Hochschule
Augsburg University of
Applied Sciences

Bachelorarbeit

Fakultät für
Informatik

Studienrichtung
Informatik

Marc Rochow

Einsatz von Websockets zur Echtzeit-Administration eines Hosting Clusters

Prüfer:
Prof. Dr. Nik Klever

Abgabe der Arbeit am: 20.09.2012

Aufgabenstellende Firma: Skylime GbR

Hochschule für angewandte
Wissenschaften Augsburg
University of Applied
Sciences

An der Hochschule 1
D-86161 Augsburg

info@hs-augsburg.de

Fakultät für Informatik
Telefon: +49 821 5586-3450
Fax: +49 821 5586-3499
Verfasser der Bachelorarbeit:
Marc Rochow
Seilerstr. 1
D-86153 Augsburg
Telefon: +49 821 455 62 93

Eidesstattliche Erklärung

Marc Rochow

Seilerstr. 1

D-86153 Augsburg

Hiermit erkläre ich, Marc Rochow, geboren am 09.11.1986 in Lindau, dass ich die vorliegende Bachelorthesis mit dem Titel „Einsatz von Websockets zur Echtzeit-Administration eines Hosting Clusters“ selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Augsburg, den 31. August 2012

Abstrakt

to be done...

Vorwort

Die vorliegende Bachelorarbeit wurde in der Zeit vom 16. Mai bis 20. September erstellt. Sie ist Bestandteil des Bachelor-Studiengangs Informatik an der Hochschule Augsburg und entstand in Kooperation mit der Firma Skylime GbR in Tettnang.

Mit der Arbeit wird versucht durch Einsatz von Websockets und neuer Webtechnologien, die Administration von Hosting Clustern zu vereinfachen und einen sinnvollen Kompromiss zwischen Funktionalität und Usability zu finden.

Danksagung

Die Arbeit entstand bei der Firma Skylime GbR sowie der Fakultät Informatik an der Hochschule Augsburg unter der Leitung von Prof. Dr. Klever. An dieser Stelle möchte ich mich bei einigen Personen bedanken, die mich während dieser Arbeit unterstützt haben.

Hiermit möchte ich mich bei Sebastian Wiedenroth, meinem fachlichem Betreuer bei Skylime, für sein Engagement und seiner Geduld bedanken. Ebenso danke ich Prof. Dr. Nik Klever, der diese Arbeit auf Seiten der Hochschule Augsburg betreut hat und mir diese Abschlussarbeit erst ermöglicht hat.

Ein besonderer Dank gilt meinen Eltern für die Unterstützung während des Studiums. Ohne ihrer Hilfe wäre diese Arbeit niemals entstanden.

Inhaltsverzeichnis

1 Einleitung

1.1 Projektträger

Der Auftraggeber und Träger dieses Projekts ist die Firma SkyLime GbR[1] (im Folgenden SkyLime genannt).

SkyLime ist ein kleines Team, bestehend aus Softwareentwicklern und Systemadministratoren. Seit 2004 betreibt SkyLime Webhosting unter der Marke reco-systems[2]. SkyLime bietet professionelle IT-Services auf Unix- und Linux-Basis. Dazu gehört die hochverfügbare core.io Infrastruktur.

Die core.io Infrastruktur bietet Dienste wie DNS, Email, Datenbanken und Cluster-Dateisysteme für jede Art von Anwendung. Hierbei setzt SkyLime auf Hochverfügbarkeit und dynamische Anpassung der Ressourcen.

Im Vergleich zu großen Systemhäusern setzt SkyLime vor allem auf eine flexible und qualitativ hochwertige Betreuung der Kunden, bei gleichzeitiger Kontinuität und Verlässlichkeit.

Mit dem Sitz am Bodensee ist SkyLime eines der bekannten mittelständischen Unternehmen der Region. Zu den Kunden zählen namhafte börsennotierte Unternehmen aber auch Bildungseinrichtungen und öffentliche Auftraggeber.

1.2 Motivation

Mit den neuen Möglichkeiten durch den kommenden Webstandard HTML5 stehen Entwicklern vielfältige Funktionen zur Verfügung, Anwendern eine bessere Usability und gleichzeitig spannende neue Nutzungserlebnisse zu liefern. Hierbei muss ein sinnvoller

1 Skylime GbR: http://www.skylime.net/

2 reco-systems: http://reco-systems.de/

Kompromiss gefunden werden, der eine einfache Bedienung der Webanwendung voraussetzt und trotz allem alle benötigten Funktionen anbietet.

