takem primeru je treba pridobivanju verodostojnih podatkov nameniti več truda, časa in kapitala.

Vhodno-izhodni podatki faze »razvijanja oblike za zmogljivostne teste«

Faza	**Vhodni**	**Izhodni**
Razvijanje oblike za zmogljivostne teste	• Konceptualna strategija • Okolje • Orodja • Izoblikovani scenariji	• Potrjeni, izvršljivi testi • Potrjena orodja za testiranje • Zbrani podatki za testiranje

Tabela 5: Razvijanje oblike za zmogljivostne teste

Pod drobnogledom:

- Poskrbeti je treba, da so testni podatki pravilno uporabljeni. Med testne podatke sodijo podatki, zbrani v zbirkah, besedilne datoteke, spremenljivke v pomnilniku ali preglednice. Pri uporabi podatkovne baze je treba biti previden predvsem zato, da nam pri testiranju testni podatki ne uničijo ostalih pomembnih podatkov, ki jih hranimo v bazi.
- Preveriti je treba, ali je prišlo med testiranjem transakcije do potrditve, da je transakcija izvedena uspešno. Velikokrat se zgodi, da strežnik pošlje potrditev o opravljeni transakciji, čeprav se nič ne zgodi.
- Preveriti je treba ključne kazalce uspešnosti.
- Pri pisanju skripte za izvajanje testov je bolje, da si za pisanje skripte vzamemo več časa, kot da jo spreminjamo, in tako zaidemo od prvotno zamišljenega scenarija.

2.3.6 Izvajanje zmogljivostnih testov

Izvajanje zmogljivostnih testov je proces, na katerega se ljudje spomnijo, ko slišijo za testiranje spletne aplikacije. Smiselno je, da se proces testiranja vodi s pomočjo tehničnih orodij, in se ob izvajanju kontrolira pravilnost testa. Z upoštevanjem in sledenjem načrtu testiranja izgleda testiranje precej enostavno, kot klikanje po gumbih, a v resnici ni tako. Na testiranje lahko gledamo kot na kombinacijo naslednjih nalog:

- Usklajevanje med izvajanjem testa in usklajevanjem ekipe.
- Potrditi je treba ustreznost konfiguracij, stanja okolja in testnih podatkov.

- Med testiranjem spremljamo potek in nadziramo sistem. Skripte lahko sami potrjujemo, pri avtomatskih testiranjih to počne sam sistem.
- Po končanem testu takoj preverimo rezultate in podamo prve ugotovitve.
- Arhiviramo test, pridobljene podatke in vse ostale podatke, da lahko ponovimo test, če je to potrebno.
- Pregledamo log-datoteko[5]. V njej naj bi se shranil celoten potek testiranja, kar pomaga pri identifikaciji podatkov.

Vhodni-izhodni podatki pri fazi »izvajanju zmogljivostnih testov«

Faza	Vhodni	Izhodni
Izvajanje zmogljivostnih testov	• Izvedba načrtovanega plana testiranja • Orodja, ki so na voljo • Veljavni testi	• Izveden zmogljivostni test • Podatki, ki se generirajo pri testiranju

Tabela 6: Izvajanje zmogljivostnih testov

Pod drobnogledom:

- Preveriti je treba, ali skript uporablja pravilne vrednosti podatkov.
- Če je možno, poženemo test vsaj trikrat. Prvič moramo upoštevati, da se lahko nalagajo knjižnice (DLL) oz. se aplikacija počasi nalaga na predpomnilnik strežnika. Drugo in tretje zaganjanje testa bi moralo biti zelo podobno, če ni, je treba ugotoviti, zakaj je prišlo do razlik in po potrebi ponovno zagnati test.
- Med samim testiranjem je treba beležiti morebitne napake.
- Če testiramo v razvojnem okolju, je treba o tem predhodno obvestiti ostale člane, da si zagotovijo varnostno kopijo[6] ob morebitnih napakah.
- Med izvajanjem testa je priporočljivo izklopiti iskanje virusov, da nam to ne bi pokvarilo podatkov.
- Če ob morebitnih popravkih programske kode ni napredovanja, je učinkoviteje, da se odpravi kakšno spremenljivko v testnih scenarijih, saj so lahko določene spremenljivke krive za slabo izvedbo testa.

2.3.7 Analiza rezultatov, poročilo in ponovno testiranje

Vodstvo in ostali, ki so zainteresirani za testirano spletno aplikacijo, potrebujejo kaj več kot le številke o testiranju. Če stranki predstavimo rezultate testiranj, ne bo razumela ničesar, zato je pomembno, da znamo dobljene rezultate predstaviti na tak način, da jih lahko razume vsak, četudi se s testiranjem ne ukvarja. Priporočene so grafične analize in primerjave s predhodnimi testiranji. Poleg stranke dobi analizo testiranja tudi razvojna ekipa. Na podlagi podatkov se odločijo, ali aplikacija zadostuje dovolj veliki stopnji kakovosti.

[5] Log-datoteka je dnevnik napak.
[6] Ang. Backup.

Preden podatke testiranja predamo ostalim, je treba analizirati podatke. Premisliti je treba, kako. Pri analizi podatkov moramo biti kritični in izbrati tako metodo analize, ki nam bo prikazala realno sliko testa, ne pa izbirati tistih analiz, ki nam bodo spletno aplikacijo predstavile v najlepši luči. Člani ekipe, ki se ukvarjajo s testiranjem, naj bi podatke analizirali individualno, nato naj bi vsak predstavil svojo analizo, skupaj pa bi sestavili poročilo. Če se v poročilu ugotovi preobremenjenost v spletni aplikaciji, je dobro, da še preden problem poročamo programerjem, sami poskušamo odpraviti težave. Po odpravi napake ponovno testiramo aplikacijo. Takoj, ko smatramo, da smo pridobljene podatke analizirali z ustrezno metodo, te posredujemo naprej. Po navadi poročila o testiranju posredujemo razvijalcem, da vidijo, kje so pomanjkljivosti spletne aplikacije, vodstvu, da je seznanjeno s potekom projekta, in oddelku trženja, da o morebitnih zapletih ali o uspešnosti aplikacije obvesti stranko.

Poročila, ki nastanejo po testiranju, lahko razvrstimo v več kategorij:

- **Tehnična poročila**

  Tehnična poročila vsebujejo podatke o projektu. Pri tem mislim na opis projekta, vključno z obremenitvijo, ki je je bila deležna spletna aplikacija na testiranju, in z opisom okolja. Predstavljeni so osnovni scenariji. V zaključnem delu so opisane izjave, predlogi, skrbi itd. sodelujočih pri implementaciji in testiranju.

- **Poročila posameznih skupin**

  Pri poročilih, ki jih izda posamezna skupina, lahko pričakujemo podrobnejše informacije o določenih dejavnikih v testiranem projektu.

  - Merila, na katera se nanašajo podatki testiranja
  - Vizualne predstavitve najpomembnejših podatkov
  - Kratek povzetek z uporabo grafov
  - Razni podatki o izvedenem testu

Ključ do učinkovitega poročanja je predstaviti informacije v takem obsegu, da jih bo naslovna publika hitro in enostavno razumela, pri čemer je najpomembnejše vprašanje, kdo je naslovna publika in kakšno znanje ima, temu primerno mora biti opremljeno tudi poročilo.

Vhodni-izhodni podatki pri fazi »analize podatkov«

Faza	Vhodni	Izhodni
Analiziranje rezultatov, poročil in ponovno testiranje	• Rezultati testiranja • Zmogljivostna merila sprejemljivosti • Tveganja, avtomatski testi	• Rezultati analize • Priporočila • Poročila

Tabela 7: Analiziranje rezultatov, poročil in ponovno testiranje

Pod drobnogledom:

- Poročilo mora biti oddano dovolj zgodaj, da se udeleženci projekta lahko organizirajo o nadaljnjem poteku projekta.
- Poročilo naj poleg besedila vsebuje grafične vmesnike.
- Poročilo naj bo intuitivno.
- Uporaba pravilne statistične metode in predstavitev podatkov na učinkovit način.
- Pisanje mora biti kratko in jedrnato.

2.4 Zakaj izvajati zmogljivostne teste

Pri razvoju spletnih aplikacij preobširno testiranje zmogljivosti spletne aplikacije s strani vodstva ni zaželeno. Temeljni razlogi za takšno razmišljanje so previsoki stroški testiranj, poraba časa, zamude pri predaji produkta stranki itd. Razloge, zakaj je testiranje koristno, bom predstavil v naslednjih štirih sklopih.

Ocena za javnost

Testiranja nam predstavijo, koliko je vreden naš izdelek na realnem trgu, kar nam omogoča napovedovanje rasti ali padanje uporabe spletne aplikacije. S pomočjo napovedi se tako lažje odločamo o tem, ali aplikacija potrebuje boljšo strojno opremo za delovanje, in ali je treba aplikacijo le grafično »olepšati«. S pridobljenimi podatki ugotovimo, kakšna je kakovost izdelka. Na podlagi kakovosti lahko ocenimo povpraševanje po spletni aplikaciji, iz povpraševanja pa lahko izračunamo, kakšen bo dobiček od prodaje.

Ocena ustreznosti arhitekture

Ocena ustreznosti arhitekture pomeni predvsem to, da zmoremo oceniti ustrezno zmogljivost spletne aplikacije. Ob oceni ustreznosti pridobimo tudi oceno stabilnosti. Stabilna aplikacija daje uporabniku varnost, razvijalcem pa možnost nadaljnjega razvoja v smeri razvoja različnih konfiguracij in odkrivanja novih funkcionalnosti.

Ocena zmogljivosti programske opreme

Ocena zmogljivosti programske opreme je prav tako pomembna kot ocena arhitekture. Pri programski opremi preverjamo, koliko je aplikacija zmogljiva. Pomemben faktor pri aplikacijah je uporabniški vmesnik, ki mora biti do uporabnika čim bolj prijazen. Preverjanje zgornjih dveh faktorjev je lahko ključno za zadovoljstvo uporabnikov.

Projektno ozadje

Za uspešno izvajanje testiranja je zelo pomemben pristop. Brez razumevanja vsebine projekta je izvajanje kakršnih koli testov nesmiselno. Pri testiranju morajo biti prisotni vsi sodelujoči, pa tudi ostali, ki so pasivno vključeni v projekt, da se določijo, katere funkcije je treba testirati. Če pri testiranju sodeluje le testna ekipa, se mora sama odločiti, kaj je na določenem projektu vredno testirati.

2.5 Zmogljivostno, bremensko in stresno testiranje

Zmogljivostne teste izvajamo zato, da preverimo hitrost izvajanja, stresno izvajanje in pravilnost delovanja spletne aplikacije. Zmogljivost testiranja (ang. performance testing) lahko razdelimo v tri kategorije testiranja spletnih aplikacij. Poznamo test zmogljivosti, bremensko testiranje in stresno testiranje. Poleg zgoraj omenjenih testov je treba testirati pri spletni aplikaciji tudi druge stvari, kot so varnost, združljivost, odzivnost itd., a tega ne bom opisoval, ker je moj namen testirati zmogljivost spletne aplikacije.

2.5.1 Test zmogljivosti

Zmogljivostni test nam pove, koliko uporabnikov oz. koliko transakcij, ki jih podpira spletna aplikacija, lahko opravimo naenkrat, in pri tem aplikacijo obremenimo do te mere, da ne povzroča nobenih zamud. Kapaciteta testiranja poteka v povezavi z načrtovanjem zmogljivosti. Načrt zmogljivosti se je pričel graditi v fazi prepoznavanja okolja in se je dokončno izoblikoval v fazi načrtovanja. Zmogljivostni testi nam podajo odgovor o morebitnih pomanjkljivostih v arhitekturi.

2.5.2 Bremenski testi

Cilj bremenskih testov je preveriti obnašanje aplikacije pri normalnih obremenitvah in v primeru maksimalne predpisane obremenitve. Bremensko testiranje se izvaja za preverjanje tega, ali aplikacija izpolnjuje zastavljene zmogljivostne kriterije. Kriteriji so po navadi definirani v pogodbi med izvajalcem in naročnikom. Bremenski test nam omogoča meritev odzivnega časa, prepustnost in porabo virov. Omogoča tudi identifikacijo prelomne točke, v primeru, da se ta zgodi pred maksimalno predpisano obremenitvijo.

Vzdržljivostno testiranje je podmnožica bremenskega testiranja. Osredotoča se na določanje ali potrjevanje obnašanja aplikacije, v daljšem časovnem obdobju.

Vzdrževalne teste lahko uporabimo za izračun povprečnega časa med odpovedmi (MTBF) in povprečnega časa do prve odpovedi (MTTF).

2.5.3 Stresni testi

Določajo ali potrjujejo obnašanje spletne aplikacije, ko obremenitev preseže normalne oz. maksimalne predpisane vrednosti.

Cilj stresnega testiranja je odkrivanje napak v aplikaciji, ki se pojavijo pod pogoji visoke obremenitve. Napake lahko vključujejo sinhronizacijske probleme ali puščanja pomnilnika

(angl. memory leaks). Zmogljivostno testiranje nam omogoča identifikacijo šibkih točk in prikaže obnašanje aplikacije v ekstremnih obremenitvah.

Test hitrosti je podmnožica stresnega testiranja. Osredotoča se na določena obnašanja aplikacije, ko obremenitev za kratek čas preseže maksimalne predpisane meje.

2.5.4 Pridobitve pri posameznih testnih kategorijah

Vsaka kategorija zmogljivostnih testov nam poda določen nabor podatkov o naši informacijski rešitvi (aplikaciji in infrastrukturi). V nadaljevanju so predstavljeni podatki pri posameznih kategorijah zmogljivostnih testov.

Koristi, ki jih pridobimo s testiranjem zmogljivostnih testov spletne aplikacije, so navadno večje od stroškov, ki nastanejo ob zmogljivostnem testiranju. Zaradi negotovih rezultatov testiranja veliko podjetij dvomi o smotrnosti zmogljivostnih testiranj, to je posledica nepravilnega postopka pri testiranju spletnih aplikacij [3]. Kljub temu praksa kaže na to, da zmogljivostno testiranje, čeprav nedosledno, pripomore k zmanjševanju izpadov sistema v produkciji. Prav tako nam pomaga določiti kritično skupino parametrov, ki nas opozarjajo na poslabšano delovanje aplikacije, ali na neposredni izpad sistema.

