Travel.ch Integrationstests

Simon Lang

26. Juli 2015
Version 1.0.0

Studiengang	Informatik 5 Ba 2012
Semesterarbeit	Travel.ch Integrationstests
Dozent	Matthias Bachmann
Schule	ZHAW - School of Engineering

Kurzfassung

Vorliegende Arbeit stellt verschiedene Workstation-Umsetzungen, sprich Möglichkeiten Computer-Arbeitsplätze bereitzustellen, gegenüber.

Es wird zwischen on-premise (klassische Computer am Arbeitsplatz) und einer virtuellen Arbeitsumgebung unterschieden. Die virtuellen Umsetzungen lassen sich nochmals in internes und externes Hosting unterteilen.

Schlagwörter: Integration tests, Selenium, Testing

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

Bezeichnung	Beschreibung
SVN	Subversion

Glossar

Ausfallsicherheit

Mit der Ausfallsicherheit wird die minimale zeitliche Erreichbarkeit (resp. maximale Ausfallzeit) eines Systems angegeben. Ist diese Ausfallzeit sehr gering spricht man von Hochverfügbarkeit (High Availability), dazu ist mindestens eine Verfügbarkeit von 99.9 % nötig. Die Verfügbarkeit berechnet man wie folgt:

$$\text{Verfügbarkeit} = (1 - \frac{\text{Ausfallzeit}}{\text{Periode}}) * 100$$
$$\text{Ausfallzeit} = (1 - \frac{\text{Verfügbarkeit}}{100}) * \text{Periode}$$

Cloud

Der Begriff *Cloud* ist im **??** definiert.

on-premise

On-premise Software bezeichnet jegliche Ausführung von Software die vorwiegend auf dem Endgerät selbst läuft. Alternativ existiert das Cloud Modell, bei dem der Grossteil der Software auf einem Server ausgeführt wird.

Thin Client

Ein Thin Client ist ein günstiger, rechen-schwacher Computer. Er wird dazu verwendet, um Arbeiten zu erledigen die auf einem rechen-starken Server statt finden. Ein Thin Client übernimmt hauptsächlich die Bereitstellung von

KAPITEL 1

Einleitung

1.1 Ziele

Es soll ein detaillierter Überblick über das jetzige Angebot von *Workstation in der Cloud* erarbeitet werden. Steht ein Umdenken von der bisherigen *on-premise* Strategie zur Cloud-Philosophie an?

1.2 Begründung

Trotz extrem starker Entwicklung von Cloud-Angeboten in den letzten 10 Jahre, hat sich der Anwender kaum in diese Richtung bewegt. Zwar werden oft einzelne Dienste in die Cloud ausgelagert, jedoch kaum der ganze Arbeitsplatz. Dies würde einige Vorteile bieten.

KAPITEL 2

Beschreibung der Aufgabe

2.1 Ausgangslage

Alle 3 bis 4 Wochen wird auf der Webseite von Travelwindow AG eine neue Version eingespielt. Dies verlangt jedes Mal ein ausführliches Testen, was bislang immer manuell durchgeführt wurde. Dies ist eine sehr monotone und repetitive Arbeit, da die selben Testfälle auf unterschiedlichen Browsern überprüft werden müssen und dadurch oft Fehler übersehen werden. Der Testaufwand würde sich auf 3 bis 4 Tage belaufen, wenn alle kombinierbaren Möglichkeiten von Browser, Sprache und Produkten ausführlich geprüft werden müssen.

2.2 Ziele der Arbeit

Die Einführung von automatisierten Integrationstests sollen die repetitiven Testarbeiten minimieren. Diese sollen mittels Selenium umgesetzt werden. Das Ziel der Arbeit ist es, dass die Testpersonen bei jeder neuen Version nur noch testen müssen, was neu umgesetzt wurde. Dadurch sollen mehr Programmfehler aufgedeckt und menschliche Fehler ausgeschlossen werden. Später sollte durch die Automatisierung auch ein Continous Delivery ermöglicht werden.