Auch der Anteil an mobilen Endgeräten im Internet, allen voran durch Smartphones, steigt Jahr für Jahr kontinuierlich an. Dadurch wird von Webapplikationen einiges mehr gefordert als simples darstellen auf einem Desktop Rechner. Es wird immer wichtiger eine Cross-Browser und Geräte unabhängige Plattform zu entwickeln, an deren erster Stelle die Benutzerfreundlichkeit („Web-Usability“) sowie eine weitestgehende barrierefreie Nutzung in Vordergrund stehen. Durch den immer weiter steigenden Verkauf von Smartphones wächst auch die Verwendung mobiler Webseiten. Diese Webseiten haben jedoch andere Anforderungen als Webseiten für Laptops oder Desktop Computer. Die Herausragendsten Merkmale sind der kleinere Bildschirm sowie die Bedienung mittels Touchscreen. Hierauf muss beim Aufbau der Webapplikationen genauso geachtet werden, wie eine Ressourcenschonende Struktur, da auf mobilen Endgeräten die mobile Datennutzung durchaus noch als sehr teuer zu bezeichnen ist. [marketshare12]

Wie Google mit seinem großen Produkt-Portfolio[3] bereits zeigt, verlagern sich klassische Desktop-Programme immer mehr ins Internet. Auch Microsoft stellte zuletzt ihr Office Büroprogramm als reine Webanwendung den Benutzern zur Verfügung.[4] Aufgrund der Verschiebung von klassischer Software in die „Cloud“ bzw. zu einer Anwendung die immer und überall erreichbar und nutzbar sein sollte, ergeben sich neue Probleme die es zu bewältigen gilt.

1.3 Problemstellung

Der Administrations- und Verwaltungsbereich einer Webhosting-Backends muss vielfältige Aufgaben übernehmen um ihren Kunden alle wichtigen und benötigten Funktionen zur

3 Auflistung Webbasierter Google Produkte: http://en.wikipedia.org/w/index.php?title=List_of_Google_products&oldid=505899549#Web-based_products

4 Microsoft Office365: http://www.office365.com/

Verfügung stellen um deren Webseite, Datenbanken oder Cloudinstanzen zu administrieren.

Die Aufgabe des Backends liegt darin ein Kontroll- und Administrationszentrum als Webplattform bereitzustellen, welches den Kunden einfach zugänglich, übersichtlich und leicht zu bedienen ist.

Erfahrungen der Kunden zeigen, dass das aktuell zur Verfügung stehende Interface diese Aufgaben nicht vollständig übernimmt und dabei in keinster Weise ein intuitives Bedienkonzept aufweisen kann.

Um dieses Problem anzugehen wurde ein Webfrontend entwickelt um die Verwaltung von Webhosting Clustern zu vereinfachen. Dieser Prototyp soll durch Einsatz neuer Webtechnologien, wie Websockets[5] und Canvas[6], erweitert werden um eine einfache Bedienung und schnelleres zurechtfinden innerhalb des Webfrontends zu erreichen.

1.4 Zielsetzung

Es soll ein verteiltes System entwickelt werden mit dem ein Hosting Cluster verwaltet werden kann. Der Hosting Cluster stellt Dienste wie Webseiten, Email, Datenbanken und DNS zur Verfügung. Das User Interface soll als Webanwendung realisiert werden. Das System soll hochverfügbar und skalierbar sein.

Die Webapplikation kommt nicht nur auf Desktop Computern zum Einsatz, sondern auch auf mobilen Geräten wie Tablets oder Smartphones, somit ist auf eine einfache und intuitive Bedienung zu achten. Bei der Anwendung steht neben der Benutzerfreundlichkeit auch das Austesten von modernen Webtechnologien in Vordergrund.

Zum Aufbau des Webfrontends wird das Python Webframework Django[7] verwendet. Als

5 Websocket API: http://www.w3.org/TR/websockets/

6 Canvas HTML5 Element: http://www.w3.org/TR/2010/WD-html5-20100624/the-canvas-element.html

7 Django Webframework: https://www.djangoproject.com/

Schnittstelle zwischen den virtuellen Servern und der Webanwendung wird libvirt[8], eine API zur Verwaltung von verschiedenen Virtualisierungstechnologien, zum Einsatz kommen. Die Kommunikation des VNC-Streams wird mittels Websockets ermöglicht und per Canvas in die Webanwendung gezeichnet.