- Zmogljivostni testi
  - Koristi:
    - Določajo hitrost, prilagodljivost in stabilnost aplikacije, kar zagotavlja podporo pri poslovnih odločitvah.
    - Osredotočajo se na uporabnikovo zadovoljstvo z odzivnimi časi aplikacije.
    - Določajo neskladnost med realnimi in pričakovanimi zmogljivostmi.
  - Nepredvidena področja:
    - Možno je, da ne odkrijemo nekaterih funkcionalnih napak, ki se pojavljajo le pod obremenitvijo.
    - V primeru, ko se testi izvajajo na produkcijskim strežniku in je breme generirano iz dejanskih uporabniških računalnikov, bo vedno nekaj odstopanj v rezultatih.
- Bremenski testi
  - Koristi:
    - Določajo potrebno zmogljivost virov za produkcijsko okolje.
    - Določajo ustreznost strojne opreme.
    - Odkrivajo težave pri sočasnem izvajanju transakcij.
    - Zbirajo podatke z namenom planiranja razširljivosti.
    - Pomagajo določiti število sočasnih uporabnikov, preden je ogroženo delovanje aplikacije.
    - Pomagajo določiti količino obremenitve, ki je potrebna, da postane uporaba virov s strani sistema kritična.
  - Nepredvidena področja:
    - Niso namenjeni zgolj določitvi hitrosti odziva.
    - Rezultate uporabljamo za primerjavo z drugimi sorodnimi testi.
- Stresni testi

- Koristi:
  - Določajo, ali pride pri prekomerni obremenitvi sistema do nekonsistentnosti oz. napak v podatkih.
  - Omogočajo ocenitev obremenitve aplikacije, pri kateri poleg upočasnjenega delovanja začne aplikacija tudi nepravilno delovati.
  - Zagotavljajo, da se z obremenitvijo ne odpirajo varnostne luknje-
  - Definirajo neželene učinke izpadov strojne in programske opreme.
- Nepredvidena področja:
  - Ker so stresni testi nerealni glede na realen model uporabe, lahko nekatere stranke zavržejo rezultate testov.
  - Težko je vedeti, kakšno obremenitev je smiselno testirati .
  - Če testno okolje ni izolirano, lahko povzročimo izpad omrežja ali nepravilno delovanje ostalih aplikacij .

2.6 Prednosti, slabosti in smernice pri izvajanju avtomatskih testov

Zakaj izvajati avtomatske teste? V dobi hitrega razvoja v vse bolj kompleksnih okoljih predstavljajo avtomatski testi še dodaten korak pri vzpostavljanju kakovosti produktov.

Avtomatizacija je uporaba strategij, instrumentov in artefaktov, ki zmanjšujejo potrebo po ročni ali človeški vpletenosti.

V zgodovini razvoja programske in spletne opreme je bilo veliko žalostnih dogodkov. Dogodilo se je, da je po uspešnem testiranju opreme sistem na produkciji ni deloval kot bi moral. Škoda je bila v večini primerov ogromna. Avtomatski testi funkcionalnosti pogosto nadomeščajo ročno testiranje. Pred avtomatsko izvedbo zmogljivostnih testov je treba posneti teste in jih hraniti za kasnejše izvajanje avtomatskih testov. Ko shranimo postopek avtomatskega testiranja, se test požene še enkrat. Pridobljene podatke in dokumentacijo o uspešnosti izvajanega testa inženirji preučijo in odpravijo morebitne napake. Ko se odpravi napake, se avtomatsko požene že prej pripravljen test, ki ponovno testira aplikacijo. Opisani proces se ponovi tri- do petkrat, da je raven procesa zadovoljiva.

Vrste testiranj, ki jih lahko avtomatiziramo

- Funkcijsko testiranje – testiranje sistema, da deluje v skladu s pričakovanji
- Regresijska testiranja – testiranje ob dodajanju novih funkcij, ki ne spremenijo sistema
- Stresni testi – določajo absolutno zmogljivost opreme in aplikacije
- Zmogljivostno testiranje – zagotavljanje ustreznosti določenih standardov
- Testi nalaganja – določanje točke, na kateri bo treba nadgraditi zmogljivost in delovanje sistema

Odločitev, ali uporabiti orodja za avtomatsko testiranje, je precej težka. Zavedati se je treba, kakšna bi bila naložba, povezana z nakupom orodja in ustrezno usposobljene ekipe za njihovo učinkovito in uspešno izvajanje testiranj.

Prednosti avtomatskih testov:

- **Točnost in popolnost izvedbe**

  Z avtomatskim testiranjem programske oz. spletne opreme imamo zagotovilo, da bodo testni scenariji v celoti izvedeni pravilno in natančno. Poleg tega so testi ponovljivi in lahko ugotovimo, kako se programska oprema obnaša na ponavljajoče se izvajanje. Ročno izvajanje testov je izpostavljeno utrujenosti in nizki koncentraciji inženirja. Posledično se lahko zgodi, da napačno izvedeni testni scenariji dajo neuporabne rezultate.

- **Natančen izpis dnevnika napak**

  Avtomatski testi so sprogramirani tako, da v primeru napake sistem zapiše napako v dnevnik napak (angl. log). Pri ročnem testiranju nimamo podobnega dnevnika, zato mora inženir, ki se ukvarja s testiranjem, sam prepoznati napako.

- **Celovitost informacij**

  Ko so podatki avtomatskih testov enkrat shranjeni v zbirki podatkov, so na voljo kadar koli jih inženir, ki se ukvarja s testiranjem aplikacije, potrebuje.

- **Krajši čas trajanja testiranja**

  Avtomatska testiranja so navadno precej krajša kot ročna testiranja. Obenem se lahko avtomatski testi poganjajo neprekinjeno 24 ur na dan, sedem dni v tednu. Pri ročnem izvajanju testov za potrebujemo ekipo, ki se s tem ukvarja profesionalno.

- **Popolnost regresijskih testov**

  Zaradi pomanjkanja sredstev, časa in delovne sile je ročno izvajanje regresijskih testov zelo naporno. Avtomatski testi imajo zato na tem področju veliko prednost. Za izvajanje testiranj ne potrebujejo veliko časa in ne delovne sile. Avtomatizacija bistveno zmanjša tveganje, ki pri dodajanju novih funkcionalnosti ne ogroža samega sistema.

- **Možnost izvajanja testiranj**

  Sprogramiramo lahko zapletene testne scenarije in obenem ne ogrožamo podatkov o implementaciji spletne aplikacije.

  Avtomatizacija nam omogoča izvajanje testov nalaganja in testov učinkovitosti za srednja in velika podjetja. Take teste je skoraj nemogoče izvajati ročno.

Slabosti avtomatskih testov:

- **Visoka naložba, potreben je nakup orodja in izobraževanje**

  Organizacija, ki se odloči za izvajanje avtomatskega testiranja, mora vlagati v programski razvoj in dodatno usposabljanje osebja za izvajanje avtomatskih zmogljivostnih testov. Kljub precej lažji uporabi avtomatskih testiranj, je izobraževanje osebja dolgo, kar je za organizacijo velik strošek.

- **Dodatni stroški**

  V primeru, ko osnovni paket za izvajanje avtomatskih testov ne ustreza sistemu, na katerem se izvaja test, moramo prilagoditi ali dokupiti takšno orodje, ki bo našemu sistemu ustrezalo.

- **Težave pri pripravi testov**

  Dokler osebje še nima izkušenj, pa tudi kasneje, je priprava avtomatskih testov precej zahtevna. Problem lahko nastane že pri napaki v skripti. V tem primeru se test izvaja napačno in lahko pride do hudih težav (npr. napačnih končnih podatkov o stanju aplikacije).

- **Pokritost avtomatskih testov**

  Trenutna avtomatizacija programske opreme ne zajema celotne palete orodij in vrste aplikacij. Inženirji rešujejo te probleme tako, da, kjer je možno, izvajajo avtomatske teste, kjer pa to ni možno, preidejo na ročno testiranje.

Nekatere od zgoraj navedenih pomanjkljivosti pogosto povzročajo ogromno težav, kljub temu pa je avtomatizacija testiranja zaželena.

3. PREGLED ORODIJ ZA IZVAJANJE ZMOGLIVOSTNIH TESTOV

Za potrebe testiranja spletne aplikacije ePero sem potreboval orodje, ki bi mi omogočilo izvajanje avtomatskih zmogljivostnih testov. Do trenutka, ko se z avtomatskimi testi spletne aplikacije nisem začel ukvarjati sam, se pri aplikaciji ePero ni še nihče. Orodij za izvajanje avtomatskih testov je na spletu ogromno. Poiskati sem moral le pravo, takšno, ki bo služilo mojemu namenu. Večina dobrih orodij za testiranje spletnih aplikacij je dostopnih na spletu le s plačilom. Žal je bila moja naloga ta, da poiščem tako orodje za testiranje, ki je brezplačno. S pojmom dobro orodje mislim predvsem to, da lahko z njim opravljamo najrazličnejše zmogljivostne teste.

Lastnosti dobrega orodja:

- podpora čim večjemu številu protokolov,
- kratek čas zagona orodja,
- uporabnost,
- enostavno za pisanje testnih skript,
- hitro nalaganje in poganjanje testnih skript,
- pregleden izpis rezultatov,
- možnost shranjevanja rezultatov in testnih skript.

Pogosto se orodja za testiranje, ki jih pridobimo proti plačilu, ločijo od brezplačnih orodij le po tem, da plačljiva orodja uporabljajo lepši grafični vmesnik. Lepši grafični vmesnik nam ne pomaga pri zmogljivosti naše spletne aplikacije, vendar nam je v veliko pomoč pri izvajanju avtomatskih zmogljivostnih testov in pri nadzoru med izvajanjem testov. Prednost plačljivih orodij je ta, da dobljene rezultate uredijo v preglednice in avtomatsko ustvarijo izčrpno poročilo s kazalniki kakovosti, tako da oseba, ki se ukvarja s testiranjem spletne aplikacije, hitro dobi želene informacije o zmogljivosti spletne aplikacije. Prednost orodij, ki jih dobimo zastonj, je ta, da za njihovo uporabo ni treba plačati. Pogosto so taka orodja odprtokodna, kar pomeni, da lahko orodje modificiramo in prilagodimo lastnim potrebam. Pri brezplačnih orodjih je grafični vmesnik slab ali pa ga celo ni.

Vsa orodja za avtomatsko testiranje zmogljivostnih testov predvidevajo potek testiranja v treh fazah. V prvi fazi posnamemo skripto, ki opisuje določen testni scenarij dogajanja na spletni aplikaciji. V drugi fazi pridobljeno skripto popravimo, dodamo morebitne nove spremenljivke itd. To je odvisno od orodja, koliko nam omogoča spreminjati ali dodajati nove zahteve. Nato poženemo test. V tretji fazi nam orodje generira ali pa le predstavi dobljene podatke iz samega testiranja, spet odvisno od posameznih orodij.

3.1 Predstavitev orodij za avtomatsko testiranje

3.1.1 Forecast

Forecast je orodje, s katerim lahko testiramo obremenitev spletne aplikacije in merimo njeno uspešnost. Z omenjenim orodjem lahko testiramo: spletne aplikacije, J2EE, .NET, strežnik/odjemalec itd. Z njim lahko izvajamo testne skripte, ki so zapisane v Javi, C++ , C# in VB.net. S Forecastom je preprosto izvajati aplikacijsko snemanje. Za ustvarjanje in izvajanje novih testnih skript je zelo preprost in ima nizke strojne zahteve [2].

3.1.2 LoadTracer

LoadTracer [Slika 3] je orodje, ki uporablja grafični vmesnik za testiranje zmogljivosti, obremenitve in funkcionalnost spletne aplikacije. Orodje nam omogoča kreiranje več virtualnih klientov, ki bodo ob določenem času klicali spletni strežnik. V okviru različnih obremenitvenih ali stres-testiranj, preverjamo učinkovitost spletne aplikacije. Celoten proces testiranja je razdeljen v tri faze, snemanje skript, simulacijo testov in analizo pridobljenih rezultatov.

Slika 3: Logotip testnega orodja LoadTracer

- LoadTracer – snemalnik

  Snemalnik deluje kot proxy strešnik med spletnim brskalnikom in spletnim strežnikom. Proxy evidentira vse interakcije med brskalnikom in strežnikom v obliki skriptne datoteke. Snemalnik podpira parametrizacijo, ki omogoča pridobivanje trenutne vrednosti med izvajanjem.

- LoadTrancer – simulator

  Simulator je sposoben ustvariti do več tisoč virtualnih uporabnikov, vendar je število uporabnikov odvisno od naših zahtev in težavnosti testne skripte. Simulator nam omogoča, da vsaka izvedena skripta uporablja svoje nastavitve. Npr. število virtualnih uporabnikov, ponavljanj testa, zaporedje izvajanj skript, ponuja celo možnost avtomatskega generiranja piškotkov.

- LoadTracer – analiza rezultatov

  Po opravljenih testih nam orodje samodejno generira različne vrste grafov in poročil za namen analize. Generirana poročila je možno izvoziti v HTML-formatu.

3.1.3 OpenLoad

OpenLoad je enostavno orodje za uporabo na osnovi brskalnika, ki omogoča optimizacijo spletnih aplikacij za funkcijsko in obremenitveno testiranje. Omogoča nam avtomatsko testiranje, kar zagotavlja hitro odpravljanje napak in visoko stopnjo kakovosti. Podobno kot ostala plačljiva testna orodja ima tudi ta lepo razvit grafični vmesnik. Orodje je možno preizkusiti le v okrnjeni različici, saj je njegova celotna uporaba možna le z nakupom [7].

3.1.4 JMeter

JMeter [Slika 4] je namizna javanska aplikacija za izvajanje avtomatskih zmogljivostnih in obremenitvenih testov. Aplikacijo se uporablja za merjenje in analiziranje pridobljenih podatkov tipa odjemalec - strežnik. Jmeter ima možnost testiranja dostopa do podatkov preko JDBC, LDAP, spletnega strežnika ter testiranje aplikacij preko Interneta, TPC-, POP3-, IMAP-, FTP-, http-, HTTPS-protokolov. Jmeter je brezplačno in odprtokodno orodje, zato ga lahko prilagajamo in nadgrajujemo. Za prilagajanje in nadgradnjo je treba poznati programski jezik Java.

Slika 4:Logotip programskega orodja JMeter [1].

Sestavni del orodja JMeter so tudi razširitve. Najpomembnejše in najuporabnejše razširitve so za predstavitev, analizo dobljenih rezultatov in časovno generiranje istočasnih navideznih uporabnikov.

Na prvi pogled zgleda JMeter kot brskalnik. Čeprav lahko v njem pregledujemo spletne strani, pa ne podpira vseh tehnologij tipičnih brskalnikov. Za razliko od brskalnikov ne podpira Java-Scripta, ki se danes najpogosteje uporablja pri programiranju spletnih aplikacij na strani odjemalca. Zato morajo poskrbeti testerji, ki morajo »na roke« prepisati ustrezne parametre v razpoložljive dokumente [1].

3.1.5 vTest

vTest je orodje za avtomatsko funkcionalno in regresijsko testiranje spletne aplikacije. Omogoča temeljito preverjanje spletne aplikacije v različnih okoljih. Poveča produktivnost z uporabo avtomatiziranih testnih skript, ki ustvarjajo poročila o opravljenih testih.

Testne skripte se lahko samodejno izvedejo z minimalnim človeškim posredovanjem. Avtomatski testi se lahko izvajajo na podlagi načrta o avtomatskem testiranju. To po navadi pripomore k večji produktivnosti in omogoča, da se ljudje osredotočajo na druge faze razvoja spletne aplikacije [12].