2.3 Aufgabenstellung

In der Arbeit werden während vier Phasen die automatisierbaren Testfälle entwickelt.

2.3.1 Recherche

Zu beginn werden die einzusetzenden Frameworks, welche für die Entwicklung und Ausführung der Testfälle benötigt werden, evaluiert.

2.3.2 Anforderungen und Analyse

Als Grundlage für die Konzeptionierung werden bestmöglichst alle Testfälle spezifiziert. Diese sollen die Funktionalität der Webseite vollständig abdecken. Der Product Owner wird die Testfälle priorisieren, so dass entschieden werden kann nach welcher Reihenfolge die Testfälle konzeptioniert und implementiert werden können.

2.3.3 Konzept

Die Testfälle, welche während dieses Projektes umgesetzt werden, müssen aus spezifiziert werden. Sprich es sind Vorbedingungen, Testabläufe, Soll-Resultate und Nachbedingungen zu definieren. Zusätzlich wird die gesamte Infrastruktur beschrieben.

2.3.4 Umsetzung

Die Testfälle werden Implementiert und schlussendlich in den Application Lifecycle eingebunden, so dass sie automatisiert und zu definierten Zeitpunkten durchgeführt werden können. Je nach Anzahl der Testfälle werden alle oder ein Untermenge davon, in Abhängigkeit der erwähnten Priorisierung, umgesetzt.

2.4 Erwartete Resultate

Als Testvorbereitung werden die Testfälle spezifiziert und dokumentiert. Anschliessend werden verschiedene Testframeworks evaluiert und eine Architekturbeschreibung der Testinfrastruktur erstellt. Es werden alle möglichen Testfälle spezifiziert. Diese werden sehr umfangreich ausfallen weshalb Vorbedingungen, Testabläufe, Soll-Resultate und Nachbedingungen nur für jene Tests definiert werden, welche auch tatsächlich umgesetzt werden sollen. Die zu implementierenden Tests werden von der gesetzten Priorisierung bestimmt.

Der nächste Schritt ist die Implementierung der ausformulierten Tests. Dazu wird die Testinfrastruktur erstellt und die Tests programmiert. Ein vergleich mit den Ist-Resultate ist nicht nötig, da dies direkt von den automatischen Tests vorgenommen wird.

Schlussendlich müssen die Test noch in den Application Lifcecycle integriert werden.

- Testfallspezitifkation inkl. Priorisierung
- Evaluation von Testframeworks
- Architekturbeschreibung der Testinfrastruktur
- Aufsetzen der Testinfrastruktur
- Implementierte Testfälle
- Testfallintegration in den bestehenden Application Lifecycle

KAPITEL 3

Recherche

In diesem Kapitel werden die Rahmenbedingungen, die verschiedene Testing Frameworks sowie die Zielgruppe recherchiert.

3.1 Rahmenbedingungen

Getestet werden soll die Webseite der Firma *travelwindow AG*, welche unter `http://www.travel.ch` erreichbar ist. Die Seite ist mit C# implementiert, weshalb auch die Tests in dieser Programmiersprache umgesetzt werden sollen.

3.1.1 Infrastruktur

Es gibt folgende drei Umgebungen:

- Test
- Quality
- Production

Bei der Entwicklung wird jeder Checkin in das Versionierungssystem (siehe Abschnitt 3.1.2 Tooling) automatisch auf die Test-Umgebung aufgespielt. Sobald ein Stand erreicht ist, welcher vom Business der travelwindow AG getestet werden soll, wird diese Version auf die Quality-Umgebung geladen. Sobald diese fertig mit dem Testen sind, wird diese Version auf die Production-Umgebung gespielt und ist damit Live.

3.1.2 Tooling

Der Sourcecode der Softare ist auf einem eigenen Subversion (SVN)[1] Versioniert. Es ist jedoch eine Umstellung auf GIT[2] geplannt, weshalb neuer Sourcecode fortan nur noch auf dem GIT Hosting Dienst BitBucket[3] gespeichert werden soll.