1.5 Produktiveinsatz

Das System ist für den Einsatz in Webhosting Firmen gedacht.

Es soll Administratoren, die den Cluster verwalten, manuelles bearbeiten von Konfigurationsdateien abnehmen. Kunden können selbstständig ihre Webseiten, Datenbanken und Emailkonten verwalten.

1.6 Unique selling proposition

Es gibt bereits Produkte (cPanel[9], Plesk[10], SysCP[11]) zur Verwaltung von Webhosting Systemen, diese sind aber darauf beschränkt, dass alle Dienste auf einem physikalischem System ausgeführt werden. Ist die Leistungsgrenze dieses Systems erreicht, ist es sehr schwierig mit solchen Produkten zu skalieren.

Durch den vollständig verteilten Design-Ansatz skaliert das System weiter als bestehende Produkte. Ebenso verbessert sich die Ausfallsicherheit.

1.7 Zielgruppe der Thesis

Die Leserzielgruppe der Arbeit umfasst Entwickler von Webanwendung und Systemadministratoren. Bei der Entwicklung wird Wert auf Usability genommen, daher ist die Thesis ebenso für UX-Designer und Frontend Entwickler interessant.

8 libvirt: http://libvirt.org/
9 cPanel: http://cpanel.net/
10 Plesk: http://www.parallels.com/de/products/plesk
11 SysCP: https://github.com/flol/SysCP

1.8 Struktur und Aufbau der Arbeit

Die Bachelorarbeit ist in mehrere Teilbereiche gegliedert, die den Verlauf des Projekts widerspiegeln.

In der *Einleitung* wird der Projektträger vorgestellt und die Ziele der Arbeit beschrieben. Die für das Projekt relevanten Begriffe, sowie der verwendeten Technologien werden in den *Grundlagen* erklärt.

Im Kapitel *Analyse* erfolgt eine Darstellung der aktuellen Situation und welche Möglichkeiten im Web bereits möglich sind. Hier wird auf verschiedene, vorhandene Bibliotheken eingegangen und deren Vor- und Nachteile erläutert. In diesem Abschnitt wird auch die Konzeption der Anwendung dargestellt.

In der *Realisierung* werden die technischen Aspekte der Entwicklung der Webanwendung angesprochen und die Quelltexte näher erläutert.

Den Abschluss der Thesis bildet das Kapitel *Fazit & Ausblick*. Hier wird ein Fazit über das gesamte Projekt und die Arbeit gezogen. Es folgt ein Ausblick auf den Verlauf des Projekts außerhalb der Bachelor-Arbeit.

1.9 Formale Hinweise

Unbekannte Begriffe oder weitere Hinweise zu einem Wort werden per Fußnote erklärt. Meist erfolgt ein weiterführender Link zu einer Internetseite. Zitate oder Quellenverweise sind per Eckiger-Klammer gekennzeichnet und im Quellenverzeichnis beschrieben. Verweise auf Kapitel sind *kursiv* geschrieben, während wichtige Textstellen in Fettdruck abgebildet sind.

2 Grundlagen

In diesem Kapitel werden einige Grundlagen und Grundbegriffe näher erläutert. Die Erklärung erfolgt auf Basis der benötigten Informationen für die Bachelorarbeit. Zudem werden die verwendeten Technologien kurz vorgestellt.

2.1 Frameworks

Als Framework wird ein Programmiergerüst in der Softwaretechnik verstanden. Das insbesondere im Rahmen der objektorientierten Entwicklung sowie bei komponentenabhängigen Entwicklungsansätzen verwendet wird.

Frameworks sind keine fertigen Programme, sondern stellen ein Grundgerüst bereit, das dem Programmierer einfache Aufgaben abnimmt und ihm eine geeignete Rahmenstruktur zur weiteren Entwicklung seiner Anwendung vorgibt. Meist gibt hierbei das Framework die Anwendungsarchitektur vor. Von daher sollte für jeden Anwendungsfall eine genaue Analyse der zur Verfügung stehenden Frameworks gemacht werden. [ramsey05]

Im Folgenden werden die in der Bachelorarbeit verwendeten Web-Frameworks genauer vorgestellt.