3.1.6 QEngine

QEngine je avtomatizirano orodje, ki podpira obremenitveno testiranje in funkcijsko testiranje spletnih aplikacij. Poleg podpore za testiranje spletnih aplikacij omogoča testiranje SOAP Web Services. Program omogoča in zagotavlja popolno delovanje celotne spletne aplikacije. Z njim lahko testiramo izvajanje aplikacije pod visokimi obremenitvami in to dalj časa, kar simulira situacijo polne obremenitve med delovnim dnem. Pomaga nam pri odkrivanju ozkih grl med izvajanjem testiranja zmogljivosti spletne aplikacije. Po zaključenem testiranju zmogljivosti si lahko s pridobljenimi rezultati pomagamo določiti seznam zahtev, ki so še potrebne za učinkovito delovanje aplikacije [5].

3.1.7 WAPT

WAPT[7] 7.1 je program za izvajanje testov nalaganja in obremenitev. WAPT nam omogoča enostavno uporabo skupaj z uporabo z grafičnim vmesnikom. Z orodjem lahko testiramo spletne strani, spletne strežnike, spletne aplikacije in njene vmesnike. WAPT vsebuje nabor funkcij, ki nam omogočajo testiranje dinamičnih spletnih vsebin in varovane HTTPS-strani. Z orodjem je možno generirati do več tisoč uporabnikov in testirati zmogljivost spletne aplikacije. Program je plačljiv, možno pa je testirati njegovo okrnjeno različico za obdobje tridesetih dni [13].

3.1.8 Scapa Test and Performance Platform

»Scapa Test and Performance Platform« je hitro in poceni orodje za izvajanje zmogljivostnih testnih primerov, diagnoze in za spremljanje proizvoda. Zagotavlja nam pregled delovanja sistema in značilnosti uporabniške izkušnje. Platforma deluje tako, da simulira več uporabnikov računalniškega sistema, tako da bi dobili čim boljši približek realnemu stanju. Namen uporabe orodja je zmanjšati stroške proizvodnje produkta.

---

[7] Program za izvajanje avtomatskih zmogljivostnih testov (Ang. Web Application Load, Stress and Performance Testing).

3.1.9 OpenSTA

»OpenSTA« je orodje, ki ga uporabljajo za kontrolo rezultatov pri posameznih zmogljivostnih testih. Testiranje se prične tako, da najprej posnamemo želeni scenarij, ki ga bomo kasneje testirali. Posnetki se izvajajo v lastnem testnem brskalniku, tako da je potreba po uporabi drugih brskalnikov odveč. V istem brskalniku je nato možno ustvariti in poganjati različne testne skripte. Ustvarjene scenarije se lahko nato poljubno predvaja in simulira na poljubnem številu uporabnikov. Z uporabo teh procesov smo lahko sigurni, da bomo ustvarili realne teste, ki nam bodo podali pravilne rezultate za izvajanje testov [8].

3.1.10 Siege

»Sigle« je http obremenitveno orodje, ki je nato postal program za izvajanje testnih scenarijev. Zasnovan je bil predvsem zato, da bi razvijalci spletnih strani uredili svojo kodo. Siege podpira piškotke, HTML- in HTTPS-protokole [11].

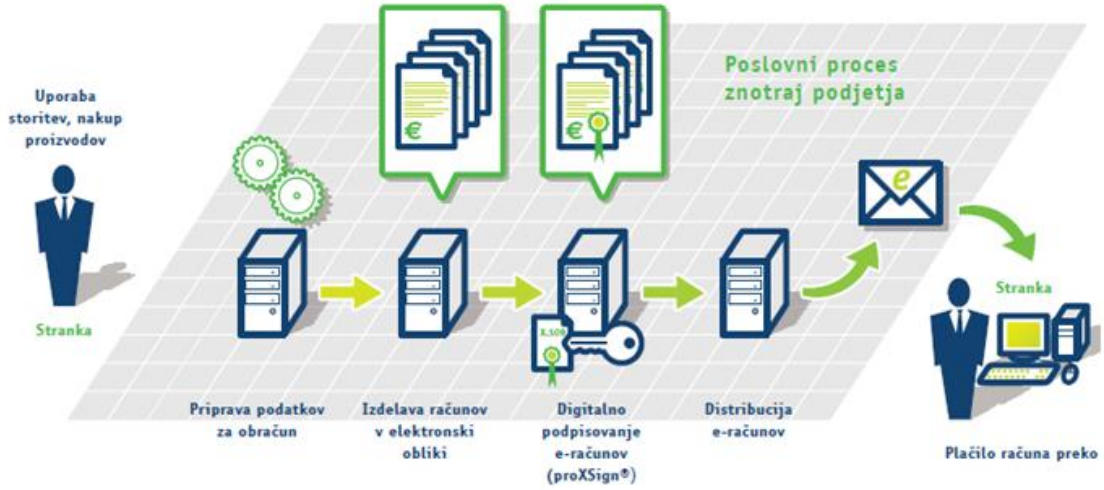
3.1.11 Pylot

»Pylot« je odprtokodno orodje za testiranje zmogljivosti in razširjenosti spletnih storitev. Z orodjem lahko izvajamo http-obremenitvene teste, ki so koristni za načrtovanje zmogljivosti, primerjalno analizo in za sistemsko uravnavanje. Pylot uporabljamo tako, da določimo testne primere v datoteki XML. Orodje lahko upravljamo tako iz orodne vrstice kot preko grafičnega vmesnika. Program nam omogoča, da sledimo izvajanju testnih scenarijev med samim izvajanjem testa. Po končanem avtomatskem testu nam generira poročilo o uspešnosti opravljenega testa [9].

4. PREDSTAVITEV PROBLEMA

4.1 SETCCE proXSign

Družina produktov proXSign vsebuje vtičnike za digitalno podpisovanje, časovno žigosanje ter asimetrično šifriranje PDF- in XML-dokumentov. Z uporabo te komponente, pod katerimi ima elektronska oblika enakopravno veljavnost kot pisna, dosežemo dematerializacijo poslovnih procesov ter uvedbo brezpapirnega poslovanja. Dokumente, ki so podpisani digitalno, je mogoče z digitalnim podpisom ohraniti izključno v elektronski obliki in takšne tudi uporabiti [Slika:5].



Slika 5: postopek proXSign

4.2 SETCCE ePero

Spletna aplikacija SETCCE ePero® se uporablja za elektronsko lastnoročno podpisovanje različnih dokumentov in obvladovanje dokumentnih delotokov. Splošna rešitev za elektronsko podpisovanje dokumentov se imenuje SETCCE ePero®.

Na splošno o rešitvi SETCCE ePero®

Jedro aplikacije je elektronsko podpisovanje PDF-, PDF/A-dokumentov. Podpisovanje poteka po naslednjem postopku: podpisnik se lastnoročno podpiše na podpisno napravo [Slika 6]. Aplikacija SETCCE ePero® zajame sliko podpisa iz podpisne naprave. Slika podpisa se prenese na strežnik. Na strežniku se slika vstavi v PDF-dokument in dokument se digitalno podpiše s kvalificiranim strežniškim digitalnim potrdilom podjetja. SETCCE ePero odpravi tradicionalno zamudno upravljanje s papirnimi dokumenti znotraj podjetja, kar pogosto

predstavlja vir napak. Elektronski dokumenti, podpisani z ePeresom, so nemudoma na razpolago zalednim aplikacijam in delavcem za nadaljnje obdelave in vpoglede.

Slika 6: Elektronsko podpisovanje

Za uporabo rešitve ePero so se ogreli predvsem mobilni operaterji, zavarovalnice, vzajemni skladi in banke. Gre za organizacije z razvejanimi prodajnimi mrežami in agenti na terenu. Že podpisane elektronske dokumente lahko podjetje za stranko še vedno natisne, medtem ko za lastne potrebe ohranja dokumente izključno v elektronski obliki.

Spodnja slika [Slika 7] prikazuje proces elektronskega podpisovanja dokumentov s pomočjo rešitve SETCCE ePero®.

Slika 7: Prikaz procesa elektronskega podpisovanja

Primerjava med komponentami SETCCE proXSign in rešitvijo SETCCE ePero®

S pomočjo komponent SETCCE proXSign lahko digitalno podpišemo PDF-, PDF/A-dokumente z osebnim digitalnim potrdilom.

Digitalni podpis z osebnim digitalnim potrdilom daje dokumentu pomembno težo in veljavnost, vendar pri tem obstaja določena omejitev, treba je pridobiti kvalificirano digitalno potrdilo. S spletno aplikacijo SETCCE ePero® se zgornja omejitev v veliki meri skrči. Elektronski podpis je ustvarjen s pomočjo elektronskega peresa, uporabnik pa ne potrebuje osebnega digitalnega potrdila. Zadostuje lastnoročen podpis na digitalno tablico, zajet pod določenimi organizacijskimi pravili in zaščiten pred spremembami. Podpis je z uporabo rešitve SETCCE ePero® ustrezno zaščiten, saj se dokument digitalno podpiše s strežniškim digitalnim potrdilom podjetja. To potrdilo je varno hranjeno na strežniku ePero in ustrezno zaščiteno.

Rešitev ePero je plod sodelovanja podjetja SETCCE z avstrijskim podjetjem XYZMO/ SIGNificant[8].

4.3 Predstavitev spletne aplikacije ePero

V tem poglavju sem se odločil podrobneje predstaviti spletno aplikacijo ePero, kar nam bo služilo za razumevanje izvajanih avtomatskih zmogljivostnih testov v petem poglavju.

Osnovni namen spletne aplikacije ePero je elektronski podpis PDF-dokumentov in vnos podpisanih dokumentov v strankin dokumentni sistem. Poleg omenjenih funkcij lahko spletna aplikacija natisne podpisane dokumente, jih pošlje stranki po elektronski pošti ali vnese prilogo k podpisanemu dokumentu.

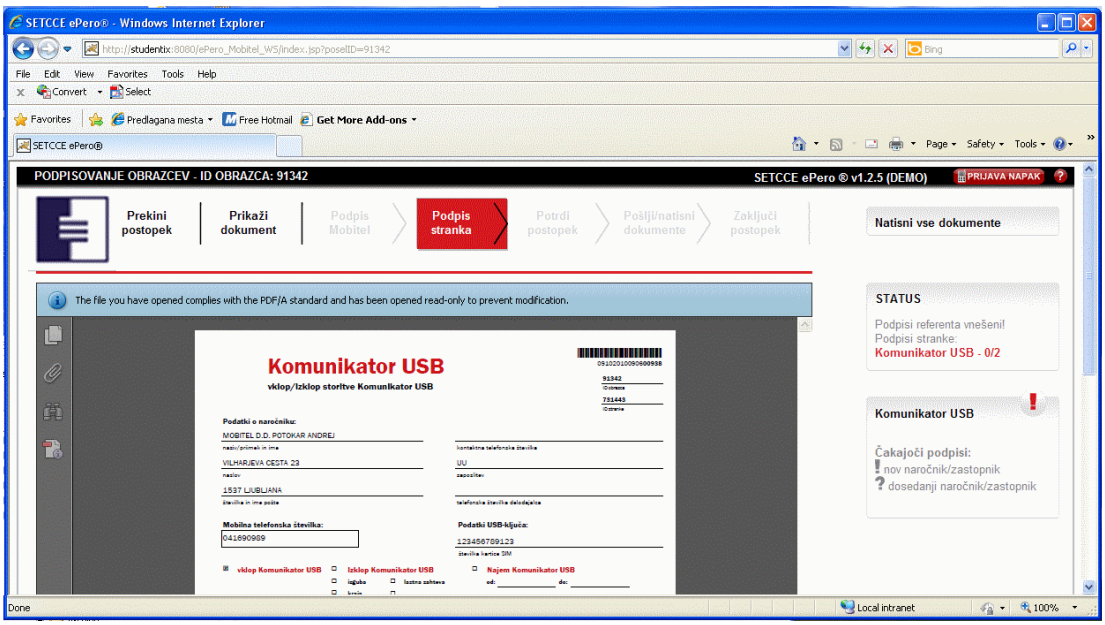
GUI ePero

Glavna stran spletne aplikacije ePero je razdeljena na pet gradnikov:

- glava,
- statusno okno,
- postopek obdelave posla,
- seznam dokumentov v poslu,
- prikaz izbranega dokumenta.

Statusno okno [Slika 8] se nahaja v desnem zgornjem kotu in nam služi za kontrolo, saj ob vsakem trenutku vemo, kaj smo storili in kateri je bil zadnji korak. Postopek obdelave posla nas vodi skozi podpisni postopek. V pomoč nam je seznam poslov, ki nas seznanja s tem, koliko dokumentov moramo podpisati.

[8] Patent za elektronsko podpisovanje.



Slika 8: Prikaz spletne aplikacije ePero

4.3.1 Arhitektura spletne aplikacije ePero in uporaba spletnih storitev

Spletne storitve, nekateri jih imenujejo aplikacijske storitve, so storitve, ki pogosto vključujejo podatke, programsko in strojno opremo, ki komunicira preko omrežja. Pogosto pa spletne storitve ne vključujejo le strojne in programske opreme, ampak je v postopek vključen tudi človek. Če poenostavimo, so spletne storitve aplikacijske komponente. Delo s spletnimi storitvami ne pozna meja. Uporabljamo jih lahko v velikih sistemih, za shranjevanje velikih količin podatkov, komunikacijo s strankami itd. Lahko pa jih uporabimo pri manjših sistemih, za preverjanje ponudb in manjših stvari.

Uporabniki lahko dostopajo do nekaterih spletnih storitev prek dogovora »peer-to-peer[9]«. Določene storitve lahko komunicirajo z drugimi službami. To izmenjavo podatkov in postopkov na splošno omogoča razred programske opreme, znan kot »middleware[10]«. Spletne storitve so omogočene z uporabo XML, saj uporabljajo predpisani format, ki velja za izmenjavo podatkov.

4.3.1.1 ePero odjemalec

Podjetje SETCCE je za uporabljanje spletne aplikacije ePero razvilo paket, ki na računalnik namesti naslednje komponente [Slika 9]:

- ePero odjemalec,
- SIGNificant Server Tablet Maneger,
- SETCCE proXSign PDF,

[9] Storitev, ki omogoča uporabnikom deljenje datotek prek interneta.
[10] Programska oprema, ki omogoča aplikaciji ali komponentam aplikacije povezovanje in izmenjavo podatkov z drugimi aplikacijami/komponentami.

- Dynamic Web TWAIN,
- ActiveX komponento.

Poleg komponent potrebujemo na odjemalcu še elektronsko tablico.

Odjemalec je bila aplikacija, ki je dostopala do storitev na drugem računalniškem sistemu, znana kot strežnik, s pomočjo internetne povezave. Izraz odjemalec se je uporabljalo za naprave, ki niso bile sposobne prikazati svojih programov, ampak so potrebovale interakcijo z oddaljenim računalnikom, imenovanim strežnik.

Model odjemalec/strežnik se uporablja še danes, a s to razliko, da danes odjemalec in strežnik tečeta na istem računalniku s pomočjo vtičnikov Unix.