1 *Apache Subversion.* URL: `https://subversion.apache.org/` (besucht am 26. 07. 2015).
2 *Git.* URL: `http://www.git-scm.com/` (besucht am 26. 07. 2015).
3 *Git and Mercurial code management for teams.* URL: `https://bitbucket.org/` (besucht am 26. 07. 2015).

Als Continuous Integration Server wird Bamboo[1] eingesetzt. Dieser Kompiliert den Sourcecode und ist dafür verantwortlich, eine Version auf die Umgebungen Test, Quality und Production zu laden.

3.2 Testing Frameworks

Der defacto Standart für Integrationstests von Benutzeroberfläche ist Selenium[2, 3]. Code für dieses Framework kann in C# programmiert werden, welcher einen Webbrowser startet und Benutzereingaben simuliert. Dadurch können repetitive Tests automatisiert und auf verschiedenen Browsern durchgeführt werden.

Die Tests können entweder auf einer eigens aufgesetzten Umgebung durchgeführt werden oder an einen Service ausgelagert werden.

3.2.1 lokale Ausführung vs Verwendung eines Services

Werden die Tests lokal oder auf einem Server innerhalb der Firma ausgeführt laufen die Tests sehr schnell ab und es muss nichts für die Ausführung der Tests bezahlt werden. Bei der Verwendung eines Services dauern die Tests länger und es fallen Laufzeikosten an. Die Serviceanbieter rechnen pro Minute ab, welche ein Test auf ihrer Infrastruktur läuft.

Der Vorteil der Serviceanbieter liegt darin, dass sie virtuelle Umgebungen für die Durchführung der Tests zur Verfügung stellen. Diese sind mit verschiedenen Betriebssystemen und Browser Konfigurationen ausgestattet. Einige Beispiele solcher Zusammenstellungen:

- Windows 7, Firefox 39
- Windows 8.1, Firefox 39
- OS X Yosemite, Safari 8
- IPhone, iOS 8.4
- Android 4.4, Samsung Galaxy S4

Es ist ersichtlich, dass auch Mobile Browser unterstützt werden. Das Aufsetzten und Pflegen verschiedener Servern mit verschiedenen Betriebssystemen, Browsern und Emulatoren für mobile Endgeräte ist aufwändig und benötigt viel Knowhow.

Als Service-Anbieter für Selenium Tests wurden zwei Kandidaten näher in Betracht gezogen. SauceLabs[4] und CrossBrowserTesting[5].

1 *Continuous Integration & Build Server - Bamboo | Atlassian.* URL: https://www.atlassian.com/software/bamboo (besucht am 26. 07. 2015).
2 *Selenium - Web Browser Automation.* URL: http://docs.seleniumhq.org/ (besucht am 26. 07. 2015).
3 *Happy 10th Birthday, Selenium | ThoughtWorks.* URL: http://www.thoughtworks.com/insights/blog/happy-10th-birthday-selenium (besucht am 26. 07. 2015).
4 *Sauce Labs: Selenium Testing, Mobile Testing, JS Unit Testing and More.* URL: https://saucelabs.com/ (besucht am 26. 07. 2015).
5 *Cross Browser Testing. Real mobile devices & browsers!* URL: https://crossbrowsertesting.com/ (besucht am 26. 07. 2015).

Beide Anbieter verfügen über 500 verschiedene Betriebsystem/Browser Konfigurationen[1, 2]. Beide können automatisierte Tests ausführen und liefern als Resultat ein Video der Durchführung sowie Screenshots der einzelnen Tests.

Vom Preis her unterscheiden sie sich auch nicht markant[3, 4]. SauceLabs hat das kleinste Packet mit 120 Minuten Automatisierten Tests für $12/Monat. Beide Anbieten verrrechnen für 10 Stunden Testing $49/Monat. Bei den teureren Packet für $149/Monat bietet CrossBrowserTesting mit 2000- mehr als Sauclabs mit 1800 Minuten.