2.1.1 Django

Das in Python geschriebene Web-Framework Django wurde im Jahr 2005 unter einer BSD-Lizenz[12] veröffentlicht und wird unter der Leitung der Django Software Foundation[13] weiterentwickelt. Mit Django soll das Entwickeln komplexer und datenbankengesteuerter Webseiten vereinfachen und beschleunigen. Django setzt komplett auf Python, selbst bei Konfigurationsdateien oder Datenbankmodellen.

12 BSD-Lizenz: http://opensource.org/licenses/bsd-license.php

13 Django Software Foundation: https://www.djangoproject.com/foundation/

Das Framework betont die Wiederverwendbarkeit und „pluggability“ der einzelnen Komponenten, sowie den Ansatz des Rapid Development.

> *„Every piece of knowledge must have a single, unambiguos, authoritative represenation within a system.“*
>
> *– Don't Repeat Yourself (DRY) Prinzip*

Django stützt die Prinzipien des Don't Repeat Yourself (DRY), die besagen Redundanz zu vermeiden oder zumindest zu reduzieren. [ford09]

Einige bekannte und große Webseiten setzen auf Django als Webframework, darunter Pinterest[14], Instagram[15] oder Mozilla[16].

2.1.1.1 Komponenten

Django folgt dem MVC-Pattern[17], jedoch in einer etwas abgewandelten Form, des so genannten Model-Template-View (MTV). [django05]

Django's Kern des **MVC** Frameworks besteht aus einer Objekt relationalen Abbildung (object-relational mapper, ORM), welcher zwischen den Datenmodellen (als Python Klassen) und einer relationalen Datenbank („**M**odell“) interagiert. Ein System um Anfragen in ein Templates zu verarbeiten („**V**iew“) und einem URL-Dispatcher[18] („**C**ontroller“), der aus regulären Ausdrücken aufgebaut ist und so dem Entwickler jegliche Freiheiten lässt. [holovatykaplanmoss07]

Abbildung 1 zeigt den grundlegenden Aufbau von Django's MVC Framework.

14 Pinterest: http://pinterest.com/
15 Instagram: http://instagram.com/
16 Mozilla Foundation: https://www.mozilla.org/en-US/
17 MVC-Pattern: http://c2.com/cgi-bin/wiki?ModelViewController
18 URL-Dispatcher: https://docs.djangoproject.com/en/dev/topics/http/urls/

