
**Fachhochschulstudiengang**
**Telematik/Netzwerktechnik**

# B A C H E L O R A R B E I T

## Einsatzmöglichkeit von Ajax für aktive Web-Informationssysteme

am Beispiel IronNetwork Live-Ticker

zur Erlangung des akademischen Grades

Bachelor of Science in Engineering (BSc)

Autor: Ing. Johannes Lang

Matrikelnummer: 0510286030

Betreuer: FH-Prof. Dipl.-Ing. Dr. Helmut Wöllik

Tag der Abgabe: 27. Juni 2008

Name: Ing. Johannes Lang
Matrikelnummer: 0510286030

Geburtsdatum: 22. Jänner 1965
Adresse: Rotschitzen, Waldweg 34,
9073 Klagenfurt – Viktring

# E i d e s s t a t t l i c h e  E r k l ä r u n g

Mit dieser Erklärung versichere ich die erstmalig vorgelegte Bachelorarbeit selbstständig erstellt zu haben.
Es wurden nur die von mir angegebenen Hilfsmittel und Quellen verwendet.

Datum: 27. Juni 2008
Ort: Klagenfurt

Unterschrift:

## Thema der Bachelorarbeit

Einsatzmöglichkeit von Ajax für aktive Web-Informationssysteme am Beispiel „IronNetwork Live-Ticker“

## Suchbegriffe

AJAX, GWT, Java, JBoss, IronNetwork Live-Ticker, Load Test, Webperformance

## Zusammenfassung

Diese Bachelor-Arbeit befasst sich mit den Performance Aspekten von Ajax Anwendungen, die die Anwender in Form von Informationssystemen ständig mit aktualisierten Informationen beliefern. Die Untersuchung basiert auf dem Web-Informationssystem „IronNetwork Live-Ticker“, das Inhalte einer Webseite aktiv mit Zeitnehmungsdaten einer Ironman Triathlon-Veranstaltung aktualisiert. Es wird die Belastung des Netzwerks durch die Aktualisierung der verbundenen Clients analysiert, die viele Stunden mit dem System verbunden sein können und wie ein zur Verfügung stehendes Serversystem in der Lage ist die Anfragen zu verarbeiten.

Es wird eine Lasttest-Software ausgewählt, die für den Test von Ajax Anwendungen geeignet und in der Lage ist, parametrisierte Abfragen von dynamischen Webseiten zu generieren und die vom Serversystem gelieferten Antworten auf ihre Richtigkeit hin zu überprüfen und die Antwortzeiten zu ermitteln.

Es werden dabei die Besonderheiten des eingesetzten Framework Google Web Toolkit (GWT) betrachtet und eine damit erstellte Applikation auf Optimierungspotenzial untersucht.

## Title of the paper

Fields of Application for Active Web Information Systems Investigating “IronNetwork Live-Ticker”


## Keywords

AJAX, GWT, Java, JBoss, IronNetwork Live-Ticker, load Test, Webperformance



## Abstract

This bachelor paper deals with the performance aspects of Ajax Appliances which supply users continuously with updated information in the form of information systems. The survey is based on the Web information system “IronNetwork Live-Ticker”, which actively updates the content of a website with time measurement data of an “Ironman” triathlon event. The load on the network due to the updating of the connected clients, which can be connected to the system for many hours, is analysed as well as the fact how an available server system can be capable of processing requests.

A load test software which is suitable for testing Ajax Appliances has been chosen that is capable of generating parameterised queries from dynamic websites, and of checking the responses supplied by the server system on their accuracy as well as identifying response times.

The special features of the adopted Framework Google Web Toolkit (GWT) are studied and an application generated with GWT is examined for optimisation potential.

# Inhaltsverzeichnis

Seite

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| AJAX | Asynchronous JavaScript and XML |
| DMZ | Demilitarized Zone |
| EAR | Enterprise Application Archive |
| EJB | Enterprise Java Beans |
| FTP | File Transfer Protocol |
| GWT | Google Web Toolkit |
| MVC | Model-View-Controller |
| PDA | Personal Digital Assistant |
| SFTP | SSH File Transfer Protocol |
| SQL | Structured Query Language |
| SSH | Secure Shell |
| WAR | Web Application Archive |
| XML | Extensible Markup Language |

# 1 Einleitung

Ajax Anwendungen bieten dem Benutzer des Web neue Möglichkeiten der Bedienung und des erhöhten Bedienungskomfort und sie bieten dem Entwickler Lösungsansätze, die bis vor wenigen Jahren nur mit Flash oder Java Applets realisierbar waren. Im fünften Semester 2007/2008 des Studiengangs Telematik/Netzwerktechnik der FH Kärnten wurde ein Web-Informationssystem mit dem Titel „IronNetwork Live-Ticker“ für Ironman Triathlon Sportveranstaltungen entwickelt, das Zuseher und Organisatoren einer Ironman Veranstaltung informiert, wann ein ausgewählter Athlet eine Zwischenzeitnehmung passiert hat. Zusätzlich liefert das System Ranking Listen, die permanent aktualisiert werden. Ziel des Projekts war, diese Funktion auf einer für einen großen Zuseherkreis erreichbaren Web-Plattform anzubieten. Dazu wurde eine Web-Applikation in Java entwickelt, die auf einem JBoss Applikationsserver ausführbar ist. Der IronNetwork Live-Ticker wurde mit den geforderten Funktionen fertig gestellt.

Außerdem wurden bei der Entwicklung noch Verbesserungspotentiale entdeckt, die in einer Weiterentwicklung umgesetzt werden können. Ein zusätzlich wichtiger Aspekt ist die Belastbarkeit des Systems. Bei einer laufenden Ironman Veranstaltung sind hunderte gleichzeitige Zugriffe möglich, die Inhalte dieser Verbindungen sind zusätzlich regelmäßig zu aktualisieren.

Das Ziel dieses Projekts ist die Verbesserung der Benutzerführung, die Umsetzung eines Model-View-Controller Konzepts für das Google Web Toolkit, die Implementierung des Systems auf einem Linux Server. Mit Belastungstests soll ermittelt werden, wie viele Benutzer bei der aktuellen Konfiguration auf das Informationssystem gleichzeitig zugreifen können.

Bei ausreichender Stabilität des Gesamtsystems ist ein Feldtest im Rahmen des Ironman 2008 Bewerbes am 13.7.2008 vorgesehen.

# 2 Aufgabenstellung

## 2.1 Zielsetzung

Ziel dieser Arbeit ist die Bewertung von Netzwerk- und Serverbelastung einer mit Google Web Toolkit in Java entwickelten Ajax Anwendung am Beispiel der Implementierung des Web-Informationssystems IronNetwork Live-Ticker für Sportveranstaltungen. Die Anwendung ist auf einem JBoss-Applikationsserver lauffähig und die Installation erfolgt auf einem Linux Serversystem.

## 2.2 Zielsetzung des Projekts

Ziel des Projekts ist die Erweiterung des im fünften Semester des Studiengangs Telematik / Netzwerktechnik entwickelten Web-Informationssystems IronNetwork Live-Ticker sowie die Implementierung auf einem Linux Server System, um das System für eine Ironman Veranstaltung lauffähig bereitstellen zu können.

In Lasttests wird die Serverauslastung und Netzwerklast der entwickelten Softwarelösung ermittelt, um aus den Ergebnissen Schlüsse für weitere Entwicklungen ziehen zu können.

Der Lasttest soll Ergebnisse über die mögliche Anzahl von gleichzeitigen Zugriffen liefern.

## 2.3 Rahmenbedingungen

### 2.3.1 Basissystem aus dem Vorprojekt

Im fünften Semester wurde ein Web-Informationssystem für Sportveranstaltungen entwickelt, Systembeschreibung und Komponenten sind in [LANG] und [SCHN] beschrieben. Auf Basis dieser Entwicklungen werden geplante Erweiterungen wie die Einführung eines Model-View-Controller Entwurfmusters und die Modifikation der Benutzerschnittstelle durchgeführt.

### 2.3.2 Server

Die Implementierung der Applikation erfolgte im fünften Semester auf den PCs der Projektmitarbeiter und war auf diesen vollständig lauffähig. In diesem Projekt ist geplant, den Applikationsserver auf einem Linux System in der demilitarisierten Zone (DMZ) des FH Netzwerks zu betreiben. Dazu wird ein virtueller Server unter der Server Virtualisierung OpenVZ [OPVZ] für das Projekt zur Verfügung gestellt.

Installation und Konfiguration des Servers, sowie Installation der erforderlichen Serversoftware sind Teile des Projekts. Für die Einrichtung und Installation steht ein Zugriff über eine Secure Shell (SSH) Verbindung zur Verfügung.

### 2.3.3 Server – Technische Daten

#### 2.3.3.1 Hardware

| | |
|---|---|
| HP Proliant 360 | 2 CPU Intel(R) Xeon(R) CPU 5130 @ 2.00GHz |
| | 6 GB RAM |
| | 4 x 160 GB Festplattenspeicher, Raid-5 |

#### 2.3.3.2 Software

| | |
|---|---|
| Betriebssystem | Debian GNU/Linux Version 4.2.3 Kernel 2.6.24 |
| Datenbank | MySQL V 5.0.51a-6 |
| Applikationsserver | JBOSS 4.2.1 GA |
| | EJB 3.0 |
| | GWT 1.5 Release Candidate 1 |
| | GWT-EXT Version 2.0.4 |

# 3 Theorie – Lasttest von Ajax Anwendungen

## 3.1 Einführung

Ein großer Teil der Zeit in einem Softwareentwicklungsprojekt ist dem Testen gewidmet. Um die Leistungsfähigkeit von Software und Hardware feststellen zu können, werden Lasttests durchgeführt. Im Gegensatz zu Unternehmensanwendungen und Intranetanwendungen, die für eine prognostizierte Anzahl von Benutzern ausgelegt sind, besteht bei Web-Applikationen die Problematik einer schweren Vorhersehbarkeit der Benutzeranzahl und der Benutzergewohnheiten. Bei Intranetanwendungen ist es das Ziel, mit einem Lasttest, die Überprüfung der bereitzustellenden Leistungsfaktoren, wie Antwortzeit, bei vorgegebener Transaktionsanzahl zu ermitteln. Bei Web-Anwendungen besteht zusätzlich der Bedarf, Informationen über das Systemverhalten im Grenzbereich der vorgegebenen Lastgrenzen und auch das Verhalten bei Überlastung zu ermitteln. Webseiten mit statischen Inhalten besitzen meist ein lineares Verhältnis zwischen Belastung und Antwortzeit sowie Transferleistung. Systeme, die im Hintergrund Transaktionen in Datenbanksystemen abarbeiten, weisen häufig ein exponentielles Verhältnis zwischen Serveranfragen und Antwortzeit auf, was zu dem Effekt führt, dass geringe Überlastungen zu Antwortzeiten führen können, die von Benutzern als Systemausfall wahrgenommen werden.

Die Überlastung von dynamischen Web-Systemen kann zu einem extremen Anstieg von CPU- und Speicherauslastung führen und damit die Verarbeitungszeit für Transaktionen stark verlängern. Der Benutzer von beeinträchtigten Web-Anwendungen drücken bei zu lange empfundener Wartezeit die „Aktualisieren“ Schaltfläche auf dem Browser und bringen damit einen zusätzlichen Anstieg von abzuarbeitenden Transaktionen. Die Folge kann ein Totalabsturz sein oder das System wird viele Stunden mit der Verarbeitung der angeforderten Transaktionen belastet, bevor wieder ein normaler Betriebszustand eintritt.

Die Lasttests liefern Informationen über diese extremen Betriebszustände und bieten eine Entscheidungshilfe für die Systemauslegung.

## 3.2 Lasttest für Web-Anwendungen

### 3.2.1 Lasttest von statischen Webseiten

Bei der Durchführung von Lasttests an Web-Anwendungen werden Parameter wie Datendurchsatz und Antwortzeiten ermittelt. Dazu werden über http requests, HTML Seiten von dem zu testenden Web-Server angefordert. Die Zeit bis zum Eintreffen der HTML Seite sowie der eingebetteten Elemente wird gemessen. Zusätzlich wir ermittelt, ob alle Elemente der Seite übermittelt wurden. Es muss daher vorausgesetzt werden, dass die zu testenden Seiten fehlerfrei vorhanden sind.

Ein Testszenario kann einfach über eine Zusammenstellung der abzurufenden Seiten in ihrer Reihenfolge zusammengestellt werden und an das Lasttest System übergeben werden. Das vorbereitete Testszenario wird vom Lasttest System parallel ausgesandt und die zuvor beschriebenen Messwerte ermittelt.

Die Lasttest-Programme stellen für die Zusammenstellung von Testszenarien Aufzeichnungstools zur Verfügung, die einen raschen Aufbau eines Testablaufs erlauben. Abweichende Testabläufe sind bei modernen Web-Servern meist nicht erforderlich, da auf Grund der Speicherausstattung der Server der gesamte Web-Inhalt im Cache der Web-Server Applikation des Servers gehalten wird.

### 3.2.2 Lasttest von Ajax Anwendungen

Der Test von herkömmlichen Web-Auftritten mit HTML Seiten ist durch die Vorhersehbarkeit, der vom Server zu liefernden Ergebnisse, relativ einfach. Ajax ist ein Begriff der für „Asynchronous JavaScript and XML“ steht, beschrieben ist der Einsatz von Ajax für das Projekt in [LANG] und in [AJIP] Die Entwicklung von Ajax Anwendungen ist nicht nur ungleich aufwendiger als das Erstellen von statischen HTML Seiten, sondern auch das Testen einer Ajax Web-Anwendung beansprucht viel mehr Vorbereitung. Der Testablauf ist auch umfangreicher in der Durchführung. Der Aufruf einer URL einer Ajax Anwendung bietet keinen Rückschluss über die zu erwartenden Ergebnisse. Die Eigenschaft von Ajax Anwendungen, Abfrageergebnisse in eine bereits geladene Seite nachzuliefern,

erfordert mehr Leistung vom Serversystem für die Verarbeitung der xml-http requests als reine HTML Seiten, ist aber gleichzeitig Ressourcenschonender im Netzwerk. Es können in eine bereits geladene Seite, Informationen nachgeladen oder aktualisiert werden, wie zum Beispiel das Laden einer Artikelliste nach Auswahl einer Produktgruppe in einem Shopsystem. Ohne Ajax wäre das neuerliche Laden der gesamten Seite erforderlich. Die Beurteilung der Vollständigkeit der gelieferten Abfrageergebnisse kann bei Ajax Seiten nicht über die Analyse der HTML Struktur der Seite erfolgen.