3.2.2 Prototyp

Um die beiden Frameworks, welche im Abschnitt 3.2.1 lokale Ausführung vs Verwendung eines Services beschrieben wurden, zu testen wurde ein Prototyp entwickelt. Dieser kann die Tests auch lokal laufen lassen. Der Browser sowie der Selenium Treiber müssen dazu jedoch auf dem eigenen Rechner 11111installiert sein. Der Quellcode ist im BitBucket unter `https://bitbucket.org/soultemptation/tw-systemtests-prototype` einsehbar.

3.3 Zielgruppe

Die Zielgruppe sind die verschiedenen Browser, welche die Benutzer der Webseite verwenden. Die Tests werden schlussendlich auf den meist verwendeten Browsern durchgeführt. Auf *travel.ch* wird Google Analytics eingesetzt. Die Auswertung des letzten halben Jahres (25.02.2015 - 25.07.2015) ergab folgendes:

- Safari: 32.61%
  - 8.0: 53.40%
  - 7.0: 12.62%
- Internet Explorer: 26.62%
  - 11.0: 74.23%
  - 9.0: 13.95%
- Chrome: 21.14%
  - 40.0: 19.24%
  - 41.0: 15.82%
- Firefox: 14.22%
  - 36.0: 25.95%
  - 37.0: 21.53%

1 *Platforms.* URL: `https://saucelabs.com/platforms/` (besucht am 26. 07. 2015).
2 *OS & Browser Configurations for Cross Browser Compatibility Testing | Pick an OS Pick a Browser Test a Website.* URL: `https://crossbrowsertesting.com/browsers` (besucht am 26. 07. 2015).
3 *Sauce Labs: Pricing.* URL: `https://saucelabs.com/pricing` (besucht am 26. 07. 2015).
4 *CrossBrowserTesting Pricing Model.* URL: `https://crossbrowsertesting.com/pricing#plans` (besucht am 26. 07. 2015).

KAPITEL 4

Anforderungen und Analyse

KAPITEL 5

Diskussion

KAPITEL 6

Schlussfolgerung

Quellenverzeichnis

[1] *Apache Subversion*. URL: https://subversion.apache.org/ (besucht am 26. 07. 2015) (siehe S. 4).

[2] *Continuous Integration & Build Server - Bamboo | Atlassian*. URL: https://www.atlassian.com/software/bamboo (besucht am 26. 07. 2015) (siehe S. 5).

[3] *Cross Browser Testing. Real mobile devices & browsers!* URL: https://crossbrowsertesting.com/ (besucht am 26. 07. 2015) (siehe S. 5).

[4] *CrossBrowserTesting Pricing Model*. URL: https://crossbrowsertesting.com/pricing#plans (besucht am 26. 07. 2015) (siehe S. 6).

[5] *Git*. URL: http://www.git-scm.com/ (besucht am 26. 07. 2015) (siehe S. 4).

[6] *Git and Mercurial code management for teams*. URL: https://bitbucket.org/ (besucht am 26. 07. 2015) (siehe S. 4).

[7] *Happy 10th Birthday, Selenium | ThoughtWorks*. URL: http://www.thoughtworks.com/insights/blog/happy-10th-birthday-selenium (besucht am 26. 07. 2015) (siehe S. 5).

[8] *OS & Browser Configurations for Cross Browser Compatibility Testing | Pick an OS Pick a Browser Test a Website*. URL: https://crossbrowsertesting.com/browsers (besucht am 26. 07. 2015) (siehe S. 6).

[9] *Platforms*. URL: https://saucelabs.com/platforms/ (besucht am 26. 07. 2015) (siehe S. 6).

[10] *Sauce Labs: Pricing*. URL: https://saucelabs.com/pricing (besucht am 26. 07. 2015) (siehe S. 6).

[11] *Sauce Labs: Selenium Testing, Mobile Testing, JS Unit Testing and More*. URL: https://saucelabs.com/ (besucht am 26. 07. 2015) (siehe S. 5).

[12] *Selenium - Web Browser Automation*. URL: http://docs.seleniumhq.org/ (besucht am 26. 07. 2015) (siehe S. 5).