Primeri uporabe odjemalca/strežnika:

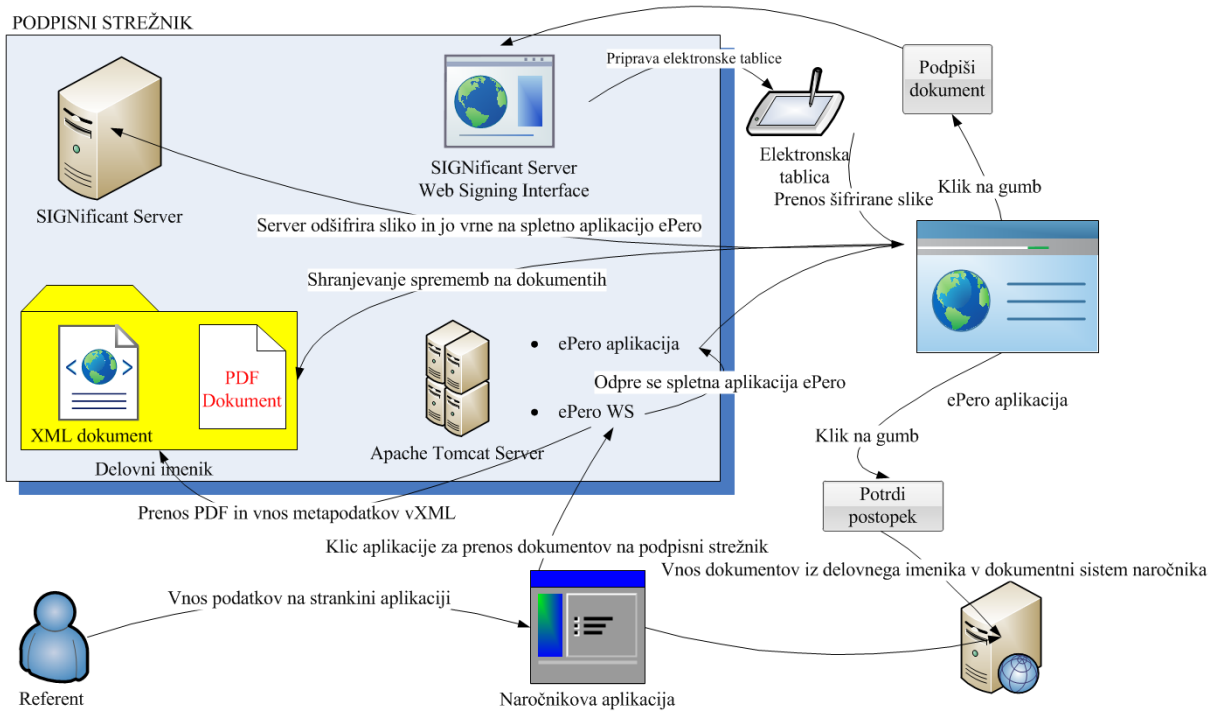
Uporabnik, kise poslužuje internetnih vtičnikov, se lahko poveže na oddaljen računalnik. Spletni brskalniki so klienti, ki se povezujejo na spletne strežnike, ti pa jim vračajo spletno vsebino, ki jo spletni brskalnik prikaže uporabniku. Ogromno uporabnikov koristi e-mail klienta za spletno pošto. Klient za elektronsko pošto od strežnika pošlje zahtevano pošto na naš poštni portal.

Obstoječi odjemalci se menjujejo s spletnimi brskalniki, tako da je brskalnik neke vrste odjemalec. Ta izboljšava pripomore k temu, da nam ni treba prenašati velikih programov za poganjanje aplikacij na novih računalnikih. Primer take rabe je WebMail.

4.3.1.2 Podpisni strežnik

Na podpisnem strežniku imamo nameščene:

- SIGNificant Server,
- Apache Tomcat,
- SIGNificant Server Web Signing Interface.



Slika 9: Podpisni strežnik

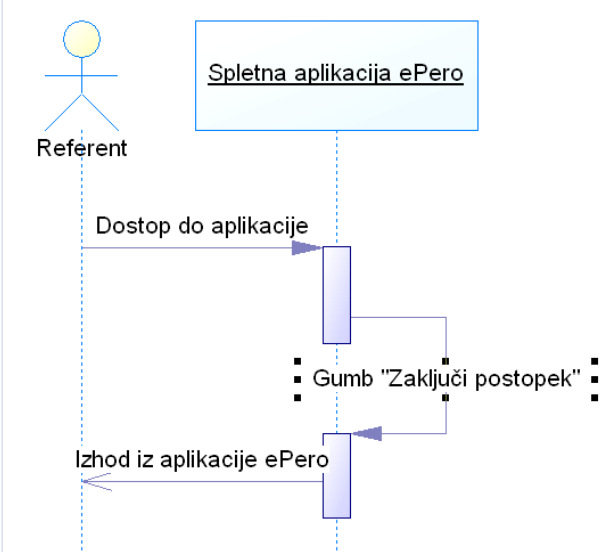
Spletni strežniki so računalniki, ki dostavljajo spletne strani. Vsak spletni strežnik ima svoj IP-naslov in ime domene. Vsak računalnik lahko spremenimo v spletni strežnik z namestitvijo strežniške programske opreme in povezavo računalnika z internetom.

Izraz »spletni strežnik« se pogosto nanaša samo na programsko opremo strežnika, ki zagotavlja funkcionalnost spletnega mesta. Spletni strežnik sprejema prošnje na uporabnikovem brskalniku in se odziva tako, da pošlje nazaj HTML-dokument (spletno stran) in ostale podatke, ki sodijo zraven. Prav tako spletni strežnik izvaja skripte, ki se nahajajo na samem strežniku.

4.3.2 Predstavitev tokov dogodkov

Za izvajanje avtomatskih testov je treba določiti scenarije oz. potek osnovnih tokov dogodkov. Poleg osnovnega toka je treba pomisliti tudi na tiste tokove dogodkov, ki se ne bodo najpogosteje uporabljali. Testiral bom tri najpogostejše scenarije pri uporabi spletne aplikacije ePero.

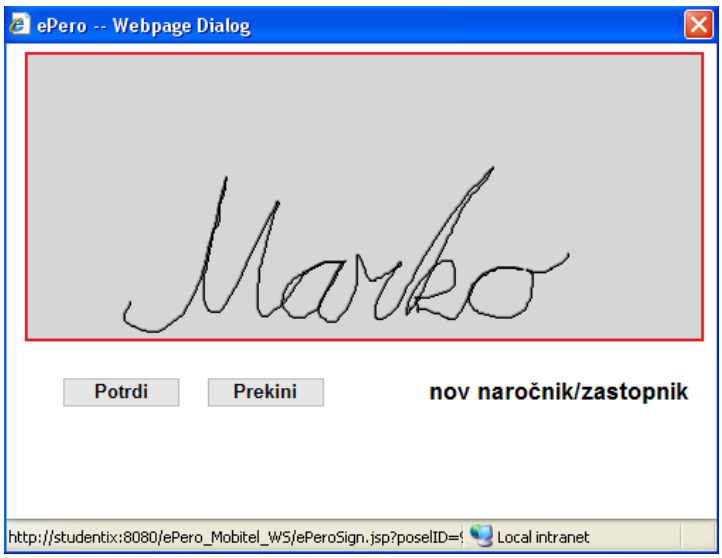
- Prvi scenarij [Slika 10] opisuje odpiranje spletne aplikacije ePero in klik na gumb »Zaključi postopek«. Pričakovano obnašanje je zagon spletne aplikacije, prikaz glavne strani s PDF-dokumentom. Ob kliku na gumb »Zaključi postopek« se bo spletna aplikacija zaprla.



Slika 10: Zaporedje dogodkov 1

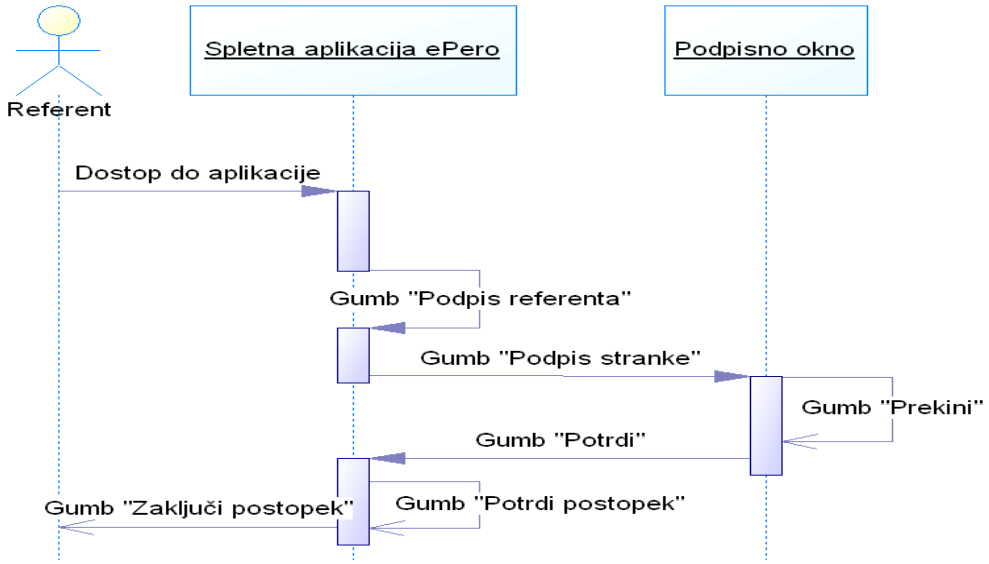
- Drugemu scenariju [Slika 12] pravimo osnovni tok dogodkov. Osnovni tok je najpomembnejši, saj bo vodil naročnika skozi elektronsko podpisovanje. Spletna aplikacija ePero se zažene, ko referent preko naročniške aplikacije dostopa do podpisnega strežnika. V tem primeru se na podpisnem strežniku zažene »Apache Tomcat Server«, na katerem je »ePero Web Service«. »ePero Web Service« prenese PDF-dokument, ki se shrani na podpisni strežnik v »Delovnem imeniku«. Poleg PDF-dokumenta »ePero Web Service« prenese ostale metapodatke[11] o dokumentu iz naročnikove aplikacije na podpisni strežnik. Pridobljeni metapodatki se zapišejo v XML-datoteko, ki se shrani na podpisni strežnik v Delovnem imeniku, poleg PDF-dokumenta. Požene se spletna aplikacija ePero. V spletni aplikaciji se nam prikaže dokument in na voljo imamo različne funkcije ob določenih korakih (npr. podpiši dokument, prikaži dokument, dodaj prilogo, natisni dokument, odstrani prilogo, dokument vnesi v dokumentni sistem itd.). Ob kliku na gumb »Podpiši dokument« se na podpisnem strežniku zažene »SIGNificant Server Web Signing Interface«. Naloga zadnjega je, da pripravi okolje za podpis. Elektronsko podpisno tablico aktivira s parametri o zajemu podpisa (velikosti podpisa, hitrost podpisa, naklon peresa, moč pritiska pisala na podlago itd.). »SIGNificant Server Web Signing Interface« poleg priprave elektronske podpisne tablice komunicira s »SIGNificant Server Tablet Maneger«. »Tablet Manager« je vtičnik za elektronski podpis. Zadnji odpre okno v spletni aplikaciji [Slika 11], da lahko vidimo podpis tudi na ekranu.

[11] Metapodatki vsebujejo informacije o nekem podatku, sami pa niso deltega. So podatki o podatkih.



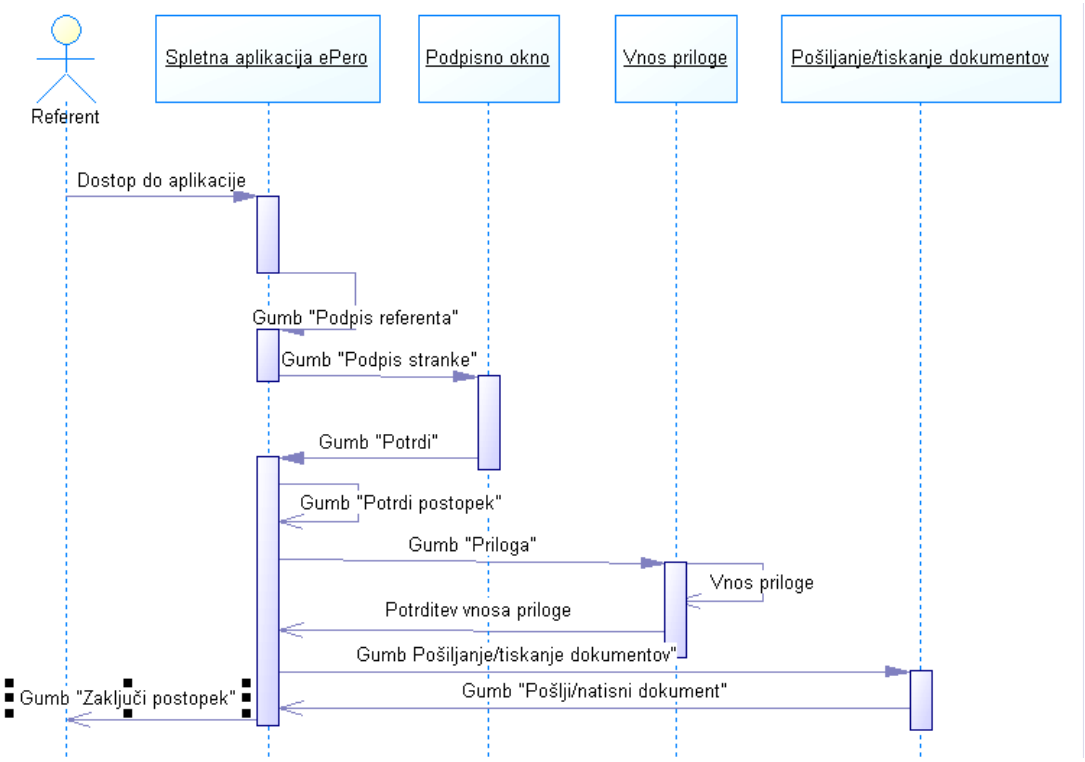
Slika 11:Prikaz podpisa

Referent se lahko podpiše na elektronsko tablico. Ob potrditvi podpisa se slika podpisa prenese nazaj na spletno aplikacijo ePero. Slika je v tem trenutku šifrirana. Spletna aplikacija ePero mora poslati podatke o sliki SIGNificant strežniku, ki je edini, ki lahko podatke o sliki dešifrira. Dešifrirano sliko se posreduje nazaj na ePero. ePero pridobljeno sliko podpisa prilepi na PDF-dokument in metapodatke zapiše v XML-dokument. V tem trenutku sta oba dokumenta elektronsko podpisna in se nahajata v »Delovnem imeniku« na podpisnem strežniku. Pred izhodom iz spletne aplikacije ePero nam tako ostaneta le še dva koraka. S klikom na gumb »Potrdi postopek« se dokument vnese v naročnikov dokumentni sistem. Pri tem koraku sta nam na voljo dve akciji. Dokumentu lahko dodamo prilogo, podpisane dokumente pa lahko pošljemo naročniku preko elektronske pošte. Za dokončanje postopka podpisovanja kliknemo na »Zaključi postopek«. Akcija podpisovanja je tako zaključena in spletni brskalnik se zapre.