Da intelligente Applikationsserver mit Caching Mechanismen ausgestattet sind, die Daten einer Benutzersitzung speichern um die Verarbeitungszeit zu minimieren, kann es beim Erstellen von Testszenarien erforderlich sein, die Eingabe von unterschiedlichen Benutzerkennzeichen oder Artikelcodes zu simulieren.

### 3.2.3 Anforderungen an Ajax Lasttests

Für das Erstellen von einem Testszenario für eine Ajax Anwendung ist eine gründliche Planung erforderlich, in der die möglichen Anwendungsfälle aufgenommen werden. Bei der Gestaltung der Mustergeschäftsfälle sind auch die unterschiedlichsten Zustände von Transaktionen einzuplanen, wie nicht abgeschlossene Warenkörbe in Shopsystemen, die bis zum Ablauf einer Karenzzeit gespeichert bleiben. Bei Testszenarien ist auch zu beachten, dass Datenbanksysteme mit unterschiedlichen Abfragen belastet werden, damit diese den Test nicht ausschließlich aus ihren Caches bedienen können. Bei einem Shopsystem sind zum Beispiel von verschiedenen Musterkunden die unterschiedlichen Produktgruppen auszuwählen um eine breite Belastung über das gesamte Warenangebot in der Datenbank zu erzeugen.

### 3.2.4 Anforderungen an Ajax Lasttest Systeme

Die zuvor beschriebenen Anforderungen an Lasttests für Ajax Applikationen sollten Lasttest Systeme auch erfüllen können. Die Anforderungen an die Testsysteme, die sich dadurch ergeben sind:

- Parallel ablaufende Testszenarien mit unterschiedlichen Eingabeparametern. Zum Beispiel:
    - Kunde A, wählt Produkt 1,7,9, löscht 7 aus Warenkorb, beendet Einkauf mit Kreditkartenzahlung.
    - Kunde B, durchsucht Produktgruppe A2, Seite 1 bis 15, wählt Produkt 44 ohne den Einkauf abzuschließen.
- Hilfsmittel für die Aufzeichnung von Geschäftsfällen für den Testablauf.
    - Aus der Datenbank nachladbare Inhalte sollten auswählbar sein, wie etwa durch Auswahl einer Produktgruppe lädt die Ajax Anwendung die Produktuntergruppen in eine weitere Auswahlliste.
    - Ein nicht mehr lagerndes Produkt kann nicht mehr in den Warenkorb übernommen werden – Bedingungen sind für diese Fälle in das Szenario zu integrieren.
- Werkzeuge für die Festlegung von Eingabewerten in Formularfelder.

Zu beachten ist, dass das Lasttestsystem nicht das Ausführen von Javascript emuliert, sondern die Ereignisse und Interaktionen detailliert aufzeichnet und für den Ablauf des Szenarios diese Interaktionen detailgetreu wiedergibt.

### 3.2.5 Ablauf des Lasttests

Die Leistungsfähigkeit einer Web-Applikation hängt von Faktoren wie CPU-Auslastung, Speicherauslastung, Netzwerklast, Datenbank ab. Bei der Durchführung eines Lasttests ist daher eine Überwachung des Server- und Datenbank Systems erforderlich, um während dem laufenden Test Flaschenhälse erkennen zu können und diese so weit wie möglich auszuschalten.

Der Lasttest ist kein einmaliger Vorgang, sondern ein mehrstufiger Prozess, in dem unterschiedliche Testansätze gewählt werden. In einem ersten Test wird die Anzahl der virtuellen Benutzer langsam erhöht, bis die geplante Anzahl von gleichzeitig zugreifenden Benutzern erreicht ist. Dieser Test sollte solange fortgesetzt werden, bis eine maximale Benutzeranzahl durch Erreichen der Lastgrenze ermittelbar ist.

In einer weiteren Testphase wird in einem mehrstündigen Dauertest die Langzeitstabilität überwacht. Ein weiterer Test gibt Auskunft über das Verhalten bei schlagartiger Änderung der Last, dabei wird das System einer Grundlast ausgesetzt und in kurzer Zeit die Anzahl der virtuellen Benutzer auf 80% der ermittelten Maximalbelastung erhöht. Nach kurzer Zeit sollte das System den Belastungsanstieg verarbeitet haben und wieder stabil laufen.

#### 3.2.5.1 Planung von Lasttests

In der Planung sind die Testgeschäftsfälle auszuwählen und vorzubereiten und eine Testsoftware für die geforderte Anforderung zu wählen. Für das Testsystem ist eine geeignete Hardware vorzubereiten, Informationen über die erforderliche Hard- und Softwareausstattung sind für die gewählte Testsoftware einzuholen.

Für den Testablauf ist die Zusammenstellung der geplanten virtuellen Benutzer erforderlich. Am Serversystem ist sicherzustellen, dass ein Performance Monitor für CPU, Speicher und Netzwerküberwachung zur Verfügung steht.

#### 3.2.5.2 Auswahl einer Testsoftware

Der Softwaremarkt bietet eine Vielzahl von Lasttestsystemen für Web-Applikationen mit unterschiedlichsten Leistungsspektren und Lizenzmodellen. Viele angebotene Testplattformen sind auch für den Test von Ajax Anwendungen geeignet. Bei der Auswahl des Systems sind Möglichkeiten der Parametrisierung von Testfällen zu berücksichtigen, Hilfsmittel zur Aufzeichnung von Testgeschäftsfällen und die Zusammenstellung von unterschiedlichen Testgeschäftsfällen zu einem gesamten Testszenario. Systeme, die ausschließlich über Scripts zu steuern sind, erfordern eine umfangreiche Einarbeitungszeit. Viele Testanwendungen erlauben die Steuerung von Lastgeneratoren, die einen über mehrere Lastquellen verteilten Test durchführen. Die Steuerung erfolgt zentral über das testende System und die Platzierung der Lastgeneratoren kann wahlfrei im Internet erfolgen. Diese Funktion wird von professionellen Unternehmen genutzt um die Erreichbarkeit und Performance von unterschiedlichen Kontinenten und Ländern aus zu ermitteln.

#### 3.2.5.3 Vorbereitung zum Test

Die vorbereiteten Testgeschäftsfälle sind im Testsystem aufzubereiten, wie zum Beispiel die Parameterdateien mit Benutzernamen oder Artikelcodes, die im Test wahlweise aufgerufen werden sollen. Die erforderlichen Einträge sind auch im zu testenden System vorzubereiten, da im Test verwendete Parameter, die im zu testenden System nicht abgebildet sind, zu Fehlern führen. Weiters sind durch fehlerhafte Parameter Abweichungen in den Messergebnissen möglich, bei besonders hoher Anzahl von fehlerhaften Parametern, kann diese zu einem Abbruch des Tests führen. Durch einen Test der Testgeschäftsfälle sollte ihre Funktionsfähigkeit sichergestellt werden. Eine Netzwerkanbindung für den / die Lastgenerator/en mit ausreichender Bandbreite sind vorzubereiten.

#### 3.2.5.4 Testdurchführung

Die Durchführung des Lasttests sollte in den geplanten Phasen ablaufen, wobei es zu einem iterativen Prozess führen kann, wenn wiederholt Flaschenhälse auftauchen, die zu einer Adaption in der Web-Applikation, am Betriebssystem oder im Netzwerks führen. Testergebnisse können umfangreiche Anpassungen an den Servereinstellungen erforderlich machen. Eine Überwachung des Last generierenden Systems ist ebenfalls erforderlich um sicherzustellen, dass die zu erzeugende Belastung generiert werden kann.

#### 3.2.5.5 Auswertung

Die wichtigsten Messergebnisse, die von Lasttestsystemen geliefert werden sind die mittlere Antwortzeit und der Datendurchsatz im Verhältnis zu den aktiven virtuellen Benutzern. Die Antwortzeit sollte bei einer Webseite unter 10 Sekunden betragen, darüber wird die Wartezeit von Benutzern meist nicht abgewartet und abgebrochen. Da bei Ajax Anwendungen die Möglichkeit besteht den Ladevorgang der Seite anzuzeigen, wird die Konzentration des Benutzers auf die sich verändernde Seite verlängert.

## 3.3 Lasttest und Web Usability

*„Todsünde Nummer Vier*

*Der unendliche Ladevorgang*

*Gespräche zwischen Freunden können längere Schweigepausen überleben, aber nur wenige Webseiten sich lange Ladezeiten leisten“* [SIEG]

Die Auswertung vom Lasttest zeigt, ab wie vielen gleichzeitigen Zugriffen auf die Web-Applikation die Antwortzeit so lange wird, dass Benutzer das Warten aufgeben und die „Aktualisieren“ Schaltfläche drücken oder den Browser schließen.

*„Zehn Sekunden sind das Limit, um die Aufmerksamkeit des Benutzers auf einen Dialog zu konzentrieren. Bei längeren Verzögerungen wendet sich der Benutzer anderen Aufgaben zu, während er wartet, dass der Computer seine Arbeit beendet.“* [NIEL]

Um eine Web-Applikation für den Benutzer attraktiv zu halten, ist auf Grund der Ergebnisse des Lasttests Potenzial für die Optimierung der Applikation auszumachen.

Ein wichtiger Faktor, der mit Lasttests nicht ermittelt werden kann jedoch die Antwortzeiten von Ajax Web-Anwendungen maßgeblich beeinflussen kann, ist die Ausführungsgeschwindigkeit der Javascript Routinen im Browser des Endbenutzers. In vielen Fällen ist die Start- und Ausführungszeit von Javascript Routinen vielfach länger als die Dauer der Datenübertragung. Dies kann besonders bei leistungsschwachen PCs oder mit beschränkter CPU Leistung ausgestatteten PDAs oder Smartphones zum Tragen kommen.

## 3.4 Usability von Ajax Anwendungen

Web-Anwendungen können durch den Einsatz von Ajax den Anwendern den Vorteil einer besseren Bedienbarkeit bieten. Bedienelemente können interaktiv gestaltet werden, Inhalte durch xml-http requests nach dem Seitenaufbau aktualisiert und abhängig vom Kontext mit neuen Inhalten geladen werden. Diese Interaktivität bringt jedoch nur solange Vorteile für den Benutzer, wie sie intuitiv bedienbar sind und die Anwendung mit den Features ausgestattet wird, die eine Erleichterung bringen. Besonders wichtig ist, dass die interaktiven Elemente für den Benutzer erkennbar sind. Dies kann durch schrittweise Führung mit einem Dialog erfolgen, in dem zum Beispiel aktualisierte Felder hervorgehoben werden. Ajax Anwendungen haben jedoch den Ruf besonders schwer bedienbar zu sein,

da Benutzer mit Ajax Funktionen überfordert werden, wo diese nicht erforderlich sind und die für die Bedienbarkeit keine Vorteile bringen. Der Einsatz von Ajax bringt bei Web-Anwendungen dort Vorteile, wo bewährte Bedienkonzepte durch Interaktivität ergänzt werden.

# 4 Projektumsetzung

## 4.1 Ausgangssituation

Aus dem Projekt im fünften Semester steht eine Web-Informationsplattform zur Verfügung, die als Basissystem die grundlegenden Funktionen erfüllt. Um die Bedienung zu erleichtern, sind Detailverbesserungen in der Benutzerführung erforderlich. Des Weiteren würde eine Modularisierung im Programmcode die Erweiterbarkeit erleichtern. Um das System für den Einsatz in einem Feldversuch vorzubereiten, ist die Installation auf einem dedizierten Web-Server vorgesehen. Der Ironman Triathlon findet am Veranstaltungsort in Klagenfurt nur einmal jährlich statt. Sollte das System an diesem Veranstaltungstag eingesetzt werden, muss die Stabilität und Leistungsfähigkeit gewährleistet sein, um eine festzulegende Anzahl an Zugriffen an diesem Veranstaltungstag verarbeiten zu können.

## 4.2 Anforderungen

Die allgemeinen Anforderungen an das System und die zur Verfügung stehenden Schnittstellen sind in [LANG] beschrieben. Die besonderen Anforderungen dieser Projektumsetzung sind:

- Entwicklung eines Model-View-Controller Konzepts für GWT.
- Installation der Serversoftware auf einem Linux Serversystem.
- Optimierung der Benutzerschnittstelle.
- Durchführen von Lasttests zur Ermittlung der Leistungsfähigkeit des Systems.

### 4.2.1 Model-View-Controller Konzept

Bei der Entwicklung von Benutzerschnittstellen für die Verarbeitung von Daten wird häufig auf das Model-View-Controller Konzept (MVC, Modell – Präsentation – Steuerung) zurückgegriffen, das eine Strukturierung der Aufgaben mit einzelnen Modulen in der Softwareentwicklung erlaubt. Die einzelnen Komponenten des MVC sind in Abbildung 1 dargestellt. Das Modell stellt die Daten zur Verfügung und kann auch die Geschäftslogik enthalten. Die Darstellung der Daten wird von

der Präsentationsschicht übernommen, die diese vom Modell abruft. Die Benutzeraktionen werden vom Controller entgegengenommen, verarbeitet und durch die Benutzeraktion geänderten Daten an das Modell übergeben.

Für GWT steht kein MVC als Standardimplementierung zur Verfügung und wurde in der Umsetzung im Projekt des letzten Semesters nicht berücksichtigt. Die Erweiterbarkeit des Systems wird durch MVC verbessert.