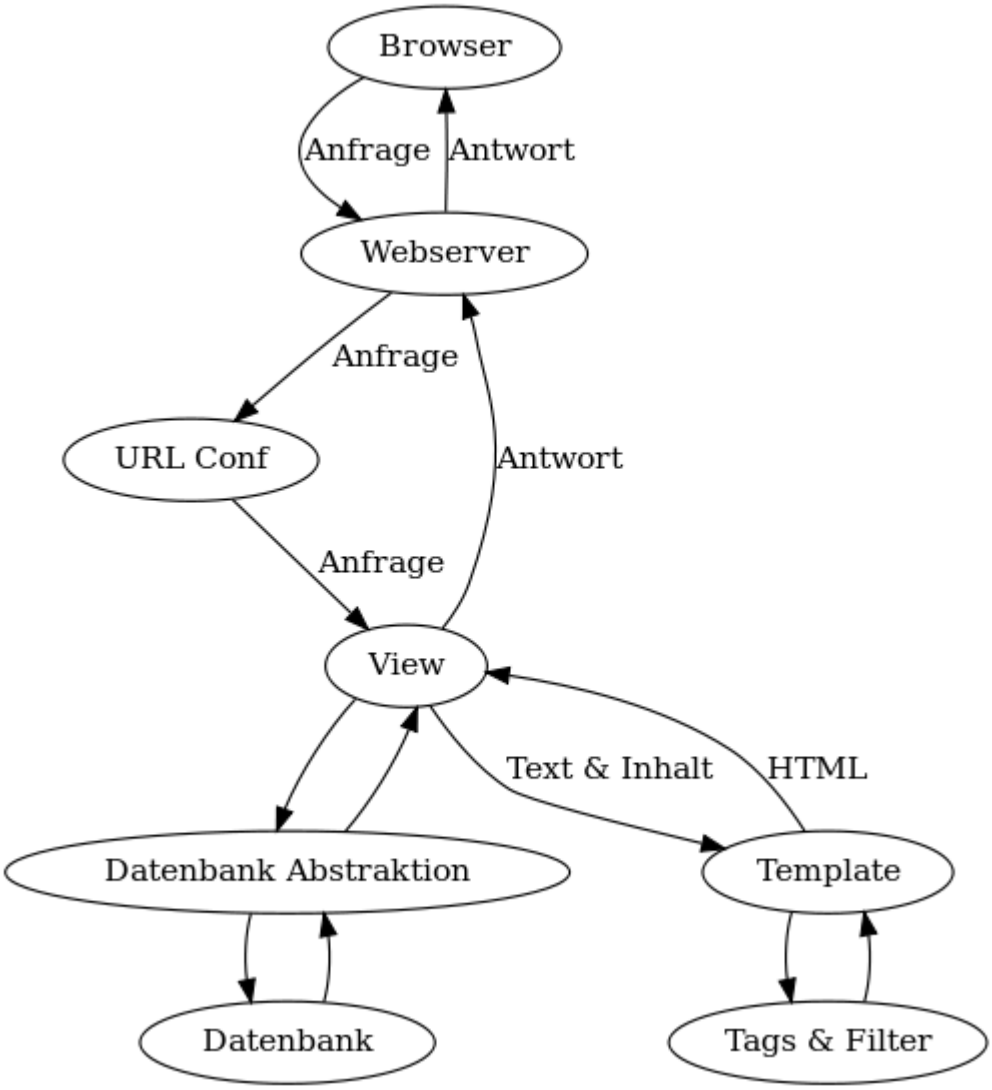

Abbildung 1: Model-Template-View in Django

Django hat im Kern weitere Komponenten integriert, dazu zählt unter anderem ein einfacher, zum Entwickeln und Testen gedachter, Webserver. Django unterstützt Middleware-Klassen, die in verschiedenen Stadien der Request-Verarbeitung eingreifen. Das Framework validiert und serialisert HTML Formulare direkt und hat mehre integrierte Caching-Frameworks. Zudem bietet es ein eigenes System um die Template-Engine zu erweitern und hat eine Schnittstelle zum Python eigenen Unit-Test-Framework. Die interne Kommunikation ist über Signale möglich. Ein weitere große Kernkomponente ist die Internationalisierung von Templates, Datenbanktabellen und Django's eigner Komponenten.

2.1.1.2 enthaltene Applikationen

Django besitzt eine Reihe von enthaltenen Applikationen[19] im „contrib“-Packet des Frameworks, darunter unter anderem:

- ein erweiterbares Authentifizierungssystem
- ein dynamisches Administrations-Interface, das sich ebenfalls erweitern lässt
- ein flexibles Kommentarsystem
- Sicherheitstools[20] um Cross-Site-Scripting, Cross-Site-Request-Forgery, SQL-Injections und weiteren Angriffen vorzubeugen
- Werkzeuge um Sitemaps, RSS und Atom Feeds zu erstellen

2.1.2 Node.js

Node.js[21] ist ein serverseitiges Framework, zum Erstellen von hochskalierbaren Internetapplikation und Konsolenanwendungen. Es setzt die JavaScript-Engine V8[22] von Google ein, die JavaScript-Code vor der Ausführung in Maschinensprache übersetzt. V8 ist derzeit eine der schnellsten JavaScript Implementierungen. Der Entwickler des Frameworks setzen hohen Wert auf die Skalierbarkeit, die vor allem eine große Anzahl gleichzeitiger Verbindungen ermöglichen soll. [heise10] [ochs12]

Projekte die auf Node.js aufsetzen, sind unter anderem PDFKit[23], WebOS[24] von HP oder StackVM[25].