Slika 12: Zaporedje dogodkov 2

- Scenarij za tretji tok dogodkov[Slika 13] je najobsežnejši. Začetni del poteka dogodkov bo podoben kot pri drugem, le da bomo po podpisanem dokumentu ta dokument dodali kot prilogo. Prilogo in podpisani dokument bomo poslali naročniku in nato zaključili celoten proces.



Slika 13: Zaporedje dogodkov 3

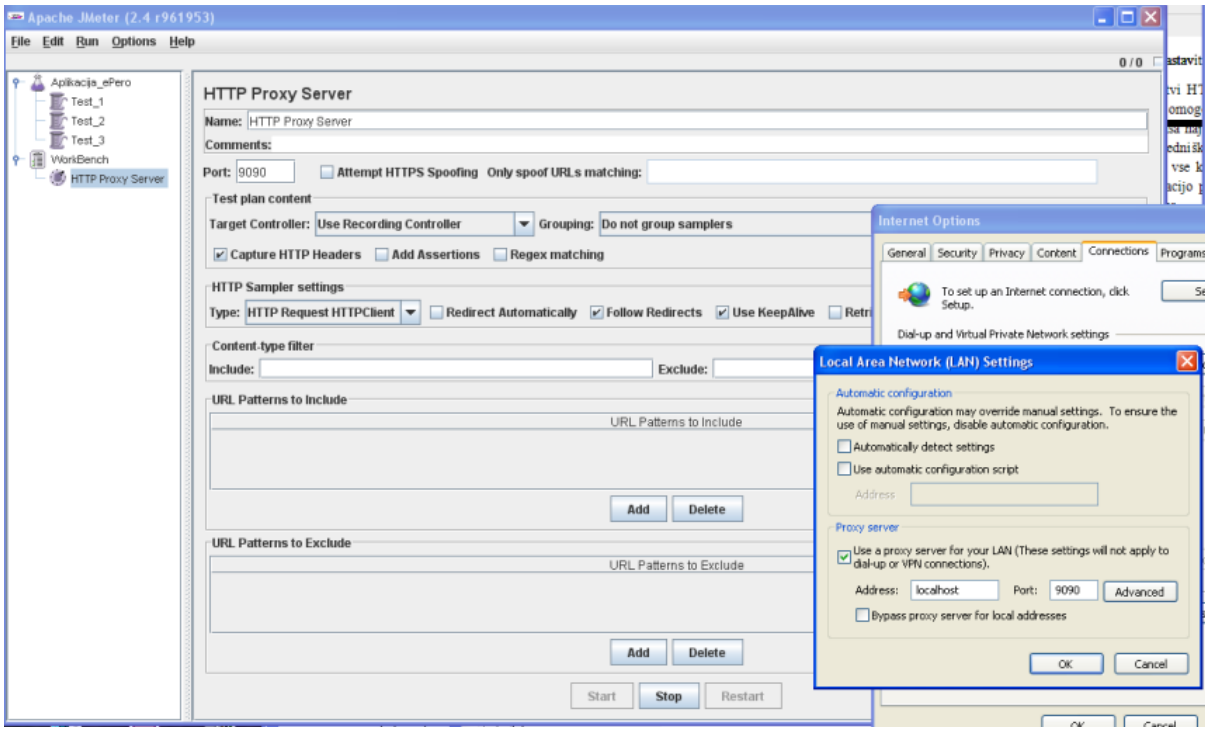
5. IZBIRA ORODIJ ZA IZVAJANJE AVTOMATSKIH ZMOGLJIVOSTNIH TESTOV IN IZVAJANJE MERITEV

Orodij za testiranje zmogljivostnih testov je na spletu ogromno. Veliko časa sem potreboval, da sem se odločil, katera orodja bom preizkusil na spletni aplikaciji ePero. V podjetju SETCCE smo se odločili, da testiranje preizkusim z več različnimi orodji, tako plačljivimi kot neplačljivimi. Z različnimi izmed njih sem testiral iste testne scenarije in rezultate primerjal med seboj. V tretjem poglavju sem preizkušena orodja na kratko predstavil. V nadaljevanju bom od prej predstavljenih orodij podrobneje opisal potek izvajanja avtomatskih zmogljivostnih testiranj. Testiral sem spletno aplikacijo ePero, pri čemer sem uporabil vse tri opisane testne scenarije. Opisani sta dve orodji, prvo je JMeter, ki je brezplačno, drugo pa WAPT Pro, ki je plačljivo, zato sem preizkusil njegovo preizkusno različico.

5.1 Priprava in izvajanje zmogljivostnih testov z orodjem JMeter

Orodje JMeter je brezplačno in ga lahko prenesemo iz uradne spletne strani »http://jakarta.apache.org/jmeter/«. Shranimo ga v »root« direktori »C:\«. Program poženemo tako, da v mapi »C:\jakarta-jmater\bin« poženemo skripto »jmeter.bat«. To je grafični način za poganjanje JMetra. JMeter nam poleg grafičnega načina ponuja možnost poganjanja v tekstovnemu načinu preko orodne vrstice.

Po predhodni definiciji testnih scenarijev v prejšnjem poglavju je treba te sestaviti. Za testiranje moramo tako vzpostaviti »HTML-zahtevo« (ang. HTML request). Vsi programi, ki sem jih sam testiral, so imeli možnost posneti scenarije, nobena izjema ni JMeter. Za potrebe testiranja je bilo treba pripraviti http Proxy strežnik (ang. »Proxy Server«). Proxy Server zapisuje zahteve uporabnika, ki jih nato naredi v brskalniku [1].

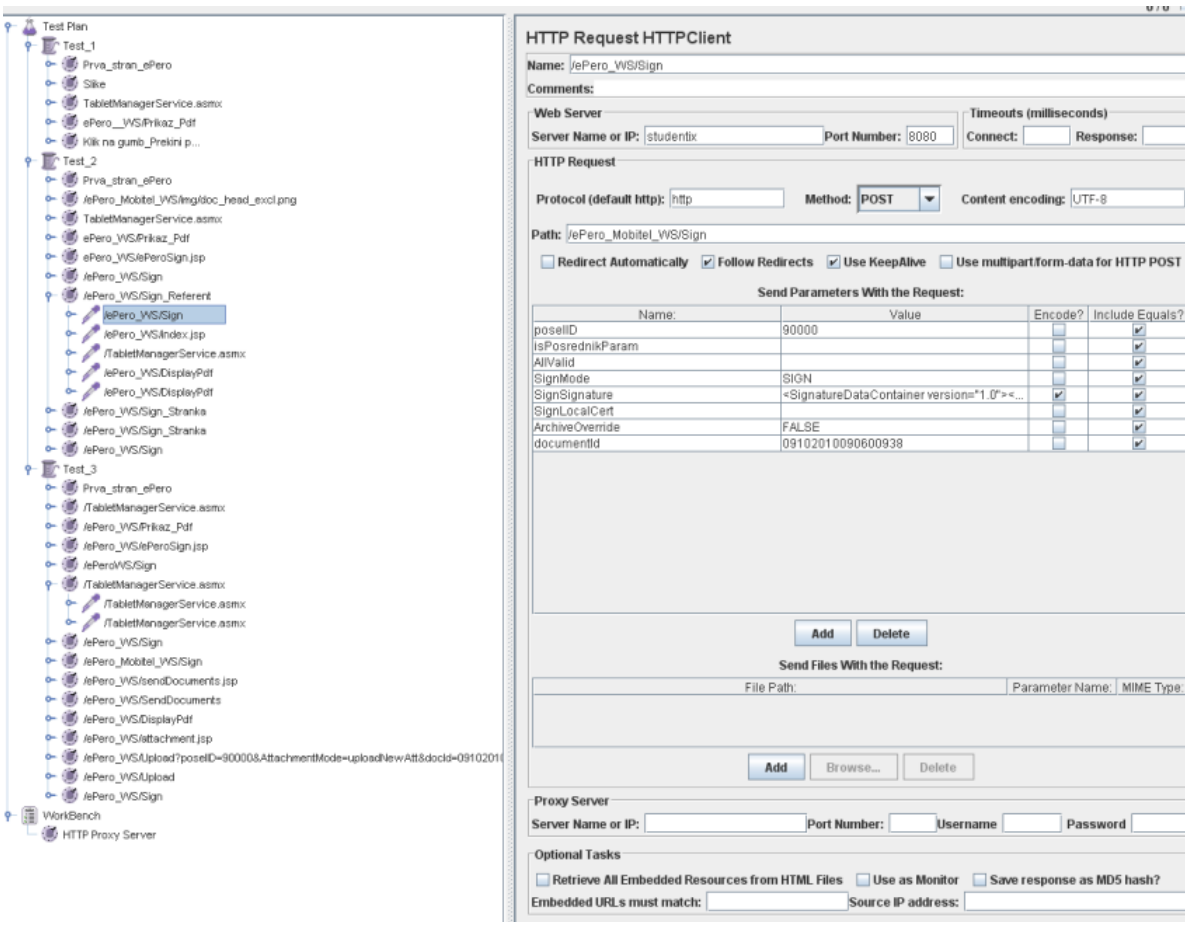


Slika 14: Prikaz orodja JMeter.

V Proxy strežniku sem nastavil vrata, pri katerih naj »posluša«. V Internet Explorerju je bilo treba nastaviti, da naj promet poteka skozi prej omenjena vrata [Slika 14]. Proxy strežnik ima še kopico drugih nastavitev, kot so:

- Ciljni krmilnik
- Združevanje http reguest
- Usklajevanje regularnih izrazov
- Izbiro tipa zahteve
- Avtomatsko preusmeritev odgovora itd.

S klikom na gumb »Start« se je Proxy strežnik zagnal. V Internetu Explorerju sem pognal spletno aplikacijo ePero in izvedel vsak testni scenarij posebej.



Slika 15: Prikaz HTTP-zahtevka

Ko sem se sprehodil skozi vse tri testne scenarije, sem pripravil okolje za testiranje [Slika 15].

Ker ePero aplikacija temelji na tem, da se ob zagonu spletne aplikacije dokumenti stranke nalagajo na strežnik, kjer aplikacija teče, sem moral narediti še dokumente, ki se jih bo podpisovalo. Ročno sem pripravil štirideset dokumentov. Pred prvim testiranjem sem moral popraviti skripto. Pri snemanju so se določene spremenljivke shranile fiksno.

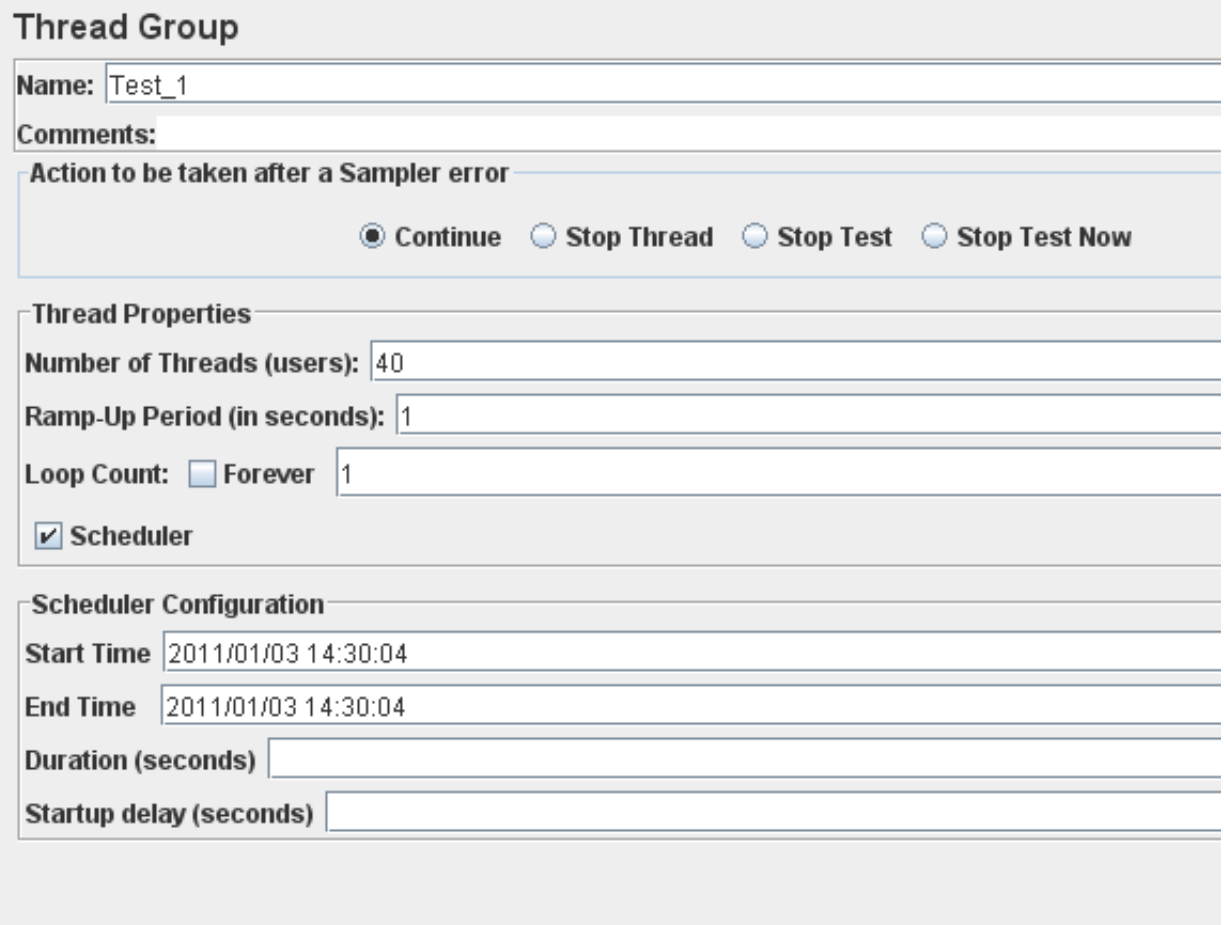
```
npr. poselID = 90000
```

To pomeni, da se ob tem, ko se požene aplikacija ePero, naloži dokument, ki vsebuje »poselID = 90000«. V tem poslu je PDF-dokument in XML-datoteka, ki vsebujeta metapodatke o dokumentu (za testiranje je vsebina xml datoteke nepomembna). S preprostim ukazom sem, kjer je bilo treba, nastavil, da se med izvajanjem zmogljivostnega testa požene spletna aplikacija z različnim dokumentom.

```
${__javaScript( ${nb.poselID}+ ${__counter(FALSE)} )}
```

Če pride do napake in se požene dokument, ki je že bil podpisan, bi se aplikacija pognala brez težav. Težava bi le nastala pri podpisovanju.