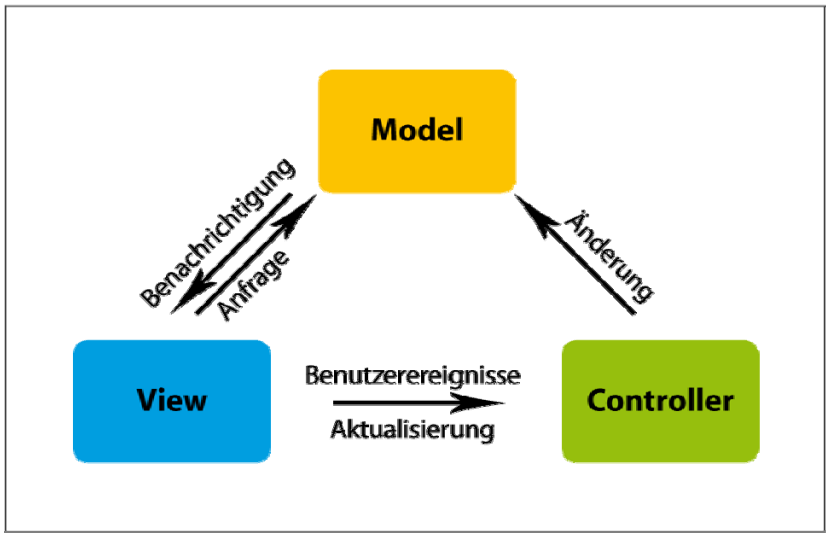



Abbildung 1: Model-View-Controller Konzept

### 4.2.2 Installation auf Linux System

Für die Realisierung des IronNetwork Live-Ticker ist in der DMZ des FH-Netzwerks ein Linux Server vorgesehen. Es wird ein Debian GNU/Linux Server zur Verfügung gestellt, Installation und Konfiguration von JBoss und MySQL ist durchzuführen. Die Installation und Konfiguration erfolgt über einen Terminalzugang mittels SSH.

### 4.2.3 Optimierung der Benutzerschnittstelle

Die Anforderungen an Web-Anwendungen bezüglich Bedienbarkeit sind im Gegensatz zu herkömmlichen Desktopanwendungen vollkommen anders. Dies gilt insbesondere für die Anwendung IronNetwork Live-Ticker, die nur an einem Veranstaltungstag zum Einsatz kommt, ohne zuvor von den Benutzern jemals verwendet worden zu sein. Es ist daher zu berücksichtigen, dass die Benutzerschnittstelle des IronNetwork Live-Tickers einfach zu bedienen ist, da

dem Anwender keine Eingewöhnungsphase zur Verfügung steht, um die Bedienung des Systems kennen zu lernen.

### 4.2.4 Durchführen von Lasttests

Die Besonderheit, dass der IronNetwork Live-Ticker an einem Veranstaltungstag im Einsatz ist, erfordert die Betrachtung der Leistungsfähigkeit des Gesamtsystems um sicherzustellen, dass das System den zu erwartenden Anfragen am Veranstaltungstag standhält. Für die Auslegung des Lasttests wird eine Maximalzahl von gleichzeitig 500 Benutzern angenommen. Die Teilnehmerzahl der Ironman Veranstaltung ist mit 2000 begrenzt. Für die Maximalanzahl an gleichzeitig verbundenen Benutzern wird willkürlich ein Viertel der Athletenanzahl angenommen. Die Auswahl eines Lasttest Tools ist Teil des Projekts. Die Ergebnisse sollen einen Rückschluss auf die Belastbarkeit des Systems mit 500 Benutzern geben.

## 4.3 Konzept

### 4.3.1 Model-View-Controller Konzept

Die Implementierung der Benutzerschnittstelle des IronNetwork Live-Ticker in der Vorstufe zu diesem Projekt mit Google Web Toolkit (GWT) hat gezeigt, dass die Integration der Datenpräsentation, Überprüfen von Feldinhalten, Aktualisieren von Daten und Datenbankschnittstelle in einem Modul mit zunehmenden Umfang zu Problemen in der Programmierung geführt hat. Die Einführung eines Model-View-Controller Konzepts wurde bereits in der Vorprojektphase als erforderlich erkannt, da bei Verbesserungen und Anpassungen der Benutzerschnittstelle immer eine Reihe von Veränderungen an den Routinen für die Datenmanipulation erforderlich wurden. Der Einsatz des Model-View-Controller Konzepts im IronNetwork Live-Ticker ist in [SCHN2] beschrieben.

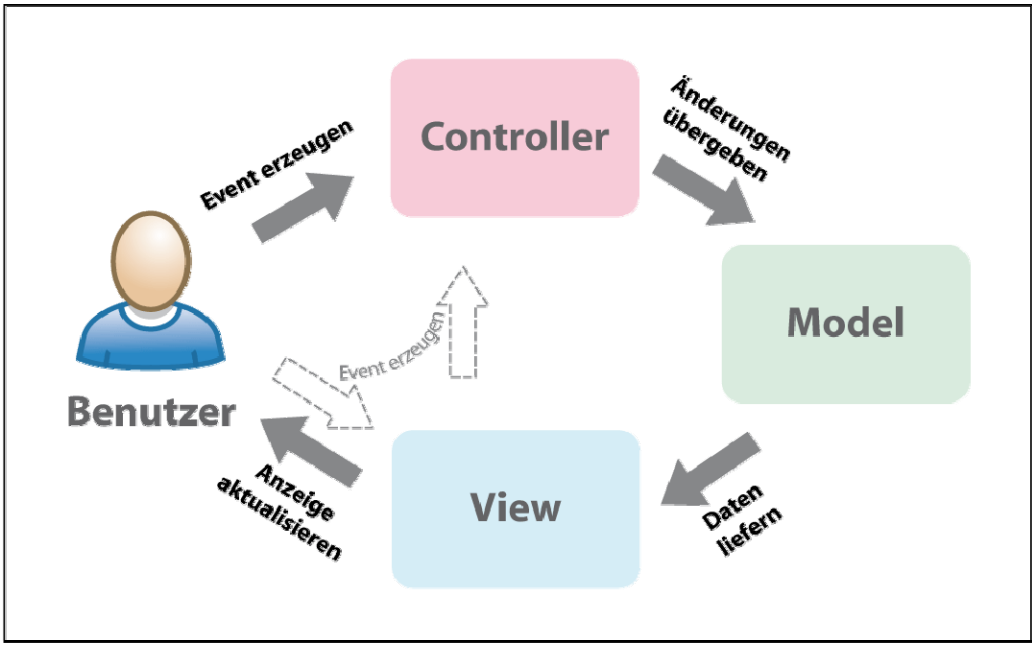



Abbildung 2: MVC Benutzerinteraktion

Die Untersuchung des Anwenderverhaltens bei der Benutzung des IronNetwork Live-Tickers hat ergeben, dass die Struktur des Benutzerinterface verbessert werden sollte. Reduktion der Eingabefelder, Vereinfachung der Benutzereingaben und Vereinheitlichung der Bedienung hat gezeigt, dass die Integration von Feldbedingungen und Datenbankabfragen in der Implementierung des Userinterface eine Vielzahl von redundanten Methoden und damit einen hohen Update- und Wartungsaufwand im Programmcode gebracht hat.

Es gibt eine Vielzahl von unterschiedlichen Model-View-Controller Entwurfsansätzen, die ebenso unterschiedliche Vor- und Nachteile besitzen. Im IronNetwork Live-Ticker wurde ein Event gesteuerter Ansatz gewählt, bei dem alle Controller in einem Eventhandler registriert werden. Bei einem eingetretenen Ereignis werden die registrierten Controller informiert und verarbeiten daraufhin die erforderlichen Informationen. Dieses Modell wurde für die Zusammenarbeit mit GWT gewählt. Die technischen Hintergründe sind in [SCHN2] beschrieben.

### 4.3.2 Konfiguration des Produktivsystems

Als Produktivsystem steht eine Debian GNU/Linux Server zur Verfügung. Für die Installation von Java, Datenbankserver und Applikationsserver JBoss sind die Installationsanleitungen zusammenzustellen. Für die Installation steht

ausschließlich die Kommandozeile über SSH zur Verfügung, die Installation einer grafischen Benutzeroberfläche für den Installationsvorgang ist nicht vorgesehen. Vom Internet ist der Server nur über die Ports http (80) und ssh (22) erreichbar. Alle weiteren Ports sind an der Firewall gesperrt. Abbildung 3 zeigt die Netzwerkkonfiguration des Produktivsystems. Der Applikationsserver und der Server für Zeitnehmungsdaten befinden sich im lokalen Netzwerk (LAN) der DMZ der FH-Kärnten und sind über eine Firewall vom öffentlichen Internet getrennt. Die Clients verbinden sich über das Internet mit dem Applikationsserver.

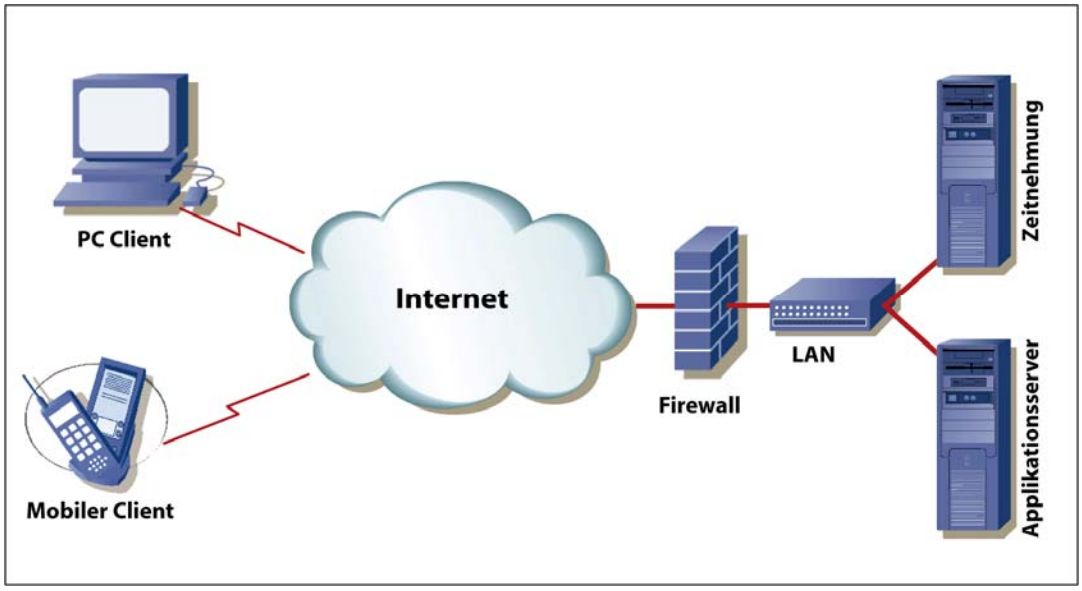



Abbildung 3: Netzwerk Konfiguration

### 4.3.3 Usability

Um den IronNetwork Live-Ticker für den Benutzer möglichst einfach bedienbar zu machen, sind die Anwendungsfälle, die voraussichtlich am häufigsten benutzt werden, mit einem Mausklick erreichbar in das Benutzerinterface zu implementieren. Mit einem Durchspielen der einzelnen Anwendungsfälle sind die möglichen Pfade zwischen den einzelnen Seiten des Systems ermittelbar und Verknüpfungen zwischen zusammenhängenden Seiten zu finden. Wenn es für die einfache Bedienung erforderlich ist, werden zusätzliche Links erstellt.

### 4.3.4 Migration auf Google Web Toolkit 1.5 RC 1

Mit Projektstart steht die Version des Google Web Toolkit in der Version 1.5 Release Candidat 1 (RC) zur Verfügung. Die Migration des IronNetwork Live-Ticker auf das Google Web Toolkit 1.5 hat für die Entwicklung folgende Vorteile:

- Der Sprachumfang von Java 1.5 wird größtenteils unterstützt.
- Der GWT Java Script Compiler erzeugt schnelleren und kompakteren Code.
- Erweiterungen an der User Interface Bibliothek wurden eingebracht.
- Die Unterstützung von mehrsprachigen Anwendungen wurde verbessert.

Exkurs GWT 1.5: GWT erlaubt die Entwicklung von Web-Benutzerschnittstellen in Java. Dieser Java Quellcode wird vom GWT Compiler in Javascript übersetzt. GWT 1.4 unterstützt den Sprachumfang von Java 1.4. Die Entwicklung der Serverapplikation erfolgt in Java 1.5, der Einsatz von GWT 1.5 erlaubt die fließende Einbindung des GWT Quell Code in das Projekt ohne Berücksichtigung des reduzierten Sprachumfangs von Java 1.4, wie es mit GWT 1.4, der Fall ist und damit können weitere Fehlerquellen ausgeschlossen werden.

### 4.3.5 Auswahl einer Lasttest Software

Es ist zu evaluieren, welche verfügbaren Produkte für die Durchführung eines Lasttests geeignet sind, welche Voraussetzungen erfüllt sein müssen und danach ist ein geeignetes Testszenario zu erstellen.

## 4.4 Linux Serversystem Installation

Dieser Abschnitt beschreibt die Installation des Linux Server Systems. Er beinhaltet Informationen welche Hardware- und Softwarevoraussetzungen vorhanden sind, welche Tools eingesetzt wurden und eine Beschreibung der Installationsschritte. In Abbildung 4 sind die am Linux Server installierten Komponenten schematisch mit ihren Abhängigkeiten dargestellt.