19 Enthaltene Django Applikationen: https://docs.djangoproject.com/en/1.4/#other-batteries-included
20 Django Sicherheitstools: https://docs.djangoproject.com/en/1.4/topics/security/
21 Node.js: http://nodejs.org/
22 JavaScript-Enginge V8: http://code.google.com/p/v8/
23 PDFKit: http://pdfkit.org/
24 WebOS: https://developer.palm.com/
25 StackVM: http://stackvm.com/

2.1.2.1 Architektur

Eine Besonderheit von Node.js ist die ereignisgesteuerte Architektur, durch dem der Programmablauf nicht durch Ein-/Ausgabeoperation, wie zum Beispiel Datenbankenzugriffe, blockiert wird. Die einzelnen Ergebnisse der Operation, die in anderen Umgebungen blockieren würden, werden immer als Rückruffunktion (Callback) übermittelt. Somit wird immer nur Thread benötigt, wodurch auch kein Locking nötig ist und Race-Conditions[26] vermieden werden. [geisendoerfer10]

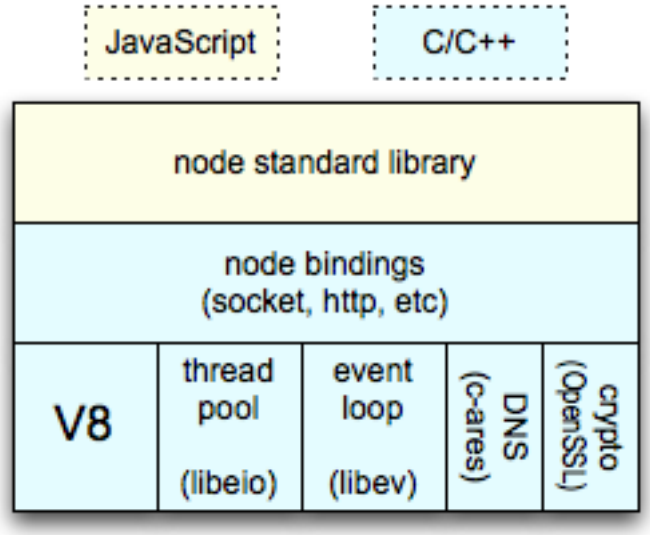

Abbildung 2: Architektur von Node.js

Der Vorteil von Node.js liegt klar auf der Hand. Eine Instanz kann so bei gleichem Arbeitsspeicher-Verbrauch erheblich mehr Verbindungen auf einmal aufrecht halten, als vergleichbare Anwendungen mit Ein-/Ausgabe-Architektur. Vor allem im Serverbetrieb ist dies ein großer Vorteil.

2.1.2.2 Komponenten

Node.js enthält einige Basismodule im Kern[27], welche direkt in das Binärpaket kompiliert wurden. Dazu gehören neben dem Net-Modul für asynchronen Netzwerkzugriff auch

26 Race-Conditions: http://www.freebsd.org/doc/de/books/developers-handbook/secure-race-conditions.html

27 Node.js Basismodule: http://nodejs.org/api/

Wrapper für das Dateisystem, Buffer, Timer und eine allgemein gehaltene Stream-Klasse. Trotz der enthaltenen Module ist der Kern von Node.js relativ schlank gehalten. Weitere Komponenten, beispielsweise von Drittanbietern, können ebenfalls benutzt werden. Zum Beispiel über vorkompilierte Dateien mit der Dateinamenerweiterung .node oder in Form von einfachen JavaScript-Dateien. Diese Module folgem dem CommonJS-Standard für Komponenten und stellen somit über eine exports-Variable Zugriff auf Funktion und Variablen des entsprechenden Moduls her. [herron11]

2.2 HTML5

Der kommende Webstandard HTML5 ist eine textbasierte Auszeichnungssprache zur Strukturierung und semantischen Auszeichnung von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten. Sie soll die Nachfolge von HTML4 antreten und befindet sich zurzeit noch in der Entwicklung. Laut dem Zeitplan des W3C[28] soll HTML5 2014 offiziell verabschiedet, d.h. Zu einer W3C-Recommendation werden. Bereits im Mai 2011 wurde der Status des „Last Call“ ausgerufen, welcher bedeutet, dass HTML5 faktisch bereits einen fertigen Zustand angenommen hat. In den meisten Browsern ist HTML5 bereits (wenn auch unvollständig) implementiert. Das W3C empfiehlt bereits heute HTML5 einzusetzen[29], gegeben falls mit Fallbacks für ältere Browser.

Die ersten Ziele von HTML5 wurde vom „Erfinder des WWW“, Tim Berners-Lee, in dessen Blogeintrag „Reinventing HTML“ festgelegt. Nach Gründung einer Arbeitsgruppe wurden diese Zielsetzungen ausführlicher beschrieben. [bernerslee06] [lilleybernerslee11]

- **Kompatiblität**

  Bestehender Inhalt muss weiterhin unterstützt werden. Neue Elemente der Sprache dürfen den bestehenden Inhalt nicht negativ beeinflussen.