V meniju poslušalec [12] so komponente, ki nam pomagajo pri vizualni predstavitvi zmogljivostnih testov. V svojih testnih primerih sem uporabil »View Results Tree«. Zadnja stvar pri nastavljanju aplikacije JMeter je »Test Plan«. V »Test Planu« nastavimo število navideznih uporabnikov [Slika 16]. Glede na to, da imam pripravljenih štirideset dokumentov, bom testiral aplikacijo za štirideset navideznih uporabnikov. (V nadaljevanju sem za potrebe testiranja moral pripraviti še več dokumentov za podpisovanje.) V polju »Schedule« imamo možnost poganjanja avtomatskih testov. To storimo tako, da vpišemo, kdaj bi radi, da se izvajajo avtomatski testi. JMeter nam ponudi možnost, da se test zmogljivosti izvaja ob določeni uri. Tu velja omeniti, da nam plačljivi programi dopuščajo večjo izbiro.



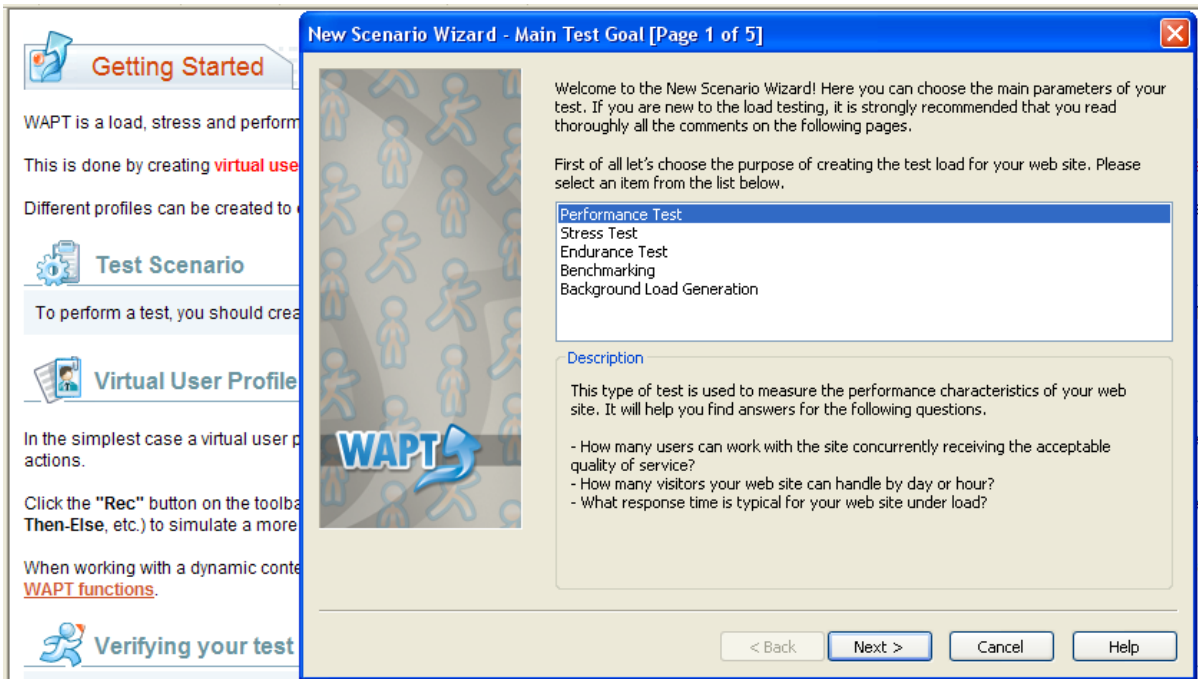
Slika 16: Nastavitve za avtomatsko izvajanje testov

V tem trenutku imam nastavljeno orodje JMeter in lahko poženem svoj prvi test. To storim tako, da kliknem na zavitek zagon [13] in nato na gumb »Start«. Med izvajanjem zmogljivostnega testa se bo v zgornjem desnem kotu gumb pobarval zeleno, levo od njega se bo pokazalo število uporabnikov, ki jih želimo generirati, in število uporabnikov, generiranih do nekega trenutka med izvajanjem zmogljivostnega testa.

[12] Ang. Listener.
[13] Ang. Run.

5.2 Priprava in izvajanje zmogljivostnih testov z orodjem WAPT Pro

Orodje WAPT Pro za testiranje spletnih aplikacij je plačljivo orodje in stane okoli 350 $. Jaz sem za potrebe testiranja prenesel tridesetdnevno brezplačno različico, ki je dosegljiva na spletni strani »http://loadtestingtool.com/«. Takoj ob zagonu prepoznamo razliko med tem orodjem in prej opisanim JMetrom. Ob prvem zagonu nas aplikacija vpraša po nekih splošnih nastavitvah glede snemanja testnih scenarijev, izvajanja testov in še marsičem. Vse te nastavitve je možno kasneje tudi popraviti oz. jih ponastaviti. Aplikacija nam takoj na začetku ponudi najrazličnejše oblike pomoči. S klikom na ikono »New« se nam zažene čarovnik za opravljanje s scenariji. Na izbiro nam ponudi pet različnih testov [Slika 17]. Zmogljivostni test, obremenitveni test, vzdržljivostni test, »Benchmarking«, »Background Load Generation«. V svojem primeru sem izbral zmogljivostni test in kliknil gumb naprej[14] [13].

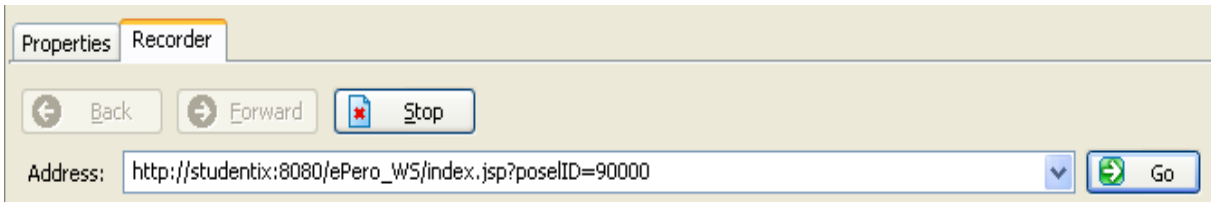


Slika 17: Čarovnik za pripravo scenarija za testiranje

V nadaljevanju povpraša še po nekaj različnih stvareh glede zmogljivostnega testa. Po končanem izboru se nam generira projekt, ki je že pripravljen na snemanje opisanih scenarijev. WAPT-aplikacija/orodje je že tako dovršena, da ni treba poganjati naše spletne aplikacije v internetnem brskalniku. Sama aplikacija vsebuje svoj brskalnik, kar nam »olajša« delo, saj nam ni treba nastavljati drugih brskalnikov.

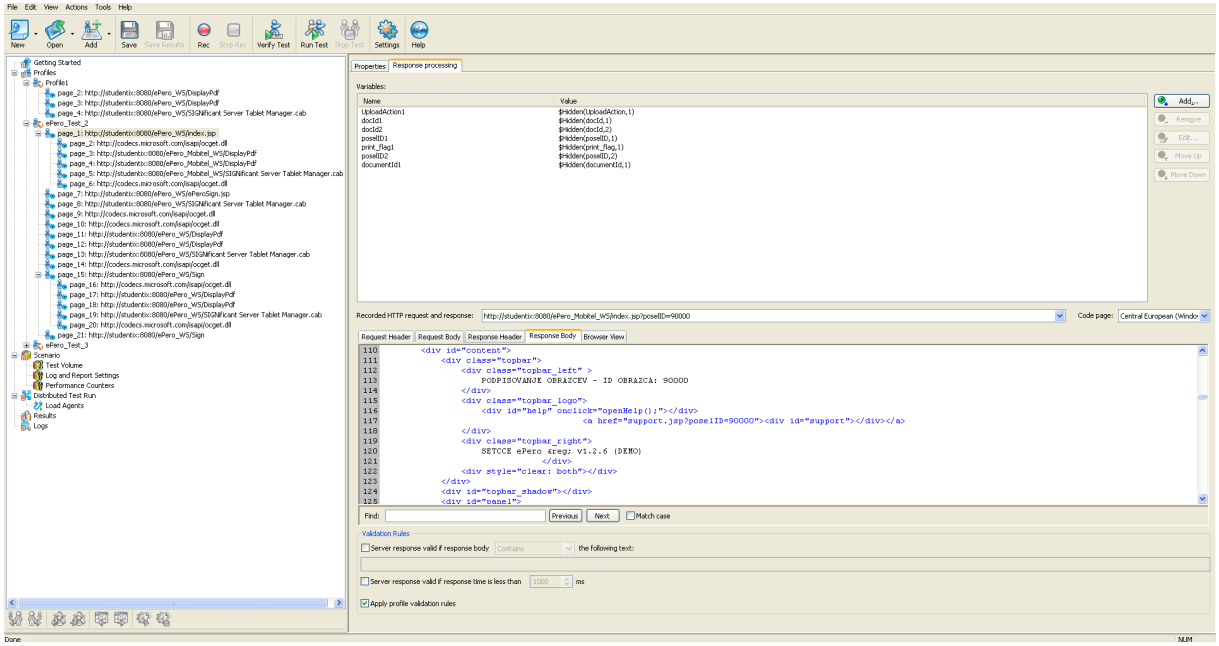
Uporabniški profil je sestavljen iz zaporedja strani, ki se kličejo med snemanjem testnega scenarija.

[14] Ang. Next.



Slika 18: Snemanje URL-naslova

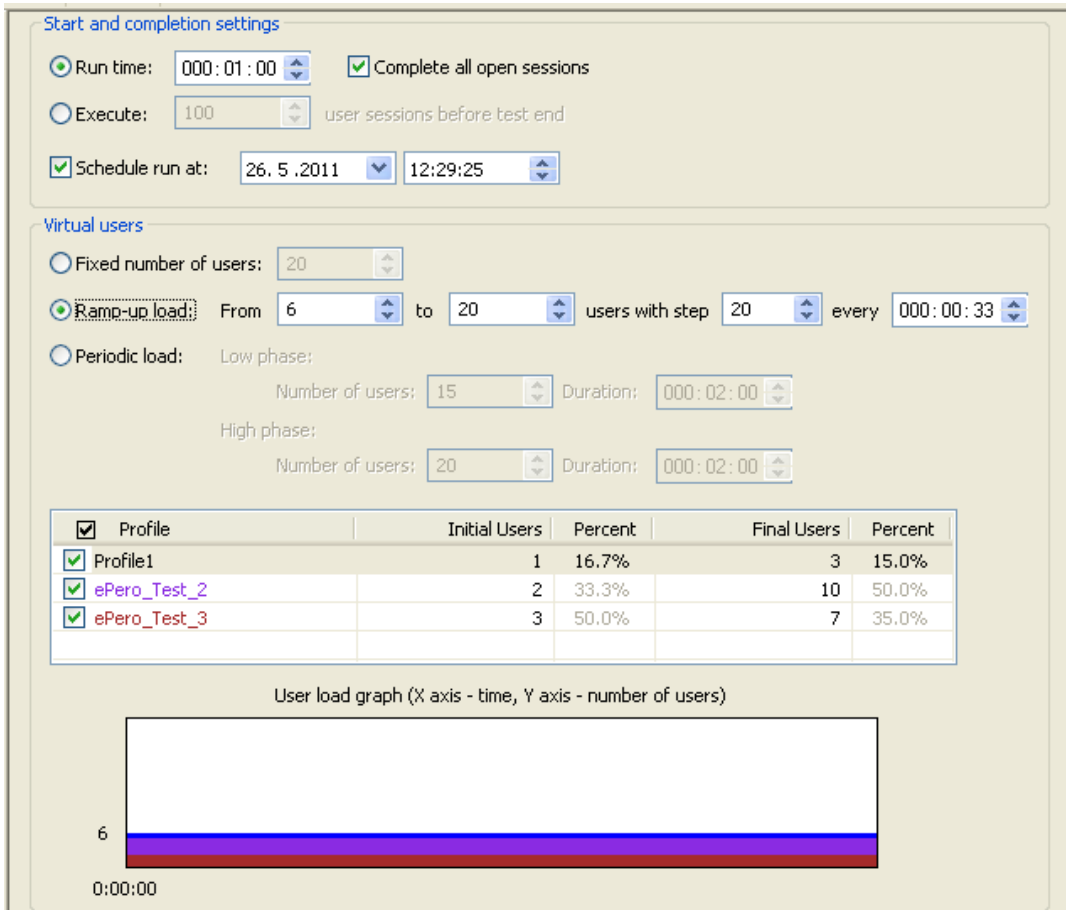
Z uporabo WAPT Pro snemalca lahko posnamemo korake oz. potek testnega scenarija in zapise kasneje uporabimo pri zmogljivostnem testiranju. Snemalec uporablja okno brskalnika Internet Exporer. V okno poleg besede »Adress« vnesemo URL-naslov naše spletne aplikacije in orodje WAMP že začne zajemati zahteve na strežnik [Slika 18].



Slika 19: Prikaz delovanja WAMP Pro orodja

Ko se sprehodimo po aplikaciji, po prej začrtanih testnih scenarijih, vidimo, da program sam poskrbi za pravilno shranjevanje in sortiranje zahtevkov, ki jih opravimo na strežniku [Slika 19]. Po želji lahko snemalnik predčasno izklopimo. Po končanem snemanju kliknemo na »koren« vsakega testa, saj je treba nastaviti profil za izvajanje testov. Sama aplikacija omogoča ogromno nastavitev, predstavil bom tiste, ki sem jih uporabil.

- Zamude uporabnika (ang. User think time): pri tem, ko ustvarimo testni scenarij, pride do zamud pri samem klikanju na gumbe. To je človeško. Program nam omogoča, da testiramo brez zamud, upoštevanje istih časovnih zamud, kot smo jih pri snemanju napravili, ali pa mu damo časovni interval, v katerem bo naključno izbiral zamude.
- Nalaganje različnih elementov (ang Load page elements): pri snemanju naših spletnih aplikacij oz. spletnih strani nam WAPT Pro snemalnik posname tudi druge elemente, kot so slike, JavaScript, CSS itd. Med testiranjem imamo možnost,izključitve opcije nalaganja slik.