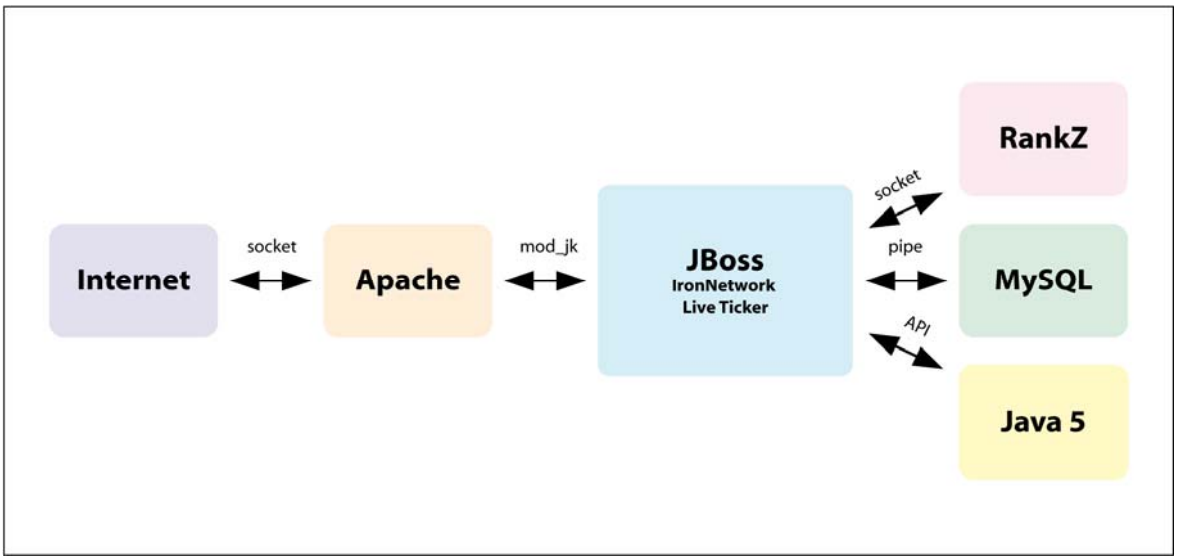



Abbildung 4: Linux Server Komponenten

### 4.4.1 Voraussetzungen

Im Folgenden sind die erforderlichen Systeminformationen und die für die Installation erforderlichen Hilfsmittel zusammengestellt.

#### 4.4.1.1 Zugangsvoraussetzung

Der Server befindet sich im Serverraum der FH-Kärnten und ist im Netzwerk in der DMZ eingerichtet. Der physische Zugang ist nicht möglich. Installation, Konfiguration und Wartung erfordern einen SSH-Terminalzugang. Über das Internet sind nur die Ports http (80) und ssh (22) frei geschaltet, für erweiterten Netzwerkzugriff für MySQL (3306) und Remote Method Invocation (RMI) Registry (1099) sind SSH Tunnel zu verwenden. Diese können z.B. mit dem Terminalemulations Programm Putty [PUTTY] eingerichtet werden. Ein Konfigurationsbeispiel ist im Anhang beigefügt.

#### 4.4.1.2 Hilfsmittel

Für die Installation eingesetzte Software Werkzeuge sind:

- Terminal Emulation: Putty [PUTTY], Poderosa [PODE]
  Diese Programme unterstützen SSH und SSH Tunnel für die Konfiguration der Datenbank und Überprüfung der Installation. Für die Einrichtung von SSH Tunnel mit Poderosa ist die Installation eines Plugin erforderlich, das von [POND] geladen werden kann.
- FTP Client: CuteFTP LE
  Das verwendete FTP Client Programm muss das Protokoll SFTP unterstützen, um Konfigurations- und Programmdateien auf den Server übertragen zu können.
- HTML Editor und FTP Client: Adobe GoLive.
- Datenbank Administration: MySQL Admin, MySQL Querybrowser [MYSQL]
  Diese Programme werden für die Konfiguration der Datenbank, Datenbank Benutzer und die Einrichtung der erforderlichen Datentabellen sowie der Berechtigungen verwendet.
- Entwicklungsframework: Eclipse [ECL]
  Für die Anlage von Testdaten und die Überprüfung der Konfiguration wird Eclipse mit einem SQL Plugin eingesetzt.

### 4.4.2 Installation allgemein

Das vorhandene Server Betriebssystem Debian GNU/Linux bietet eine Vielzahl von Installationspaketen, die eine einfache Einrichtung von Server Applikationen ermöglicht. Die verfügbaren Installationspakete können mit dem Advanced Packaging Tool (APT) installiert werden. Die Installation eines vorgefertigten Pakets erfolgt mit dem Befehl:

```
apt-get install <Paketname>
```

Weitere Informationen über den Einsatz des APT-Tools sind unter [APT] zu finden.

### 4.4.3 Installation MySQL Server

Für den MySQL Server existiert ein APT Paket, es ist jedoch zu beachten, dass für die volle Funktionsfähigkeit drei Pakete mit folgenden Befehlen zu installieren sind:

```
apt-get install mysql-server
apt-get install mysql-client
apt-get install libmysqlclient15-dev
```

Die Standardinstallation von MySQL erlaubt aus Sicherheitsgründen nur Netzwerkverbindungen vom installierten Rechner. Um den Zugriff von anderen Rechnern zu ermöglichen ist in der Konfigurationsdatei die Zeile:

```
bind-address            = 127.0.0.1
```

auszukommentieren:

```
# bind-address          = 127.0.0.1
```

(vim: i für insert-mode und # eingeben. Beenden des vim mit ESC Taste, ZZ)

und anschließend ist der MySQL Server neu zu starten:

```
/etc/init.d/mysql restart
```

Die weitere Konfiguration des MySQL Servers kann nach Einrichtung eines SSH Tunnels für das TCP-Port 3306 mit MySQL Admin durchgeführt werden.

### 4.4.4 Installation Java

Für den Betrieb von JBoss ist Java 5 erforderlich, die Installation erfolgt durch Eingabe von:

```
apt-get install sun-java5-jdk
```

Für die Installation von Java sind keine weiteren Schritte erforderlich.

### 4.4.5 Installation JBoss

Für JBoss steht kein von JBoss unterstütztes APT Package für die Installation zur Verfügung, daher ist die JBoss Installation manuell durchzuführen. Dazu ist als Vorbereitung eine Internet Quelle für JBoss auf [JBSRC] auszuwählen. Die Installation von JBoss wurde nach der Anleitung auf [JBIN] durchgeführt.

#### 4.4.5.1 Installationsverzeichnis erstellen

JBoss wird in den Ordner /opt installiert. Nach dem Download des JBoss-Binary Archivs mit

```
wget <url>jboss-4.2.1.GA.zip
```

wird dieses Archiv mit

```
cd /opt && unzip ~/jboss-4.2.1.GA.zip
```

in den vorgesehenen Ordner entpackt.

Im nächsten Schritt wird eine Benutzergruppe „jboss“ angelegt,

```
groupadd jboss
```

danach ein Benutzer „jboss“ angelegt, der der Benutzergruppe „jboss“ angehört und das Home-Verzeichnis jboss erhält.

```
useradd -g jboss -m jboss
```

Die installierten Programmdateien werden dem Benutzer „jboss“ und Gruppe „jboss“ zugeordnet.

```
chown -R jboss:jboss /opt/jboss-4.2.1.GA
```

Um bei zukünftigen Installationen vom Installationspfad unabhängig zu bleiben, wird ein Symlink “jboss” auf das Installationsverzeichnis gelegt.

```
ln -s /opt/jboss-4.2.1.GA jboss
```

Für das Logging ist ein Ordner anzulegen und dem „jboss“ Benutzer zuzuordnen.

```
mkdir /var/log/jboss/Set
chown jboss:jboss /var/log/jboss/
```

#### 4.4.5.2 JBoss in Systemstart einbinden

Damit der JBoss Applikationsserver mit dem Server gestartet wird, ist im Verzeichnis /etc/init.d die Datei jboss mit folgendem Script anzulegen:

```
#!/bin/sh
start(){
     echo "starting jboss.."
     su -l jboss -c  '/opt/jboss/bin/run.sh -b <Server-
IP> >> /var/log/jboss/log 2>> /var/log/jboss/errors &'
```

```
}
stop(){
     echo "stopping jboss.."
     su -l jboss -c '/opt/jboss/bin/shutdown.sh -S'
}

restart(){
     stop
     echo "please wait one minute ..."
     sleep 60
     start
}

case "$1" in
  start)
     start
     ;;
  stop)
     stop
     ;;
  restart)
     restart
     ;;
  *)
     echo "Usage: jboss {start|stop|restart}"
     exit 1
esac

exit 0
```

Das Installationsscript muss mit

```
chmod +x /etc/init.d/jboss
```

ausführbar gemacht werden und mit dem Befehl

```
update-rc.d jboss defaults
```

in den Systemstart eingebunden werden.

#### 4.4.5.3 JBoss Start

Für den Start des Applikationsserver ist das Kopieren der <Applikation>.ear und <Applikation>-ds.xml Datei in den Ordner /opt/jboss/server/default/deploy mittels File Transfer Protokoll (FTP) erforderlich.

Die Datei <Applikation>-ds.xml ist mit dem Benutznamen und dem Passwort der MySQL Datenbank zu befüllen.

Der Applikationsserver ist jetzt für den Start mit

```
/etc/init.d/jboss start
```

bereit.

Durch Überprüfung der Log-Datei /var/log/jboss/log kann festgestellt werden, ob der Applikationsserver erfolgreich gestartet wurde. Bei Auswahl der Serveradresse in einem Webbrowser wird die Startseite des JBoss Applikationsservers angezeigt. Die installierte Applikation ist über <Server URL>/<Applikationsname>:8080 aufrufbar.

### 4.4.6 Konfiguration JBoss

#### 4.4.6.1 JBoss Wurzelapplikation

Nach der bisher beschriebenen Installation des JBoss Applikationsservers ist dieser zwar funktionsfähig, es sind jedoch weitere Anpassungen erforderlich. Damit die installierte Applikation beim Aufruf der Server URL dem Benutzer angeboten wird, ist die Applikation als Wurzelanwendung dem JBoss Applikationsserver bekanntzugeben. Dazu ist im Enterprise Application Archive (EAR-Datei) der Applikation folgender Eintrag in der Datei application.xml hinzu zu fügen:

```
<module>
        <Web>
                <Web-uri>ironman.war</Web-uri>
                <context-root>/</context-root>
        </Web>
 </module>
```

Der Eintrag `<Web-uri>` enthält den Verweis auf das Web Application Archive der Web Applikation die auszuführen ist (WAR-Datei), der Eintrag `<context-root>`

enthält den Verweis auf den Ordner, der den Applikationsaufruf zur Verfügung stellt, in diesem Fall das Wurzelverzeichnis.

Es ist zu beachten, dass das WAR-Archiv in das EAR-Archiv der Applikation integriert wird.

#### 4.4.6.2 Server Port

Da JBoss aus Gründen der Systemsicherheit nicht unter dem „root“ Benutzer ausgeführt werden soll, ist es nicht möglich, dem JBoss Prozess das TCP-Port http (80) zuzuordnen, JBoss ist unter Port http-alt (8080) erreichbar. Um den JBoss Applikationsserver als Standard http Service einzurichten, ist eine Portweiterleitung an der Firewall oder am Server über die interne Firewall mit dem Befehl:

```
 -A PREROUTING -d 193.171.127.9/32 -p tcp -m tcp --dport 80 -
j DNAT --to-destination 193.171.127.9:8080
```

einzurichten.

### 4.4.7 Einrichten der Datenbank

Für die korrekte Einrichtung der Datenbank wurde in der Eclipse Entwicklungsumgebung bereits ein Ant Script erzeugt, das das Erstellen und Konfigurieren der Datenbank übernimmt. Nach dem Einrichten eines SSH-Tunnels vom Entwickler-PC zum Applikationsserver, werden die gesamten Datenbankeinstellungen durch Ausführen des Ant Script am Produktivsystem durchgeführt. Nach einem Neustart von JBoss ist das Produktivsystem funktionsfähig.

### 4.4.8 Einspielen von Testdaten

Mit dem IronNetwork Ticker Konfigurationstool können Veranstaltungsdaten in das Produktivsystem importiert werden. Nach dem Import der Athletendaten und der Einstellung von weiteren für die Veranstaltung erforderlichen Parametern, wie Rankz Server Adresse und Standorte der Zwischenzeitnehmung, ist das System bereit, einen Veranstaltungsablauf abzuarbeiten oder eine Veranstaltung simulieren zu können. Für die Simulation einer Veranstaltung wurde das Tool „IronNetwork TicketGenerator“ entwickelt, das in einstellbaren Intervallen Daten an das Verarbeitungssystem Rankz übermittelt und dem IronNetwork Ticker zur

Abfrage bereitstellt. Mit diesem Tool ist die Simulation einer Ironman Veranstaltung mit Zeitnehmungsdaten der Vorjahre durchführbar. Die Simulation kann im Zeitraffer abgearbeitet werden um eine höhere Last für die Verarbeitungssysteme zu erzeugen. Der Ablauf der Verarbeitung der Zeitnehmungsdaten ist in [LANG] beschrieben.

# 5 Praxis – Lasttest von Ajax Anwendungen

## 5.1 Lasttest Software Auswahl

Es existiert im Internet ein großes Angebot an Lasttest Programmen. Eine Übersicht an Open Source Tools ist unter [OPST] und [WSTT] zu finden. Bei der genaueren Recherche stellt sich jedoch heraus, dass nur ein Teil der angebotenen Produkte den Test von Ajax Anwendungen unterstützt. Neben den Herstellern von Lasttest Programmen gibt es Anbieter, die professionelle Testservices anbieten. Die Kosten für ein professionelles Lasttest Programm betragen für eine Anzahl von 500 virtuellen Usern mehrere Tausend Euro.

Bei der Recherche wurde der Fokus auf Open Source Produkte gelegt. Folgende Kriterien waren für eine Auswahl ausschlaggebend:

- Unterstützung von Ajax Anwendungen.
- Umfangreiche Dokumentation und Manuals für den Schnelleinstieg.
- Tools für die Aufzeichnung von Mustergeschäftsfällen.
- Einfache Einbindung von Parameter Dateien für die Steuerung der Geschäftsfälle.
- Intuitive Bedienung und geringe Einarbeitungszeit.

Nach einer Vorauswahl von mehreren Open Source und kommerziellen Produkten, die laut Produktbeschreibung den Lasttest von Ajax Anwendungen unterstützen, wurden die gewählten Produkte installiert.

Für die Auswahl galt die Zielsetzung mit dem gewählten Produkt in einer Stunde ein Test Szenario zu erstellen und einen Testablauf zu generieren.

Aus den zur Wahl stehenden Produkten wurde Web Performance Suite von Web Performance Inc. [WEPE], ein kommerzielles Produkt, gewählt.