28 World Wide Web Consortium (W3C): http://www.w3.org/

29 W3C – When can I use HTML5: https://www.w3.org/html/wiki/FAQs#When_can_I_use_HTML5.3F

- **Verwendbarkeit**

  Neue Funktionen sollen echte Probleme lösen, und dies vorrangig für Autoren, dann Browserhersteller und zuletzt der „reinen Lehre“ dienend. Funktion jedoch, die bereits einen bestimmten Zweck erfüllen, sollen nicht neu erfunden werden.

- **Sicherheit**

  Bei der Entwicklung neuer Funktionen müssen Sicherheitsaspekte berücksichtigt werden.

- **Konsistenz**

  Teile aus XML, die in XHTML Anwendung finden, sollen auch in HTML erlaubt werden. HTML und XHTML besitzen eine gemeinsame DOM-Abbildung.

- **Vereinfachung**

  Durch genau definiertes Verhalten (auch in Fehlersituationen) und geringe Komplexität soll HTML interoperabel implementiert werden können.

- **Universalität**

  HTML soll auf allen Endgeräten und mit Inhalt in allen Weltsprachen verwendbar sein.

- **Barrierefreiheit**

  Die Barrierefreiheit von Inhalt und Funktion soll gewährleistet werden.

HTML5, so wie es das W3C definiert[30], besteht insgesamt aus mehreren Spezifikationen und Dokumenten. Im Folgenden werden, die für die Thesis relevanten, neuen APIs und Elemente genauer erläutert.

2.2.1 Websockets

WebSockets ist eine Webtechnologie, die eine bidirektionale Vollduplexkommunikation

30 HTML5 Working Draft: http://www.w3.org/TR/html5/

über eine einzige TCP-Verbindung herstellt. Die API wurde vom W3C, sowie das WebSocket-Protokoll vom Internet Engineering Task Force (IETF)[31] als RFC 6455[32], standardisiert.

WebSockets wurde für Browser und Webserver entworfen, kann aber von jedem Client oder jeder Serverapplikation verwendet werden. Durch das Protokoll wird es möglich eine größere Interaktion zwischen einem Browser und einer Webseite zu erreichen. Möglich wird dies durch Bereitstellung einer standardisierten API, wie der Server Nachrichten empfängt bzw. an den Client weiterleitet. [weßendorf11]

2.2.1.1 Vorteile gegenüber HTTP

Bei einer reinen HTTP-Verbindung wird jede Aktion des Servers durch eine vorhergehende Anfrage des Clients ausgelöst. Beim WebSocket-Protokoll reicht es wenn der Client die Verbindung einmal öffnet. Die offene Verbindung kann im folgenden aktiv vom Server verwendet werden. Der Server muss also nicht mehr Anfragen vom Client abwarten, sondern kann neue Informationen ausliefern, ohne auf eine neue Verbindung des Clients zu warten. [ublkitamura12]

Ein Server-Push ist bei einer reinen HTTP-Verbindung nur durch verzögerte Antworten und Polling des Clients möglich. Zudem entfällt bei WebSockets der HTTP-Overhead, der bei jeder Anfrage einige Hundert Byte umfassen kann.

Technisch gesehen, startet das WebSocket-Protokoll wie ein normaler http-request, nur das nach der Übertragung der Header, die zugrundeliegende TCP-Verbindung bestehen bleibt und dadurch für binäre bzw. Zeichenketten-Übertragung in beide Richtungen frei wird.

Die WebSocket-Protokoll-Spezifikation definiert zwei neue URI-Schemen, `ws://` für unverschlüsselte und `wss://` für verschlüsselte Verbindungen.