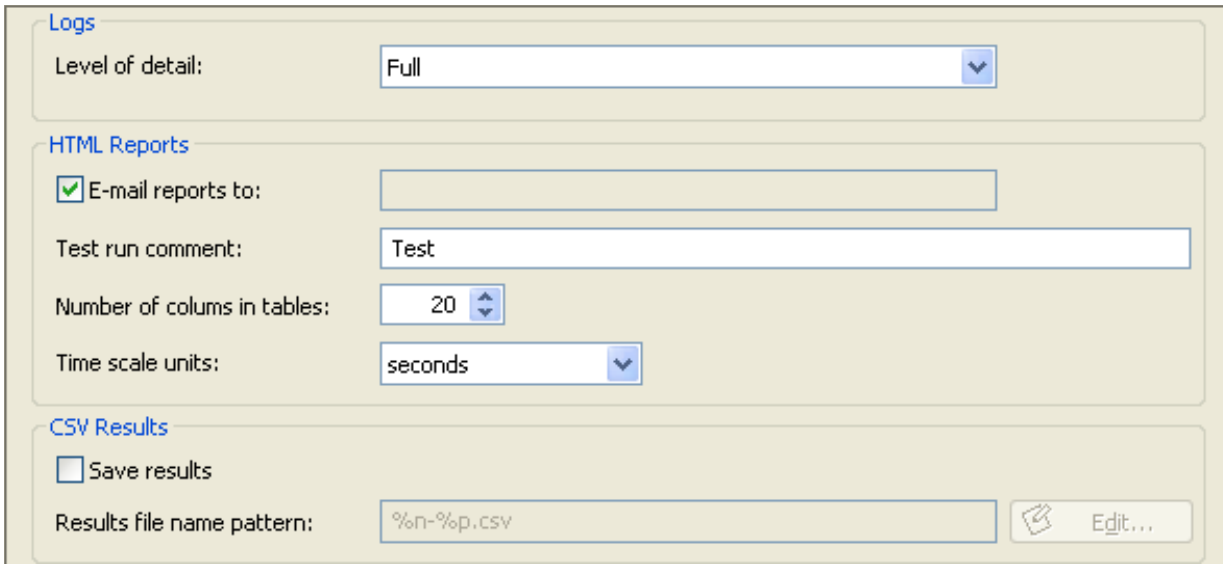
Slika 20: Nastavitve za določanje števila virtualnih uporabnikov

- Začetne spremenljivke (ang. Initial variables). za izračunavanje parametrov URL-poti lahko uporabljamo spremenljivke. V mojem primeru je to »poselID«, saj vsakič, ko kličem preko URL-naslova, kličem drugi posel.

Če želimo opraviti test, moramo ustvariti testne scenarije. S programom »WAMP Pro« to ni težko. Lastnosti scenarijev lahko poljubno spreminjamo v »Test Volume« ali »Log and report settings«.

- Test meni (ang. Test Volume): v testnem meniju lahko izberemo, katera merila se uporabljajo pri izvajanju testov, in število navideznih uporabnikov [Slika 20]. Možne nastavitve so:
    - Čas delovanja (ang. Run time) - določimo čas delovanja testa.
    - Določanje števila uporabnikov (ang. Fixed number of users) - določamo fiksno število uporabnikov, ki jih test generira. Ker je program plačljiv, sem testiral na preizkusni različici; lahko sem generiral največ 20 uporabnikov.
    - Postopno povečevanje uporabnikov (ang. Ramp-up load) -aplikacija nam ponuja možnost, da se uporabniki postopoma generirajo in testirajo. Med testiranjem lahko uporabimo zakasnitve.

- Hranjenje in generiranje poročil (ang. Log and Report Setting) - omogoča nam kreiranje in shranjevanje poročil med testiranjem in po testiranju.
    - Dnevnik napak (ang. Log) - nastavitev dnevnika napak [Slika 21] nam olajša delo pri preverjanju, kje je med testiranjem prišlo do napake.
        - Popolni pregled (ang. Full Log) - celoviti pregled izvajanja testa, ki vsebuje podatke o vseh zahtevah, odgovorih, preusmeritvah se hrani v HTML-kodi.
        - Pregled napak (ang. Error Log) – v dnevnik se vpisuje le v primeru napake.
    - HTML Poročila (ang. HTML Report) – ponuja nam možnost pošiljanja e-mail sporočila vsem osebam, ki so zainteresirane za potek izvajanja zmogljivostnega in drugih testiranj.



Slika 21: Nastavitve za hranjenje napak

Za zagon testnega scenarija nas čaka le še zadnji korak. Nastaviti je treba agenta za nalaganje (ang. Load Agents). Ta nastavitev nam omogoča, da lahko spletna aplikacija deluje na več različnih spletnih strežnikih. Spletno aplikacijo ePero sem izvajal le na svojem računalniku, zato sem to moral tudi nastaviti. Aplikacija uporablja prevzeta vrata 9474 za iskanje lokalnega zastopnika. Kasneje, ko sem dostopal na drugi strežnik, sem moral temu prilagoditi zahteve.

Zagon Testnega scenarija

Pripravljeni smo za izvajanje testnega scenarija. Do tega koraka je bilo treba narediti precej več stvari, kot pa pri predhodno testiranem orodju JMeter. Ponovitev prejšnjih korakov. Najprej sem si moral zamisliti testni scenarij, nato ga z WAMP Pro snemalnikom posneti v izdelani profil. Opredeliti sem moral, kaj naj se testira in kako naj se test izvede ter kam naj se zapišejo podatki o izvedenem testu. Kot zadnjo stvar sem nastavil še agenta za nalaganje spletne aplikacije. Ko so bila vsa našteta opravila izvedena, sem lahko pričel z izvajanjem testa.

S klikom na zagon testa (ang. Run Test) [Slika 22]

Slika 22: Ikona za zagon zmogljivostnega testa

se test začne izvajati samodejno. Med izvajanjem zmogljivostnega testa je v agentu za nalaganje vidna poraba procesorja, pomnilnika, uporaba internetnega omrežja itd. Poleg agenta imamo v spodnjem desnem kotu vrstico, ki nam prikazuje stanje poteka testa [Slika 23].

Slika 23: Prikaz stanja izvajanja testa

Vidimo lahko, koliko časa je minilo od začetka izvajanja testiranja in število navidezno generiranih uporabnikov, ter končno število uporabnikov. V našem primeru le 20.

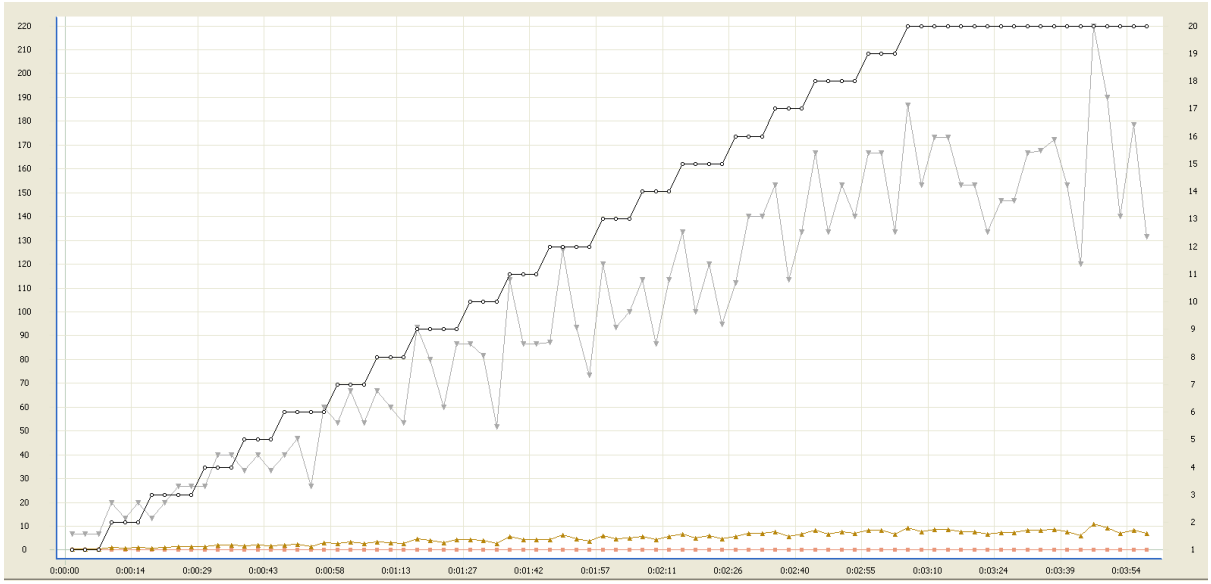
6. ANALIZA IZVAJANIH TESTOV

V prejšnjem poglavju sem predstavil, kako pripraviti orodji JMeter in WAMP Pro za izvajanje zmogljivostnih testov. V tem poglavju želim predstaviti rezultate testiranja, ki sem jih pridobil z izvajanjem vseh treh testnih scenarijev na obeh orodjih.

6.1 Rezultati testov izvajanja z orodjem WAMP

Najprej sem opravil test na programu WAMP Pro, kot sem že omenil, je program plačljiv, zato sem uporabil testno verzijo, ki mi omogoča generiranje le dvajsetih uporabnikov. Zaradi nizkega števila generiranih uporabnikov nisem mogel izvajati stresnih testov. Vsekakor pa nam bo program lepo generiral podatke in poročilo o izvedenih testih. Izvedel sem testiranje drugega testnega scenarija (osnovnega toka), s katerim bom testiral glavne lastnosti spletne aplikacije ePero.

Rezultati drugega scenarija



Slika 24: Graf testiranja spletne aplikacije

Graf nam prikazuje [Slika 24] potek testiranja drugega testnega scenarija na spletni aplikaciji ePero. Z grafa je razvidno, kako postopoma naraščajo navidezni uporabniki (od prvega do dvajsetega). V nadaljevanju je tabela, kjer so predstavljeni povprečni odzivni časi, ki so eden od pomembnejših značilnosti pri izvajanju obremenitvenih testov. Pomembni so zato, da vidimo, ali se naša aplikacija dovolj hitro odziva na zahteve.

Time	0:00:00-0:00:24	0:00:24-0:00:48	0:00:48-0:01:12	0:01:12-0:01:36	0:01:36-0:02:00	0:02:00-0:02:24	0:02:24-0:02:48	0:02:48-0:03:12	0:03:12-0:03:36	0:03:36-0:04:00	Total
	16	40	62	90	108	129	151	187	185	197	1165
Min	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01 (0.01)
Max	0.05(0.08)	0.03(0.05)	0.03(0.05)	0.05(0.05)	0.41(0.42)	0.13(0.16)	0.08(0.09)	0.05(0.11)	0.09(0.11)	0.05(0.08)	0.41 (0.42)
Avg	0.02(0.03)	0.02(0.03)	0.02(0.03)	0.02(0.03)	0.03(0.03)	0.03(0.03)	0.02(0.03)	0.02(0.03)	0.02(0.03)	0.02(0.03)	0.02 (0.03)
Avg90	0.04(0.05)	0.03(0.04)	0.03(0.04)	0.03(0.04)	0.08(0.09)	0.05(0.06)	0.04(0.05)	0.03(0.05)	0.04(0.05)	0.03(0.04)	0.04 (0.06)

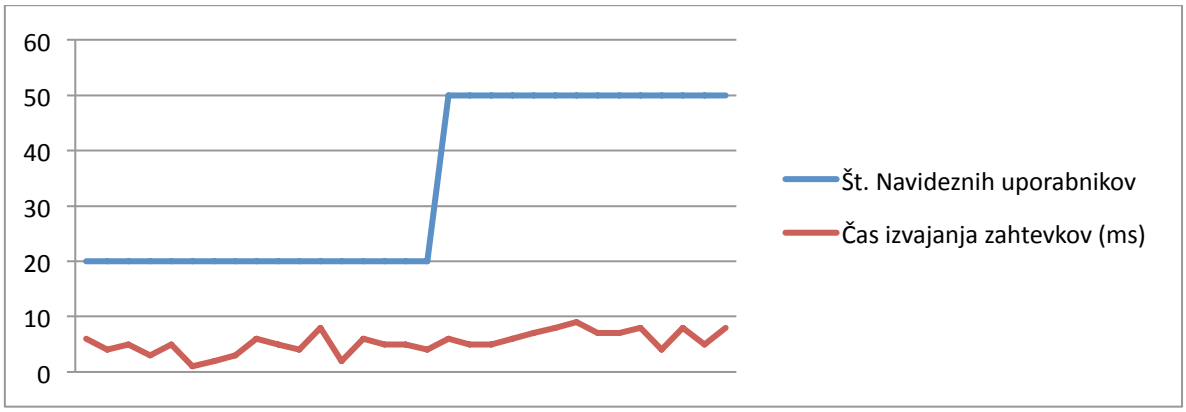
Slika 25: Izmerjani časi pri testiranju

V tabeli vidimo povprečne odzivne čase [Slika 25] pri izvajanju drugega testnega scenarija. Opazimo lahko, da spletna aplikacija ne čuti nikakršne obremenitve in so odzivni časi zelo dobri oz. nizki. V nadaljevanju sem izvedel še druge testne scenarije z istim orodjem, a se je izkazalo, da se zaradi nizkega števila uporabnikov ne poznajo nikakršne razlike med izvajanjem zmogljivostnih testov. Med izvajanjem testnih scenarijev sem s programom »Process Explorer« nadziral obremenitev strežnika. Zasedenost procesorja pri izvajanju dvajsetih navideznih uporabnikov se giblje med 25 % in 40 %. Zasedenost pomnilnika in diska se je povečala na 35 %, kar pomeni, da bo aplikacija delovala nemoteno pri dvajsetih uporabnikih. Zaradi omejitev na aplikaciji WAMP Pro sem prenehal z izvajanjem testov na tem orodju in pričel z izvajanjem testov z orodjem JMeter.

6.2 Rezultati testiranja z orodjem JMeter

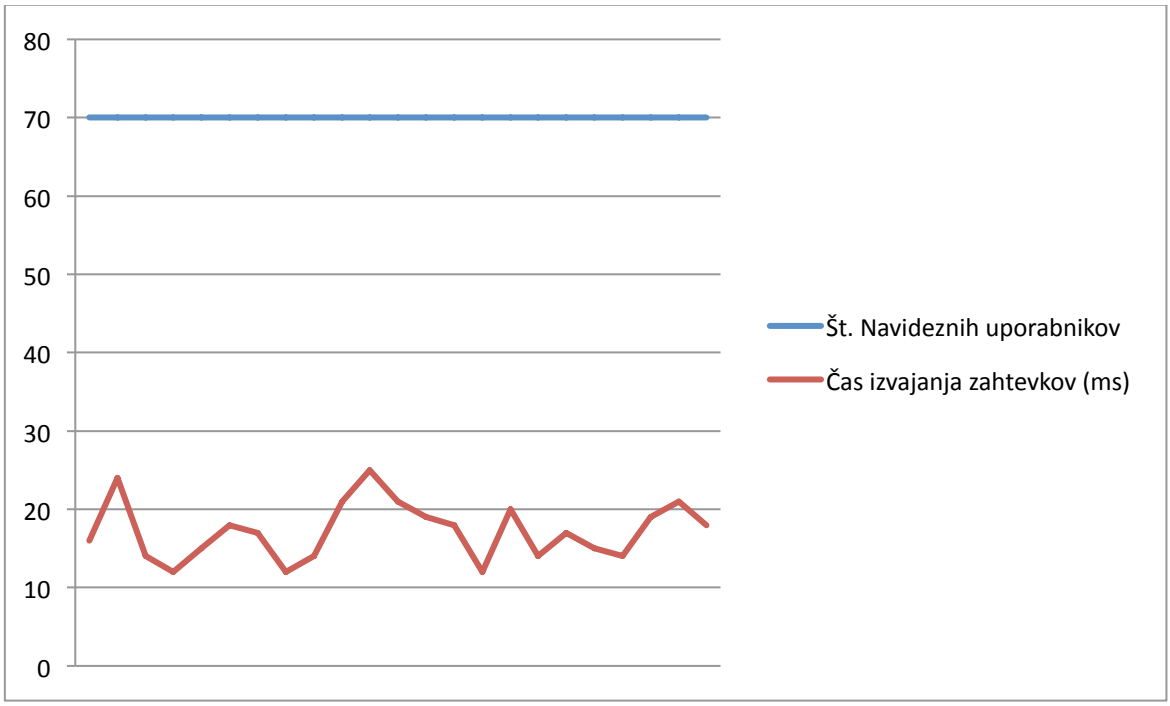
Pri testiranju z orodjem JMeter sem imel bolj proste roke. Lahko sem generiral različno število uporabnikov in nisem bil zelo omejen. Omejen sem le bil na zmogljivost spletne aplikacije oz. na strežnik, na katerem je aplikacija. Kot sem že prej predstavil, bom z orodjem JMtere testiral spletno aplikacijo ePero na vseh treh scenarijih in z različnim številom generiranih navideznih uporabnikov. Najprej bom pognal testni scenarij z dvajsetimi uporabniki, potem petdesetimi, sedemdesetimi in stotimi navideznimi uporabniki. Za generiranje navideznih uporabnikov sem uporabil komponento »Thread Group«. Vsak navidezni uporabnik pa je potreboval PDF-datoteko, ki se je podpisala.