### 5.1.1 Web Performance Suite

Die Web Performance Suite ist eine Java Anwendung mit einer Benutzeroberfläche im Stil von Eclipse. Mit Hilfe des Quick Start Guide ist in wenigen Minuten ein einfacher Test zusammengestellt und kann danach auch gestartet werden. Während dem laufenden Test werden online

Belastungsdiagramme der Messwerte dargestellt sowie die Speicherauslastung und CPU Belastung des Last erzeugenden Systems, wie in Abbildung 5, abgebildet.


Abbildung 5: Web Performance - Messung

Das Tool unterstützt die Steuerung von mehreren Lastquellen, deren Überwachung im unteren Bildschirmabschnitt möglich ist. Für den Einsatz von mehreren Lastquellen, stellt Web Performance ein ISO Image zum Erzeugen einer Start-CD zur Verfügung. Mit dem Einlegen der CD in einen PC wird dieser mit einem Red Hat Linux gestartet, die Lastsoftware initialisiert und der PC erscheint automatisch am steuernden PC auf. Die Start-CD ist ein Live System, das keine Änderungen am PC und der Festplatte durchführt. Wenn die Programmversion auf der zusätzlichen Lastquelle nicht aktuell ist, kann das Update in wenigen Sekunden in den Arbeitsspeicher erfolgen.

Für Windows- und Linux Server wird ein zusätzliches Programmmodul angeboten, das die CPU- und Speicherauslastung während des Lasttests am Server ermittelt

und online an das Testsystem zur Darstellung übermittelt. Mit Abschluss des Lasttests wird eine Protokolldatei mit der Serverauslastung dem Testprotokoll beigefügt.

Die einfache Bedienung und Konfiguration der Web Performance Suite hat zur Wahl dieses Testtools geführt. Da die Testversion nur zehn virtuelle Benutzer unterstützt, wurde auf Anfrage vom Hersteller für dieses Projekt eine Lizenz für 500 virtuelle Benutzer zur Verfügung gestellt.

### 5.1.2 Test Szenario

Der IronNetwork Live-Ticker besteht im Wesentlichen aus zwei Anwendungsfällen: der Anzeige von Athleteninformationen und der Anzeige von Top-Ten Listen.

Für diese Szenarien wurden zwei Testfälle zusammengestellt, die gleichzeitig vom Lasttest Tool ausgeführt werden.

Der Testlauf wird durch Simulation mit Zeitnehmungsdaten der Ironman 2007 Veranstaltung durchgeführt. Dazu ist die Datenbank des Applikationsservers mit den Teilnehmerdaten von 2007 zu laden und die Veranstaltung mit dem Konfigurationstool im Applikationsserver einzurichten. Durch Start der Rankz Applikation ist das System betriebsbereit. Mit dem Testtool „TicketGenerator“ können die Zeitnehmungsdaten der Veranstaltung an die Rankz Applikation übergeben werden und die Veranstaltung läuft als Simulation im Zeitraffer auf dem Applikationsserver ab.

#### 5.1.2.1 Szenario 1 – Athleten Anzeige

Für die Athletenanzeige wurde folgender Testablauf zusammengestellt:

- Aufruf der Startseite.
- Eingabe einer Startnummer als Auswahlkriterium.
- Auswahl der Register: Profil, Adresse, Google Maps im Intervall von 3 Sekunden.
- Wechsel zur Startseite.
- Wechsel zur Athletenansicht und Verbleib für 10 Minuten mit automatischer Aktualisierung.

Über die Testszenario-Konfiguration wird jedem Testfall eine Startnummer aus den möglichen Startnummern zugeteilt. Die Startnummern werden über eine

Abfrage in eine CSV Datei exportiert, diese CSV Datei kann in der Lasttestkonfiguration wieder eingelesen werden. Ein Tool erlaubt es die Feldeingabe durch eine Startnummer aus der Lastkonfiguration zu ersetzen.

#### 5.1.2.2 Szenario 2 – Top-Ten Liste

Für den Testablauf der Ergebnislisten wurden folgende Schritte festgelegt:

- Aufruf der Startseite.
- Auswahl der Gesamtergebnisliste durch Drücken der „Suchen“ Schaltfläche.
- Blättern auf Seite zwei und drei.
- Auswahl der Altersgruppe F30.
- Auswahl der Altersgruppe M30.
- Blättern auf Seite zwei und drei.
- Auswahl der Gesamtliste.

Für das Szenario 2 werden keine veränderbaren Parameter festgelegt.

### 5.1.3 Vorbereitung zum Lasttest

Die Testszenarien werden im Testtool Web Performance Suite aufgenommen und in einem Testlauf die Funktionsfähigkeit überprüft. Gültige Startnummern werden aus dem Applikationsserver exportiert und in das Testtool als Eingabeparameter importiert. Der gesamte Lasttest wird mit den Szenarios und den definierten Parametern zusammengestellt und gespeichert.

Für die Aufzeichnung der Speicher und CPU Belastung am Applikationsserver werden die erforderlichen Programme installiert. Nicht erforderliche Serverkomponenten werden gestoppt.

Mit einem letzten Testlauf werden die Einstellungen des Lasttests geprüft, um anschließend einen Testlauf durchzuführen.

### 5.1.4 Erkenntnisse aus der Vorbereitung

Die ersten Testläufe wurden auf den PCs für die Entwicklung des Projekts durchgeführt, was mit einem geringen Konfigurationsaufwand durchführbar und für die Überprüfung des Testablaufs ein gutes Hilfsmittel ist. Die CPU- und Speicherauslastung ist nicht mit einem Echtsystem vergleichbar, da die Lastquelle

und der Applikationsserver am selben System betrieben werden. Ab 100 gleichzeitigen Benutzern ist das CPU- und Speicherlimit erreicht.

#### 5.1.4.1 Web Seiten Größe

Das Tool zur Aufzeichnung der Benutzeraktivität in der Web Performance Suite liefert einen detaillierten Bericht über Dateigröße und Antwortzeit aller übertragenen Seitenbestandteile. Dabei hat sich gezeigt, dass die zu übertragenden Javascript Bibliotheken eine Größe von 1,07 MB haben, wie die Aufstellung in Abbildung 16 zeigt (main.js, mapstraction.js, yui-utilities.js, ext-yui-adapter.js, ext-all.js, bigdecimals.js, mathcontext.js). Bei einem Internetzugang mit 56 kbit/s Modem oder ISDN Anschluss führten diese großen Dateien zu einer Ladezeit von ca. 200 Sekunden. Diese lange Übertragungszeit kann für viele Benutzer ein Grund für das vorzeitige Beenden des IronNetwork Ticker sein.

#### 5.1.4.2 Test am Produktivsystem

Ein erster Test auf dem Produktivsystem hat bei ca. 50 gleichzeitigen Zugriffen zu 100% CPU-Auslastung aufgrund des Heap Überlaufs der Java Virtual Machine geführt. Da sich dieser Zustand nach sechs Stunden nicht verbessert hat, war ein Neustart des Betriebssystems erforderlich. Vom IT Service der FH wurde ein Betriebssystem Patch eingespielt und Parameter am Betriebssystem verändert. Zusätzlich wurde zwischen Internet und JBoss ein Apache Web Server, wie in Abbildung 4 dargestellt, integriert. Die Stabilität hat sich durch diese Maßnahme verbessert, jedoch nicht soweit, dass ein erfolgreicher Test abgeschlossen werden kann. Daher wurde der erste Durchgang des Lasttests auf einem Entwicklungs-PC ausgeführt.

## 5.2 Lasttest – Durchgang 1

Schwierigkeiten mit der Stabilität des Produktivsystems führten zur Entscheidung, einen Lasttest auf einem Entwicklungssystem auszuführen. Dazu wurde die in Abbildung 6 dargestellte Konfiguration gewählt. Wenn der Applikationsserver und die Lasttest Software am selben System ausgeführt werden, steigt die CPU-Auslastung sehr stark an, wodurch das Messergebnis beeinflusst wird. Mit Einsatz einer zusätzlichen Lastquelle, die von der Lastquelle gesteuert wird, wird die Belastung auf die zusätzliche Lastquelle ausgelagert und eine 30%-ige CPU

Belastung während des gesamten Tests nicht überschritten. Zum Einsatz kamen folgende PC Systeme:

### Server und Laststeuerung

**Hardware**

Intel P4 E6600, 2GB RAM, 300 GB Festplattenspeicher, Netzwerk 100Mbit/s

**Software**

Windows XP Professional, Java 5, JBoss 4.2.2 GA, MySQL 5, Web Performance Suite 3.5

### Zusätzliche Lastquelle

**Hardware**

Intel P4 2,5 GHz, 1GB RAM, 160 GB Festplattenspeicher, Netzwerk 100Mbit/s

**Software**

Web Performance Load Engine 3.5 – Linux Boot CD

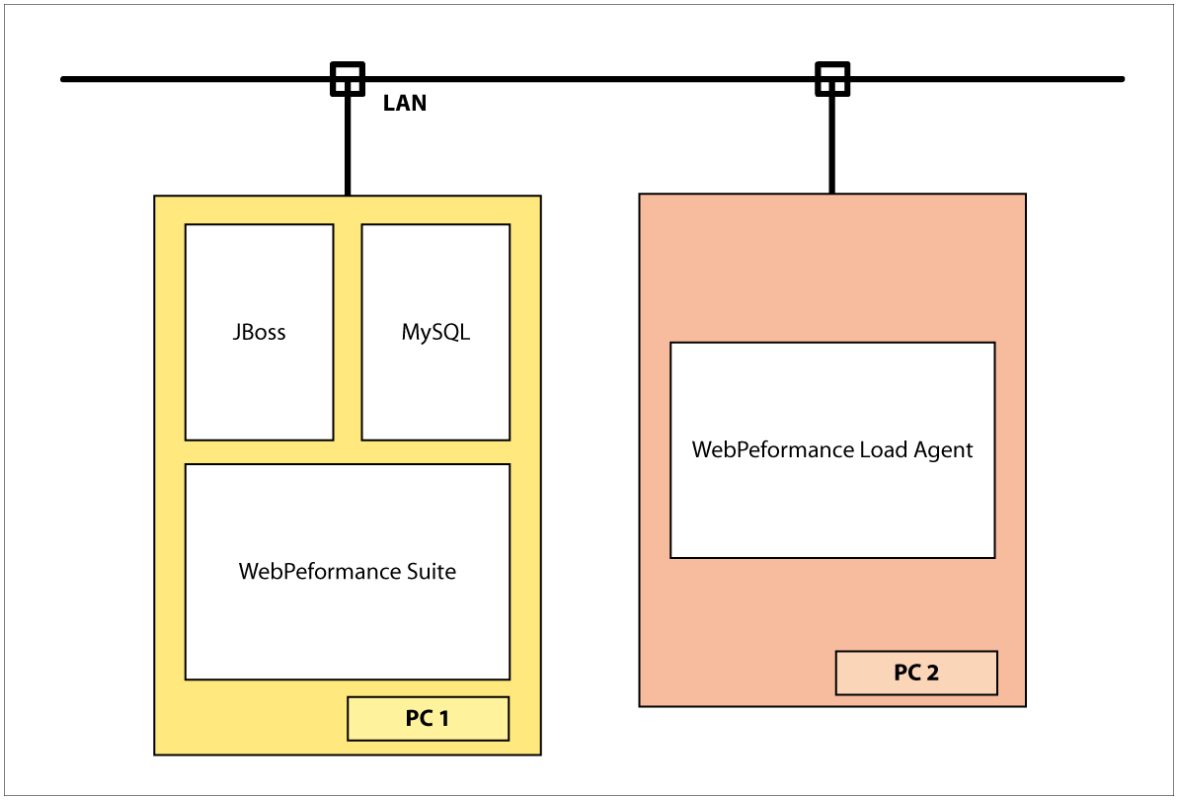



Abbildung 6: Lasttest Konfiguration – Durchgang 1

### 5.2.1 Messergebnisse

Der Test wurde mit folgenden Einstellungen durchgeführt:

- Anzahl der Benutzer beim Start: 50.
- Laststeigerung: 20 Benutzer pro Minute.
- Testdauer: 20 Minuten.
- Maximale Benutzeranzahl: 419.
- Simulierte Bandbreite der Benutzer: maximal 5 Mbit/s.

Die Abbildung 7 zeigt die Ladezeiten. Beachtenswert ist das Verhalten, dass mit zunehmender Benutzeranzahl die Ladezeit nicht ansteigt.

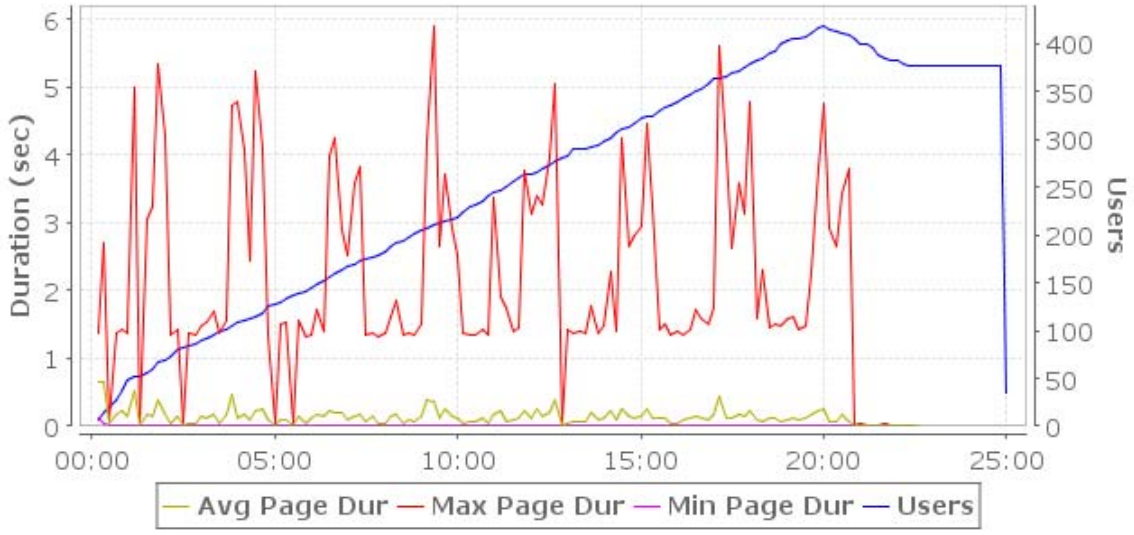



Abbildung 7: Lasttest 1 – Seitenladezeiten

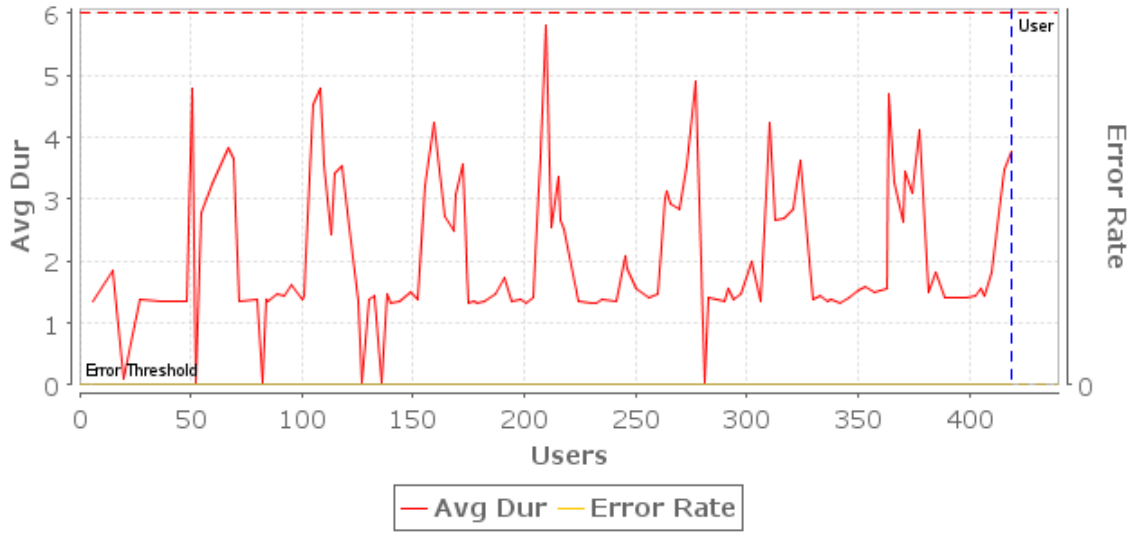



Abbildung 8: Lasttest 1 – Benutzerkapazität

Dieses Ergebnis lässt sich auch aus Abbildung 8 ablesen, die die Ladezeit über der Anzahl der virtuellen Benutzer darstellt. Dieses System zeigt keine Beschränkungen hinsichtlich der 419 Benutzer, die gleichzeitig zugreifen.

### 5.2.2 Erkenntnisse und Maßnahmen

Der lokale Test bei dem der Applikationsserver und das Lasttestprogramm am selben PC betrieben werden zeigt, dass die Konfiguration in der Lage ist, die geforderte Last zu verarbeiten. Der wichtige Bereich des Netzwerktransfers ist in diesem Testfall nicht berücksichtigt.

Um eine Gegenprobe zu machen, wurde ein weiterer Test am Entwicklungssystem mit der Lasttestsoftware auf einem eigenem PC durchgeführt um das Netzwerk in den Test mit einzubeziehen.

Mit diesen Erkenntnissen wurde eine Reihe von Tests am Produktivsystem durchgeführt und durch Anpassung von Betriebssystemparametern eine Verbesserung erzielt. Ab der Belastung von circa 150 virtuellen Benutzern war der Linux Server überlastet und nur mit einem Neustart wieder in betriebsbereiten Zustand zu bringen.

## 5.3 Lasttest – Durchgang 2

Für eine Gegenprüfung wurde ein weiterer Test in gleicher Konfiguration wie am Produktivsystem mit dem Entwicklungssystem durchgeführt. Der Aufbau ist in

Abbildung 6 dargestellt. Dazu wurde der Entwicklungs-PC mit MySQL Server und JBoss eingerichtet. Auf einem Notebook-PC wurde die Web Performance Suite installiert und zusätzlich ein PC als zusätzliche Lastquelle eingerichtet. Zur Aufzeichnung der Serverbelastung wurde am PC, der als Server fungiert, von Web Performance ein Server Agent installiert. Zum Einsatz kamen folgende PC Systeme:

**Server**

**Hardware**

Intel P4 E6600, 2GB RAM, 300 GB Festplattenspeicher, Netzwerk 100 Mbit/s

**Software**

Windows XP Professional, Java 5, JBoss 4.2.2 GA, MySQL 5, Web Performance Server Agent 3.5

**Laststeuerung**

**Hardware**

Notebook PC, Intel Centrino 1,7 GHz, 1 GB RAM, 80 GB Festplatenspeicher, Netzwerk 100 MBit/s

**Software**

Windows XP Professional, Web Performance Suite 3.5

**Zusätzliche Lastquelle**

**Hardware**

Intel P4 2,5 GHz, 1GB RAM, 160 GB Festplattenspeicher, Netzwerk 100Mbit/s

**Software**

Web Performance 3.5 Load Engine – Linux Boot CD

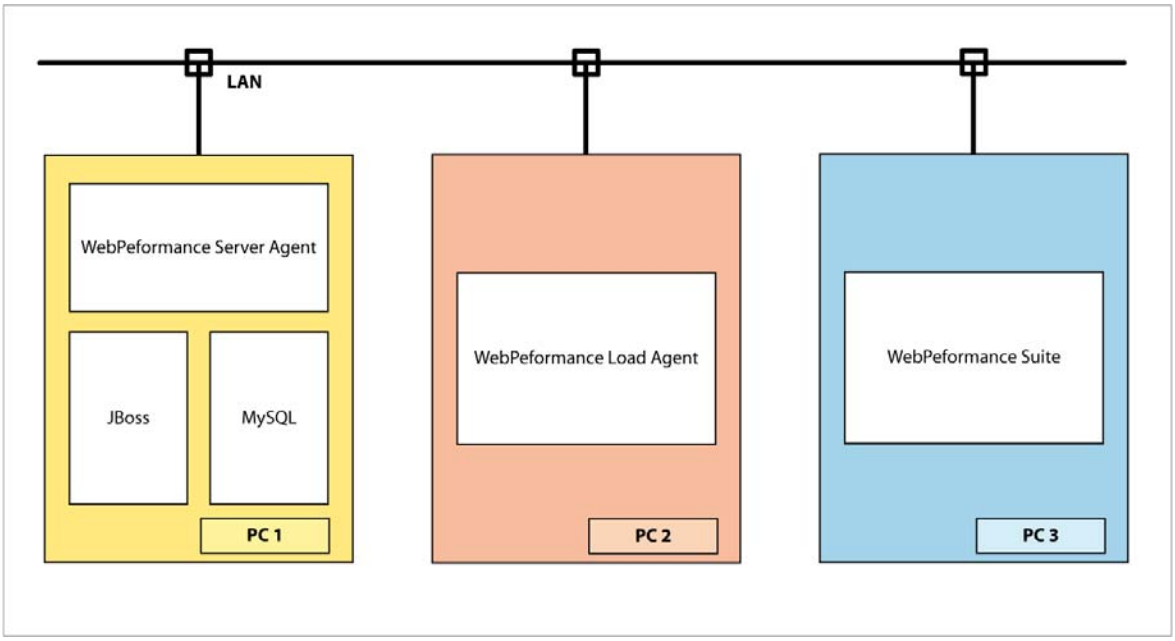



Abbildung 9: Lasttest Konfiguration – Durchgang 2

### 5.3.1 Messergebnisse

Der Test wurde mit folgenden Einstellungen durchgeführt:

- Anzahl der Benutzer beim Start: 20.
- Laststeigerung: 25 Benutzer in zwei Minuten.
- Testdauer: 20 Minuten.
- Maximale Benutzeranzahl: 195.
- Simulierte Bandbreite der Benutzer: maximal 5 Mbit/s.

Die Abbildung 10 zeigt die Seitenladezeiten. Die Ladezeit nimmt mit zunehmender Belastung zu.

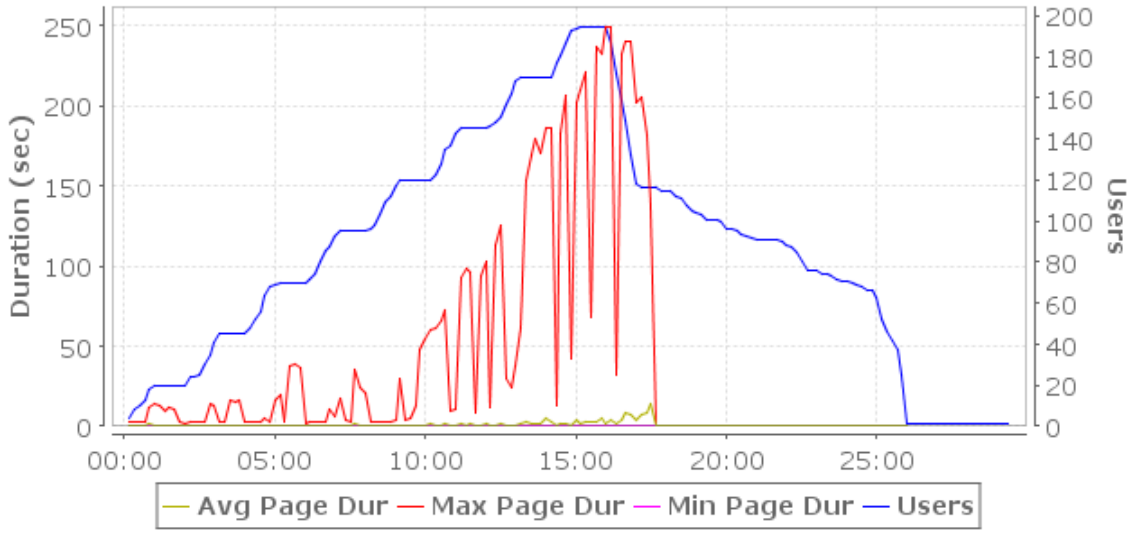



Abbildung 10: Lasttest 2 – Seitenladezeiten

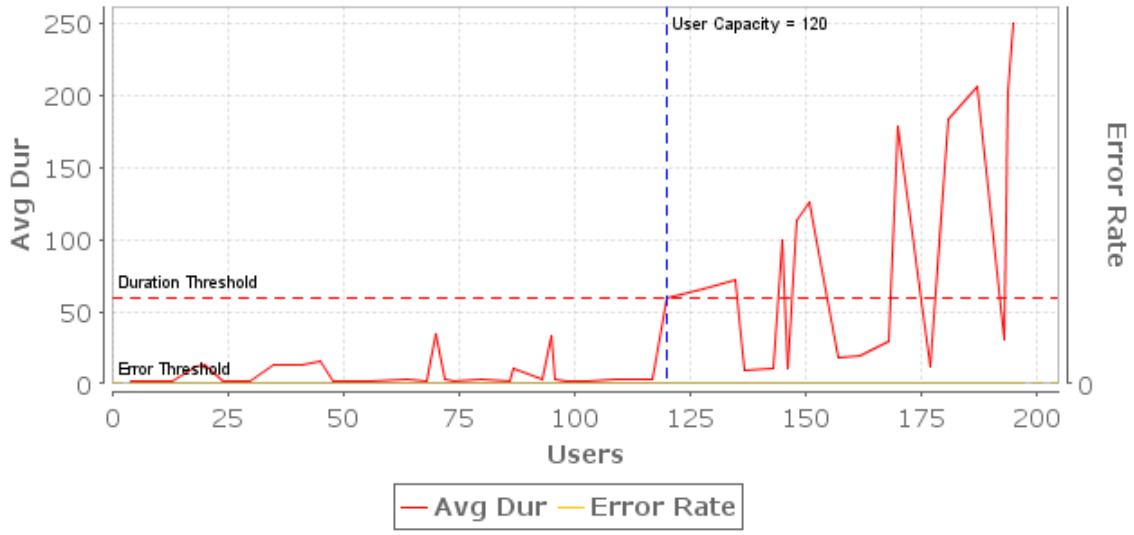



Abbildung 11: Lasttest 2 – Benutzerkapazität

Die Belastungsgrenze des Systems liegt bei einer maximalen Ladezeit von 60 Sekunden bei 120 Benutzern. Abbildung 11 zeigt bis zu einer Belastung von 120 Benutzern eine geringe Ladezeit, darüber kommt es zu einem starken Anstieg der Ladezeit. Die CPU-Auslastung zeigt in Abbildung 12 einen Anstieg, der proportional zur Anzahl der Benutzer ist, und bei 200 Benutzern das Limit erreicht hat. Die Speicherauslastung des Systems bleibt jedoch konstant.