Testiranje prvega scenarija [Slika 26] z dvajsetimi in petdesetimi navideznimi uporabniki je potekalo brez večjih težav. Odzivni časi http zahtev, ki so bili izvedeni, so zelo dobri.



Slika 26: Prvi testni scenarij z orodjem JMeter

Pri povečevanju navideznih uporabnikov iz petdeset na sedemdeset pride do višjih časovnih intervalov, a tudi ti so zelo kratki in uporabnik ne občuti nikakršnih zakasnitev.



Slika 27: Povečevanje števila navideznih uporabnikov

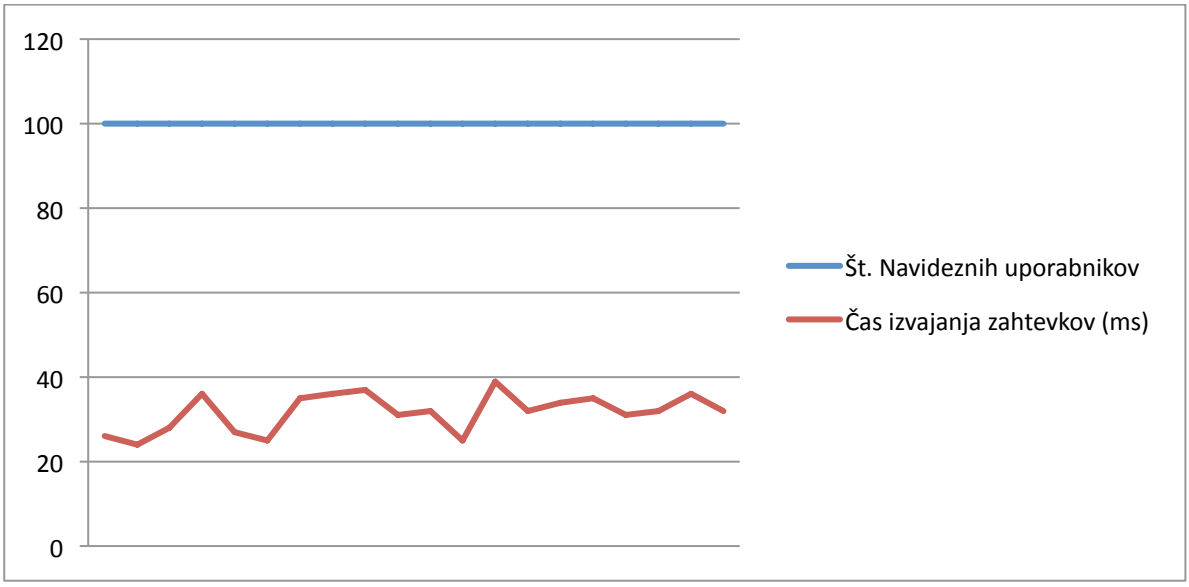
Prav tako sem med izvajanjem prvega testnega scenarija spremljal obremenitev strežnika. Opazil sem, da je poraba CPU na strežniku precej nizka, kar je le nakazovalo, da je aplikacija zelo hitra.

Izvajanje drugega testnega scenarija

Pri testiranju drugega testnega scenarija sem imel pri testiranju dvajsetih navideznih uporabnikov možnost dobljene rezultate primerjati s tistimi, ki sem jih pridobil pri testiranju z orodjem WAMP Pro. Kot je bilo pričakovano, so bili rezultati med seboj zelo podobni,

zmogljivost podpisnega strežnika pa mi je dopuščala testirati z orodjem JMeter spletno aplikacijo ePero za večje število navideznih uporabnikov.

Nadaljeval sem s testiranjem petdesetih navideznih uporabnikov. Zmogljivostni test se je izvedel brez težav. Opazil sem, da so se po štiridesetem navideznem uporabniku pričeli časi povečevati, vendar le za nekaj 10 ms. Sistem je občutil minimalno povečanje obremenitve samo pri nalaganju dokumentov na strežnik. Pri prenosu podpisanih dokumentov s spletne aplikacije v dokumentni sistem pa so se časi prenosa ponovno spustili na predhodno raven. V spodnjem grafu so predstavljeni časi, ki so bili potrebni, da so se dokumenti prenesli na spletni strežnik, kjer so se kreirali posli, ki so bili nato podpisani.



Slika 28: Drugi testni scenarij - prenos dokumentov

Opazil sem, da se je pri testiranju petdesetih navideznih uporabnikov povečala obremenitev strežnika. Pri začetnem nalaganju dokumentov [Slika 28] se je na podpisnem strežniku povečala poraba procesorja. Poraba je tako znašala okoli 60 %, medtem ko se je poraba diska in zasedenost pomnilnika povečala na 45 %. Pri vnosu na podpisni strežni, je poraba procesorja še vedno znašala 45 %, medtem ko se je poraba diska in pomnilnika povečala za kar 20 %. Po videnih podatkih sem sklepal, da bom z naslednjim testiranjem, s sedemdesetimi navideznimi uporabniki, močno obremenil sistem. Izkazalo se je prav nasprotno. Spletna aplikacija se je izvedla zelo hitro. Primerjalni časi so bili praktično identični. Poraba virov na podpisnem strežniku pa ni kazala nobenih večjih odstopanj. Aplikacijo sem nato pognal na navideznih stotih uporabnikih. Tudi s stotimi navideznimi uporabniki nisem odkril nobenih nepravilnosti pri testiranju spletne aplikacije ePero. Spletna aplikacije se je izvedla prav taki hitro kot v prejšnjem testiranju. Spremembe so nastale le pri porabi virov podpisnega strežnika. Sama poraba je tako narasla za kakšnih 10 %, a to ni vplivalo na odzivnost aplikacije.

Izvajanje tretjega testnega scenarija

Že pri testiranju druga testnega scenariju s stotimi navideznimi uporabniki sistem ni občutil nikakršnih obremenitev, zato sem se odločil, da bom kar takoj prešel na testiranje sedemdesetih in stotih uporabnikov. Tretji testni scenarij se razlikuje v tem, da po vnosu dokumentov v dokumentni sistem naložimo še en dodaten dokument, ki predstavlja prilogo. Nato dokumente skupaj s prilogo pošljemo preko elektronske pošte. Pri testiranju pošiljanja sem to naredil le za petdeset uporabnikov, da ne bi prejel prevelike količine elektronske pošte na svoj elektronski naslov. Pri testiranju za petdeset uporabnikov nisem zaznal nobene težave. Prejel sem vseh petdeset elektronskih sporočil. Po vspešno opravljenem testu pošiljanja elektronskih sporočil na en elektronski naslov, sem testiral pošiljanje elektronskih sporočil še na več različnih poštnih naslovov. Izkazalo se je, da spletne aplikacije s tem sploh nisem obremenil. Obremenil sem le poštni predal.

Nadaljeval sem s testiranjem s sedemdesetimi uporabniki. Izkazalo se je, da se je pri sočasnem izvajanju sedemdesetih uporabnikov obremenjenost strežnika povečala zaradi nalaganja prilog. Poudariti moram le to, da so bile priloge različno velike in da so določene vsebovale končnico PDF, druge pa TIF. Obremenitev se je povečala do te mere, da sem jo lahko primerjal s prejšnjim izvajanjem testnega scenarija za sto navideznih uporabnikov. Kljub povečanju časa izvajanja in porabe virov na podpisnem strežniku, je aplikacija delovala brezhibno. Zadnji test, ki sem ga opravil, se je izvedel na stotih navideznih uporabnikih. Sto istočasnih navideznih uporabnikov je preneslo svoj dokument na strežnik, ga podpisalo, vneslo v dokumentni sistem in dokumentu dodalo prilogo brez težav. Odzivni časi so primerljivi s tistimi, ki sem jih pridobil za isti postopek s sedemdesetimi uporabniki. Pri porabi virov na podpisnem strežniku so se poznala določena odstopanja, a so ta še vedno nizka in aplikacija je delovala brez težav.

7. ZAKLJUČEK

Med izvajanjem zmogljivostnih testov sem spoznal, zakaj je to področje eno bolj »osovraženih« med programerji. Testiranje zmogljivosti spletnih aplikacij je zelo zapleten in dolgotrajen proces, če želimo izvajati kakovostne teste. Vedeti je treba, kaj se testira in kakšen pristop bomo uporabili. Zaradi velikega števila orodij za izvajanje testov pa je težka tudi odločitev, katero orodje izbrati.

Pri spoznavanju orodij sem ugotovil, da je pri vseh princip delovanja zelo podoben. Določena orodja dopuščajo več izbire pri sestavljanju testnih scenarijev, druga manj. Orodja za izvajanje avtomatskih zmogljivostnih testov se v večji meri razlikujejo v načinu predstavitve podatkov o izvedenih testih. Sam sem med testiranjem z orodjem JMeter doživel eno neprijetno izkušnjo. Opravil sem opisane teste in zaradi napačnih nastavitev dobil napačne rezultate, ki sem jih imel sprva za pravilne. S predhodnimi rezultati sem celo prišel do takega zaključka, da spletna aplikacija potrebuje večjo moč podpisnega strežnika za izvajanje spletne aplikacije ePero. S pomočjo razvojne ekipe sem ponovno izvedel zmogljivostne teste na spletni aplikaciji ePero in pri tem odkril napake, ki sem predhodno naredil, jih popravil.

Prišel sem do sklepa, da si zmogljivostnega testiranja ne smemo predstavljati kot nekaj zahtevnega, ampak kot izziv. Z upoštevanjem in izvajanjem celotnega življenjskega modela priprave in izvajanja testiranja obstaja velika verjetnost, da nam bo testiranje tudi uspelo. Sam sem ugotovil, da za testiranje potrebuješ ogromno časa, če bi ga imel na razpolago, bi lahko ustvaril več testnih scenarijev in te poganjal na več različnih orodjih. Tako bi prišel do še natančnejših rezultatov in spoznal, česa je dejansko zmožna spletna aplikacija.

8. KAZALO SLIK

Slika 1: Življenjski model razvoja spletne/programske opreme [6]. 3
Slika 2: Osnovna pravila pri zmogljivostnem testiranju [6]. 8
Slika 3: Logotip testnega orodja LoadTracer 23
Slika 4:Logotip programskega orodja JMeter [1] 24
Slika 5: postopek proXSign 27
Slika 6: Elektronsko podpisovanje 28
Slika 7: Prikaz procesa elektronskega podpisovanja 28
Slika 8: Prikaz spletne aplikacije ePero 30
Slika 9: Podpisni strežnik 32
Slika 10: Zaporedje dogodkov 1 33
Slika 11:Prikaz podpisa 34
Slika 12: Zaporedje dogodkov 2 35
Slika 13: Zaporedje dogodkov 3 35
Slika 14: Prikaz orodja JMeter. 37
Slika 15: Prikaz HTTP-zahtevka 38
Slika 16: Nastavitve za avtomatsko izvajanje testov 39
Slika 17: Čarovnik za pripravo scenarija za testiranje 40
Slika 18: Snemanje URL-naslova 41
Slika 19: Prikaz delovanja WAMP Pro orodja 41
Slika 20: Nastavitve za določanje števila virtualnih uporabnikov 42
Slika 21: Nastavitve za hranjenje napak 43
Slika 22: Ikona za zagon zmogljivostnega testa 44
Slika 23: Prikaz stanja izvajanja testa 44
Slika 24: Graf testiranja spletne aplikacije 45
Slika 25: Izmerjani časi pri testiranju 46
Slika 26: Prvi testni scenarij z orodjem JMeter 47
Slika 27: Povečevanje števila navideznih uporabnikov 47
Slika 28: Drugi testni scenarij - prenos dokumentov 48

9. KAZALO TABEL

Tabela 1: Prepoznavanje testnega okolja 9
Tabela 2: Prepoznavanje zmogljivostnih meril 10
Tabela 3: Načrtovanje zmogljivostnih testov 11
Tabela 4: Priprava testnega okolja 12
Tabela 5: Razvijanje oblike za zmogljivostne teste 13
Tabela 6: Izvajanje zmogljivostnih testov 14
Tabela 7: Analiziranje rezultatov, poročil in ponovno testiranje 16

10. VIRI

[1] Apache – Jmeter (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://jakarta.apache.org/jmeter/.

[2] Facilita - Forecast (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.facilita.co.uk/products/web.asp.

[3] Liu H. H. (2009). Software Performance and Scalability: A Quantitative Approach. Wiley.

[4] Lydia Ash The Web Testing Companion: The Insider's Guide to Efficient and Effective Tests, Wiley, May 2, 2003.

[5] ManegeEngine – QEngine (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.manageengine.com/.

[6] Meier, J., Farre, C., Bansode, P., Barber, S., & Rea, D. (2007). Performance Testing Guidance for Web Applications. Microsoft Corporation.

[7] OpenLode (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.joedog.org/index/siege-home

[8] OpenSTA (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.opensta.org/ .

[9] Pylot (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.pylot.org/ .

[10] S. Sampath, R. Bryce, Gokulanand Viswanath, Vani Kandimalla, A. Gunes Koru. Prioritizing User-Session-Based Test Cases for Web Applications Testing. Proceedings of the International Conference on Software Testing, Verification, and Validation (ICST), Lillehammer, Norway, April 2008.

[11] Siegle (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.opendemand.com/openload/

[12] Verisium – vTest (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.verisium.com/products/vTest/index.html.

[13]Web application testing – WAPT (Nazadnje obiskano: 4.6 2011) Dosegljivo na spletni strani: http://www.loadtestingtool.com/.