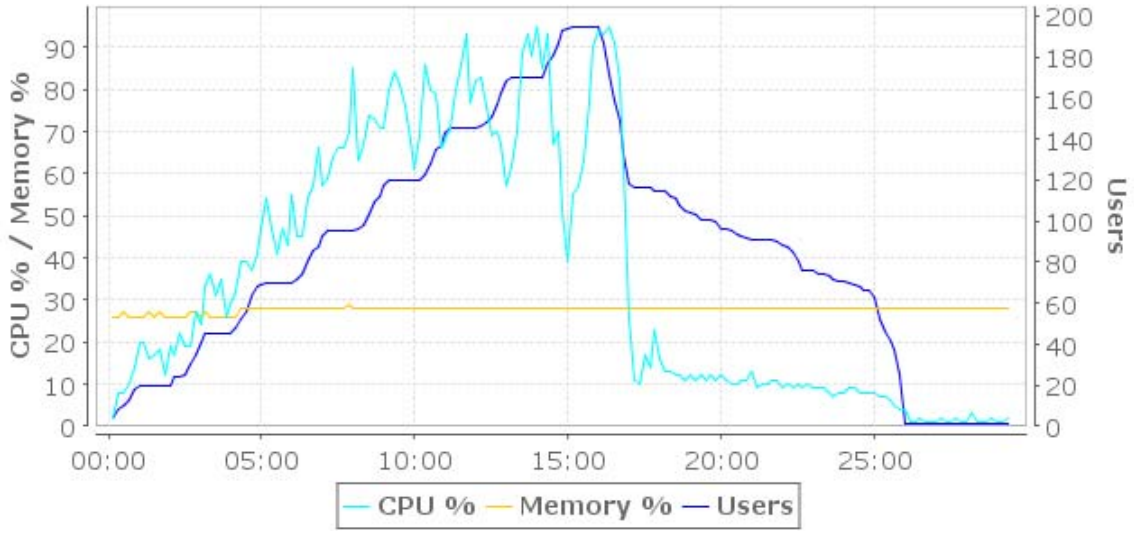



Abbildung 12: Lasttest 2 – CPU- und Speicherauslastung

### 5.3.2 Erkenntnisse und Maßnahmen

Der Test am Entwicklungssystem hat bei gleicher Testkonfiguration wie beim Produktivsystem gezeigt, dass sich die Windows Konfiguration bei Belastung anders verhält. Der Linux Server läuft bis zu einer Belastung von 250 virtuellen Benutzern mit einer geringen CPU Gesamtauslastung von unter 10%. Darüber kommt es zu einem sprunghaften Anstieg, der zu einem Ausfall des Systems führt. Dieser Betriebszustand bleibt, auch nach Beenden des Tests unverändert, und lässt sich nur durch einen Neustart des Servers beseitigen.

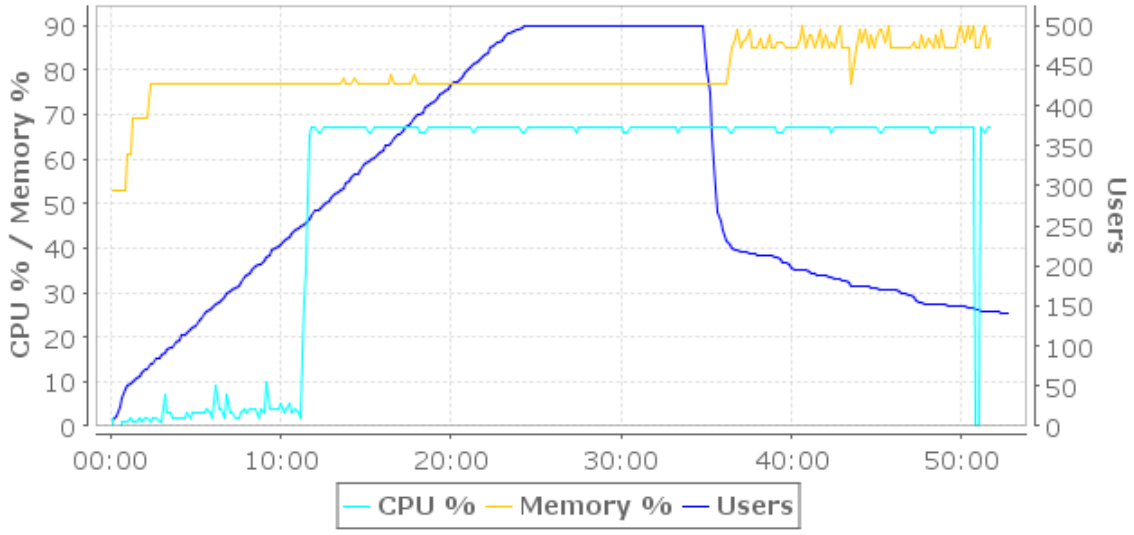



Abbildung 13: Lasttest am Produktivsystem - CPU Auslastung

Die abgebildete CPU Auslastung zeigt das beste Ergebnis einer langen Reihe von Tests, die mit Serverabstürzen bei 25 Benutzern in den ersten Minuten des Tests begonnen haben.

Der Linux Server wird unter der Servervirtualisierung OpenVZ [OPVZ] betrieben, wobei nur eine Instanz für den Applikationsserver eingerichtet ist. Beeinflussungen durch andere Serverinstanzen sind nicht möglich. Der nächste Schritt ist die Installation eines reinen Linux Servers ohne Virtualisierungsschicht. Dieser wird in der gleichen Konfiguration getestet.

Während dem laufenden Test wurde auf einem eigenen PC regelmäßig versucht den Ticker in einem Browser zu starten, um die Messergebnisse zu verifizieren. Subjektiv wird bei starker Belastung eine Ladezeit über einer Minute als Systemausfall wahrgenommen. Bei geladener Applikation erfolgt die Athletenauswahl und die Anzeige der Ranking Listen auch bei starker Belastung äußerst rasch und lässt die hohe Rechnerauslastung nicht erahnen. Dies wurde bei den Tests am Entwicklungs-PC mit Windows XP Betriebssystem festgestellt.

Am Linux System entsteht bei den Tests schon bei einer geringen Anzahl von virtuellen Benutzern eine lange Ladezeit beim Start der WEB Applikation. Dieses Verhalten spiegelt sich auch in Abbildung 14 in der hohen Fehlerrate wider, die kurz nach dem Start des Tests ab circa 20 Benutzern anzusteigen beginnt.


Abbildung 14: Lasttest Linux Server – Benutzerkapazitäten

## 5.4 Optimierungsmaßnahmen

Um den IronNetwork Live-Ticker für mehrere hundert Benutzer einsetzbar zu machen, sind die Stabilitätsprobleme am Linux Server zu beheben und mit tiefer gehenden Analysen die Ursachen für den sprunghaften Anstieg der CPU Auslastung zu ermitteln. Ein weiterer Faktor ist Betrachtung der zu transportierenden Inhalte der Web Applikation.

### 5.4.1 Analyse der Web Seite

In Abbildung 15 ist eine Aufstellung der im Testszenario 1 geladenen Seiten zu sehen. Die Startseite `<webpage> [1]` hat eine Größe von 1,2 MB und eine Ladezeit von circa 40 Sekunden. Die Seite `<webpage> [6]` enthält die Google Maps Darstellung mit 640 kB.

Der Start der Web Applikation verursacht das größte Datenvolumen und zeigt auch beim Lasttest einen großen Bandbreitenbedarf.

Testcase Report
Ticker Szenario 1 Local

webperformance

Page Metrics

The pages section contains metrics for each web page in the testcase.

| Title | Duration (MM:SS:mmm) | Size | Status | # URLs | Page Size | Think tin (MM:SS:mmr |
|---|---|---|---|---|---|---|
| <webpage> [1] | 00:40.554 | 1,2 MB | 200 | 17 | 3,3 KB | 00:00.2 |
| 92EC993D2CC5B78F52A06866ADD916BC.cache.html | 00:07.116 | 217,9 KB | 200 | 1 | 217,9 KB | 00:00.3 |
| <webpage> [2] | 00:05.802 | 112,9 KB | 200 | 19 | 2,6 KB | 00:06.8 |
| <webpage> [3] | 00:01.013 | 11,8 KB | 200 | 9 | 2,5 KB | 00:04.8 |
| <webpage> [4] | 00:00.034 | 1,7 KB | 200 | 1 | 1,7 KB | 00:04.9 |
| <webpage> [5] | 00:00.029 | 1,7 KB | 200 | 1 | 1,7 KB | 00:11.0 |
| <webpage> [6] | 00:05.400 | 639,7 KB | 200 | 39 | 1,7 KB | 00:04.3 |
| <webpage> [7] | 00:00.026 | 1,7 KB | 200 | 1 | 1,7 KB | 00:08.8 |
| <webpage> [8] | 00:00.026 | 1,7 KB | 200 | 1 | 1,7 KB | 00:04.9 |
| <webpage> [9] | 00:00.026 | 1,7 KB | 200 | 1 | 1,7 KB | 00:04.9 |

Abbildung 15: Test Szenario 1 - Ladezeiten

### 5.4.2 Belastung durch Ajax

Der große Bandbreitenbedarf beim Start der Web Applikation führt zu einem großen Ressourcenbedarf im Netzwerk und am Server, hingegen bringt der

Einsatz von Ajax für das Aktuellhalten der Informationen beim Client einen großen Vorteil: es werden je Session alle 5 Sekunden Pakete mit 1,7 kB <webpage> [4], [5], [7], [8], [9] übertragen. Die Aktualisierungspakete enthalten die gesamten Zwischenzeitdaten des gewählten Athleten.

Wenn bei 1000 Benutzern die Athletendaten aktualisiert werden, erzeugt das eine durchschnittliche Netzwerklast von 340 kBit/s.

Eine gleichwertige herkömmliche Web Seite würde bei regelmäßigem Drücken der „Aktualisieren“ Schaltfläche im 30 Sekunden Takt bei einer Seitengröße von 60 kB, 2 Mbit/s erfordern. Der Komfort für den Benutzer ist bei Einsatz von Ajax ungleich größer.

### 5.4.3 Seitenoptimierung

Die Analyse der Startseite in Abbildung 16 zeigt, dass der Großteil der Datenmenge beim Laden der Seite von den eingebundenen Java Script Bibliotheken stammt. Durch den Einsatz von Komprimierungstools können diese verringert werden. Weitere Optimierungsschritte können über das Entfernen von nicht benötigten Java Script Teilen aus den Bibliotheken erreicht werden.

Die eingesetzte GWT-EXT Bibliothek [GWTE] wird vom GWT Compiler nicht optimiert. Daher kann die Dateigröße durch Entfernen nicht erforderlicher Routinen verringert werden.

Die Verringerung der Größe der verwendeten Grafiken ist als Standardaktivität bei der Web Seitengestaltung anzusehen.

# Testcase Report
# Ticker Szenario 1 Local

## URLs for page: [1]

| Title | Duration (MM:SS:mmm) | Total Size | Status | Request Size | Response Size | Request Duration | Re Du |
|---|---|---|---|---|---|---|---|
| <webpage> [1] | 00:00.007 | 3,3 KB | 200 | 483 B | 2,9 KB | 00:00.004 | 00: |
| <text> [1] | 00:00.166 | 9,8 KB | 200 | 359 B | 9,4 KB | 00:00.000 | 00: |
| main.js | 00:00.875 | 206,1 KB | 200 | 291 B | 205,8 KB | 00:00.000 | 00: |
| transparent.png | 00:00.060 | 622 B | 200 | 285 B | 337 B | 00:00.000 | 00: |
| mapstraction.js | 00:03.658 | 67,0 KB | 200 | 272 B | 66,8 KB | 00:00.000 | 00: |
| ext-all.css | 00:04.108 | 76,7 KB | 200 | 285 B | 76,4 KB | 00:00.000 | 00: |
| xtheme-silverCherry.css | 00:04.308 | 110,6 KB | 200 | 297 B | 110,3 KB | 00:00.000 | 00: |
| Ironman.css | 00:00.004 | 1,3 KB | 200 | 267 B | 1,1 KB | 00:00.000 | 00: |
| large-loading.gif | 00:00.007 | 3,8 KB | 200 | 309 B | 3,5 KB | 00:00.000 | 00: |
| background.gif | 00:00.003 | 638 B | 200 | 271 B | 367 B | 00:00.000 | 00: |
| logo.gif | 00:00.002 | 732 B | 200 | 265 B | 467 B | 00:00.000 | 00: |
| yui-utilities.js | 00:02.693 | 82,5 KB | 200 | 290 B | 82,2 KB | 00:00.000 | 00: |
| 00:02.704 | 24,2 KB | 200 | 267 B | 23,9 KB | 00:00.000 | 00: | mathcontext.js |
| 00:01.864 | 6,9 KB | 200 | 291 B | 6,6 KB | 00:00.000 | 00: | com.ironman.ticker.gwt.Ironman.nocache.js |
| 00:05.213 | 167,7 KB | 200 | 266 B | 167,4 KB | 00:00.000 | 00: | bigdecimal.js |

Abbildung 16: Test Szenario 1 - Header Daten

# 6 Abschluss und erreichte Ziele

Dieses Kapitel gibt einen Überblick über die Erkenntnisse, die bei dieser Arbeit erlangt wurden.

## 6.1 Ergebnisse

Die Implementierung des Web-Informationssystems IronNetwork Live-Ticker auf einem Produktivsystem mit Lasttests hat die Richtigkeit des Ansatzes bestätigt, dieses Projekt als Ajax Anwendung umzusetzen.

Die Vorteile und Probleme von Ajax liegen knapp beieinander. Der Vorteil der Ajax Anwendung für den IronNetwork Live-Ticker liegt in der geringen Datenmenge, die für das ständige Aktualisieren der Athleteninformationen erforderlich ist. Diese Funktion bietet dem Benutzer große Vorteile gegenüber einer herkömmlichen Webanwendung. Gleichzeitig dauert das Laden der Java Script Bibliotheken beim Aufruf der Web Anwendung bei geringer Bandbreite zum Endgerät so lange, dass der Benutzer in vielen Fällen in der Annahme, das System sei ausgefallen, abbricht.

Optimierungspotenzial beim Verkleinern der Java Script Bibliotheken ist vorhanden, was jedoch der Grundintention, GWT für die Entwicklung von Web Applikationen mit Java Script zu verwenden, widerspricht. Dies gilt insbesondere, wenn zusätzliche Bibliotheken wie GWT-EXT in diesem Projekt eingesetzt werden.

Der Lasttest am Entwicklungssystem hat gezeigt, dass eine intensive Nutzung des Ironman Live-Ticker mit 200 gleichzeitig zugreifenden Benutzern ohne Fehler verarbeitet werden kann. Dass die Lasttests am Produktivsystem gescheitert sind ist mit hoher Wahrscheinlichkeit auf nicht optimale Einstellungen am Linux-Server oder an der Servervirtualisierung OpenVZ zurückzuführen. Recherchierte Einstellungsvorschläge haben Verbesserungen bewirkt, jedoch noch keinen Durchbruch gebracht.

## 6.2 Erreichte Ziele

Mit der Einführung des Model-View-Controller Modells wurde die Weiterentwicklung und Verbesserung des Benutzerinterface erleichtert und die Basis für Weiterentwicklungen geschaffen. Veränderungen an der Bedienerführung und damit der Usability führten zu einem einfacheren und aufgeräumten Bedienkonzept und einfacheren Dialogen. Suchfelder für unterschiedliche Auswahlkategorien wurden auf ein universelles Auswahlfeld zusammengelegt, die Athletenauswahl in die Rankingliste integriert.

Die Installation des Produktivsystems im FH Netz konnte durchgeführt und abgeschlossen werden um die Voraussetzungen für Lasttests zu schaffen. Die Lasttests haben gezeigt, dass der gewählte Ansatz für Realisierung richtig gewählt wurde.

Nach Anpassung von Systemeinstellungen am Linux Server, kann das Produktivsystem für einen Feldtest bei der nächsten Ironman Veranstaltung eingesetzt werden.

# 7 Projektplan – wirtschaftliche Betrachtung

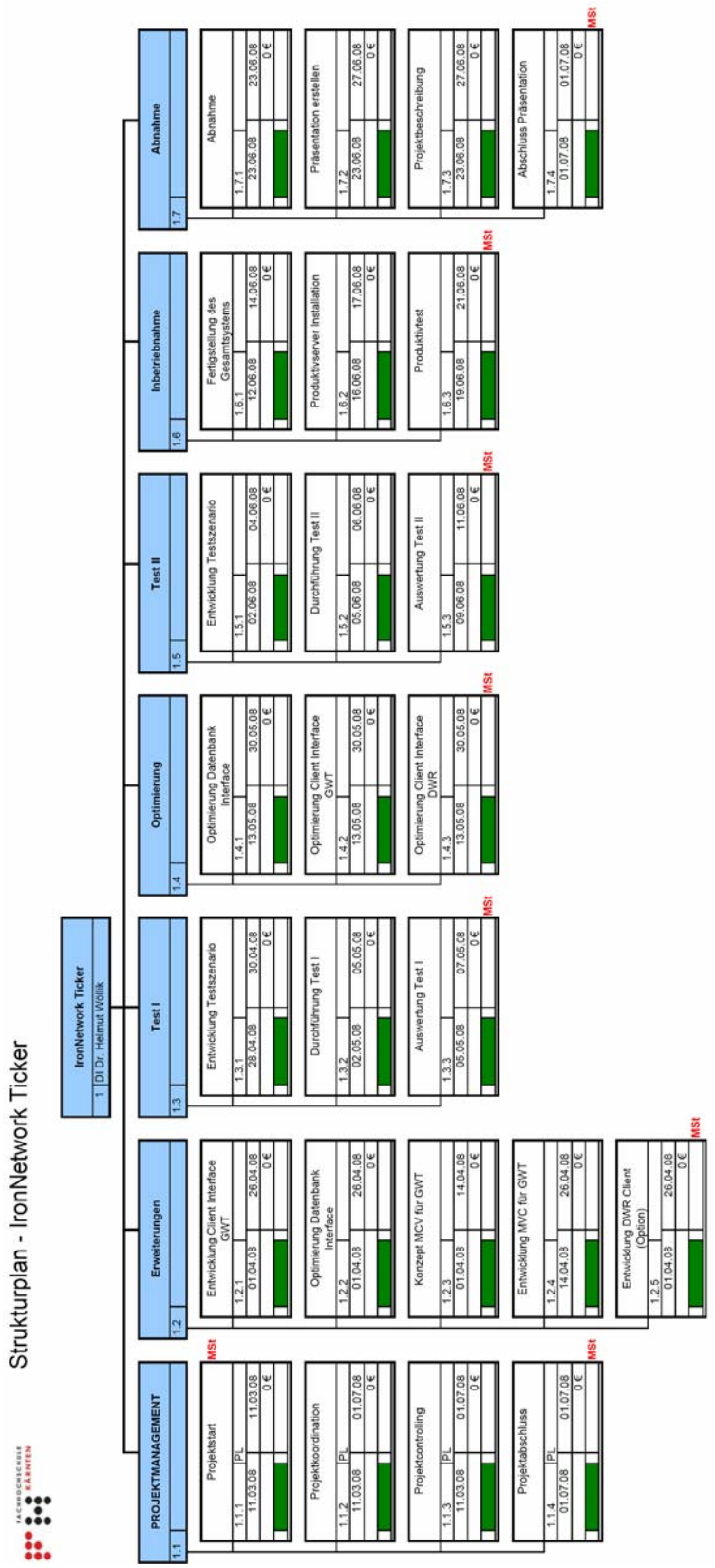
## 7.1 Projektmanagement

Für die Projektumsetzung wurden zum Projektstart die Projektstruktur mit den einzelnen Projektphasen und Arbeitspaketen definiert. Nachdem Umfang und Zeit der Arbeitspakete grob abgeschätzt waren, wurde ein grober Zeitplan festgelegt.

Diese Vorgangsweise erlaubt auch kleinen Projektteams rechtzeitig Entscheidungen zu treffen, die für die Einhaltung des vorgegebenen Rahmens erforderlich sind.

Die Kommunikation im Projekt erfolgte beinahe täglich am Telefon oder mit Internet Chat. Die intensive Kommunikation erforderte mehr zeitlichen Aufwand, brachte jedoch Vorteile in der Entwicklung, da Entscheidungen in Detailfragen schnell getroffen werden konnten und diese Teamentscheidungen die Zusammenarbeit verbesserten.

### 7.1.1 Projektstrukturplan



# 8 Anregungen für weiterführende Projekte

Entscheidungen für einen Lösungsweg für ein Projekt werden beim Auftreten von Problemen immer wieder hinterfragt und wenn größere Schwierigkeiten auftreten wird der gewählte Lösungsweg in Frage gestellt. Bei der Entwicklung des IronNetwork Live-Tickers hat es diese Phasen ebenfalls gegeben. Insbesondere bei der Entwicklung der Webschnittstelle mit GWT, die auf herkömmliche Weise mit einem HTML Editor erstellt werden würde und die erforderlichen Serveraufrufe in die Seite integriert werden würden, wurde das Frustpotenzial zeitweise sehr hoch. Einfache Elemente, die in HTML mit einer Auszeichnung einfach integriert und formatiert werden, können in GWT Stunden für Recherche kosten und eine lange Kette von erfolgslosen Versuchen nach sich ziehen.

Trotzdem war der eingeschlagene Weg, Ajax, GWT und JBoss einzusetzen, der Richtige. Die Entwicklung der Web Schnittstelle ist mit GWT vollkommen in die Applikation integriert. Schnittstellenprobleme zwischen HTML Seiten und Serverapplikationen werden durch diese Technologie weitgehend ausgeschaltet.

Die Auswahl eines Lasttest Systems und die Durchführung von Lasttests für Ajax Web Applikationen hat die Problematik des automatischen Testens von dynamischen Web Applikationen gezeigt. Die Tests haben einerseits die Richtigkeit des Lösungsansatzes bewiesen, andererseits gezeigt, dass Betriebssystemanpassungen und Optimierungen einen bedeutenden Teil des Zeitaufwands in der Integrationsphase ausmachen können und eine Portierung eines unter Windows lauffähigen Systems auf Probleme bei der Ausführung auf Linux bringen kann.

Der IronNetwork Live-Ticker ist für den Produktiveinsatz gerüstet. Mit wenigen Optimierungen und Adaptierungen ist das System soweit vorbereitet, dass es um den Anforderungen einer Ironman Veranstaltung gewachsen ist und den Benutzern die gewünschten Informationen komfortabel liefern kann.

Für die Weiterentwicklung des IronNetwork Live-Tickers sind folgende Punkte bei der Projektumsetzung als nächste Schritte für die Umsetzung als sinnvoll angesehen worden:

- Verringerung der Größe der zu übertragenen Java Script Bibliotheken.

- Endgeräte abhängige Seitengestaltung für Mobiltelefone und Smartphones.
- Durchführen von Usability Tests für die weitere Verbesserung der Bedienbarkeit.
- Optimierung der Servereinstellungen um auch bei hoher Last die Stabilität des Systems zu erreichen.

.

# Glossar

| | |
|---|---|
| AJAX | Asynchronous JavaScript and XML. Überbegriff für Technologien, die es erlauben Web – Anwendungen zu entwickeln, die sich wie Desktop Anwendungen verhalten. |
| Ant | Open Source Programm zum automatisierten Erzeugen von Programmen aus Quelltext. |
| JBoss | Open Source Java Applikations-Server mit integriertem Webserver. |
| RANKZ | Bezeichnung einer Applikation des IronNetwork, die permanent aus Zeitnehmungsdaten aktuelle Ergebnislisten erstellt. |
| Smartphone | Kombinationsgerät das Mobiltelefon und PDA (Personal Digital Assistant) in einem Gerät vereint und das damit die Möglichkeit bietet, Anwendungen wie Webbrowser oder E-Mail Clients komfortabel zu nutzen. |
| Usability | Begriff für die Benutzerfreundlichkeit von Software Produkten. |

# Literaturverzeichnis


[AJIP] Dave Crane: *Ajax in practice – Das Praxisbuch für Web 2.0-Entwicklung* – Addison-Wesley 2008 – ISBN 978-3-8273-2596-9

[LANG] Lang Johannes: *AJAX Technologie für selbstaktualisierende Webinformationssysteme am Beispiel IronNetwork Live-Ticker für Sportveranstaltungen* – Bachelorarbeit FH-Kärnten, 7. März 2008

[NIEL] Jakob Nielsen: *Designing Web Usability*. Markt & Technik 2001 – ISBN 3-8272-6206-2, http://www.useit.com/

[SCHN] Schneider Manuel: *HIBERNATE – Objektorientierte Zugriffsschicht im Applicationserver am Beispiel „IronNetwork Live-Ticker“ für Sportveranstaltungen* – Bachelorarbeit FH-Kärnten, 7. März 2008

[SCHN2] Schneider Manuel: *Model-View-Controller & Google Web Toolkit am Beispiel „IronNetwork Live-Ticker“ für Sportveranstaltungen* – Bachelorarbeit FH-Kärnten, 27. Juni 2008

[SIEG] David Siegel: *Web Site Design.* Markt & Technik 1998 – ISBN 3-8272-5331


## Internet Referenzen

Zugriffdatum: 27.6.2008


[APT] http://debiananwenderhandbuch.de/apt-get.HTML

[ECL] http://www.eclipse.org/

[GWTE] http://code.google.com/p/gwt-ext/

[JBIN] http://gateway.sourceforge.net/server-setup/jboss-setup.HTML

[JBSRC] http://www.jboss.org/jbossas/downloads/

[MYSQL] http://www.mysql.com/

[OPST] http://www.opensourcetesting.org/performance.php

[OPVZ] http://openvz.org/

[PODE] http://www.poderosa.org/

[PUTTY] http://www.chiark.greenend.org.uk/~sgtatham/putty/

[STOL] http://www.performance-test.de/

[WEPE] http://www.Webperformanceinc.com/

[WSTT] http://www.softwareqatest.com/qatweb1.HTML

# 9 Anhang

## 9.1 SSH Tunnel Einstellung mit Putty

Das Terminal Programm Putty stellt SSH Tunnel zur Verfügung. Die Einrichtung erfolgt im Konfigurationsmenü unter dem in Abbildung 17 abgebildeten Dialog. Zum Hinzufügen eines Tunnels sind die Parameter „Source port“ und „Destination“ einzugeben und durch Drücken der „Add“ Schaltfläche hinzuzufügen. Bei aktiver SSH Verbindung wird der Tunnel automatisch aufgebaut.

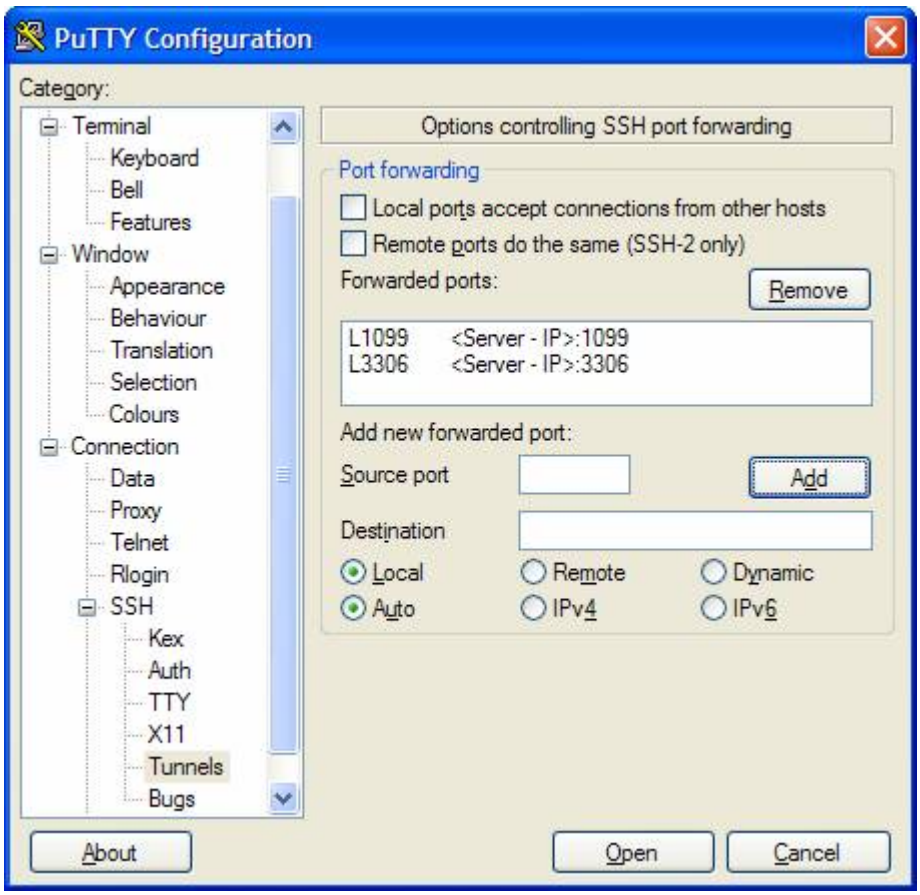



Abbildung 17: Putty - Tunnel Dialog