31 Internet Engineering Task Force: http://www.ietf.org/
32 RFC 6455: http://tools.ietf.org/html/rfc6455

2.2.1.2 WebSocket-Protokoll-Handshake

Zu Beginn einer jeden Verbindung führen Server und Client einen sogenannten Handshake durch. Dieser ähnelt vom Aufbau her dem HTTP-Header und ist vollständig abwärtskompatibel zu diesem, was die Nutzung des Standard-HTTP-Ports 80 zugleich für normale HTTP-Kommunikation als auch für die Websocket-Nutzung ermöglicht. Der Handshake beinhaltet außerdem weitere Informationen, wie zum Beispiel die verwendete Protokollversion. [lubbersgreco12]

Anfrage des Client:

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Listing 1: Anfrage des Clients

Serverantwort:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

Listing 2: Antwort des Servers

Der Server interpretiert den Handshake als HTTP und wechselt dann zum WebSocket-Protokoll. Der Client sendet einen `Sec-WebSocket-Key`, welcher base64 decodiert ist. Die resultierende Antwort tritt in der `Sec-WebSocket-Accept` auf.

Details:

- `x3JJHMbDL1EzLkh9GBhXDw==258EAFA5-E914-47DA-95CA-C5AB0DC85B11` wird SHA-1 gehashed und ergibt `0x1d29ab734b0c9585240069a6e4e3e91b61da1969` als Hexadezimalwert.
- Die Kodierung der SHA-1 Hash in base64 Werten ergibt `HSmrc0sMlYUkAGmm5OPpG2HaGWk=`, welcher der `Sec-WebSocket-Accpet` Wert ist.

2.2.1.3 Browser Unterstützung

Google Chrome 16, Firefox 11, Safari 6 und Internet Explorer 10 sind derzeit die einzigsten Browser, die die letzte Spezifikation des WebSocket-Protokolls unterstützen. Die Autobahn Testsuite[33] zeigt die Konformität der einzelnen Browser zu bestimmten Aspekten des Protokolls.

2.2.2 Canvas

Das in HTML5 eingeführte Canvas-Element gestattet ein dynamisches Zeichnen von Rastergrafiken mittels JavaScript. Ursprünglich wurde Canvas als Bestandteil von WebKit von Apple[34] entwickelt, bis es später von der Arbeitsgruppe WHATWG[35] standardisiert worden ist.

Canvas umfasst einen mit Höhen- und Breitenangaben beschriebenen Bereich innerhalb von HTML. Es stehen zudem 2D-Zeichenfunktionen zur Verfügung, um dynamisch erzeugte Zeicheninhalte innerhalb des Canvas-Bereiches zeichnen zu können.

2.2.2.1 Fähigkeiten

Canvas erlaubt das Zeichnen von normalen Linien- und Rechteckszeichenfunktionen, sowie

33 Autobahn Testsuite: http://autobahn.ws/testsuite/reports/clients/index.html
34 Apple: http://www.apple.com/de/
35 Arbeitsgruppe WHATWG: http://www.whatwg.org/

das Zeichnen von Kreisbögen, Bézierkurven (quadratisch und kubisch) und Farbverläufen. Neben Grafiken, die skaliert, positioniert und beschnitten werden können, unterstützt Canvas auch Transparenz und Text. [fulton11]

Wie bei OpenGL[36] und DirectX[37] auch können Objekte in einem Stack abgelegt werden, was die gezielte Manipulation von Objektgruppen ermöglicht. Animationen sind mittels Verwendung von JavaScript Zeitfunktionen möglich.

2.2.2.2 Beispiel

Zum besseren Verständnis des Canvas Elements, ein beispielhafte Implementierung. Im HTML Dokument selbst wird die Canvas Zeichenfläche angelegt.

```
<canvas id="example" width="200" height="200">
      Textanzeige, falls der Browser kein Canvas unterstützt.
</canvas>
```

Listing 3: Canvas HTML Einbindung

Mittels JavaScript kann nun in das Canvas Element gezeichnet werden.

```
var example = document.getElementById('example');
var context = example.getContext('2d');
context.fillStyle = 'red';
context.fillRect(30, 30, 50, 50);
```

Listing 4: Zeichnen mit JavaScript in ein Canvas Element

Der JavaScript-Code in *Listing 4* zeichnet ein rotes Rechteck in das Canvas Element.

2.2.2.3 Browser Unterstützung

Canvas wird von allen aktuellen Versionen der Browser Internet Explorer, Google Chrome,

36 OpenGL: http://www.opengl.org/
37 DirectX Developer Center: http://msdn.microsoft.com/de-de/directx

Firefox, Opera und Safari unterstützt. Alte Versionen des Internet Explorers bieten keine Unterstützung für Canvas, können jedoch mit Plugins[38], welche von Google und Mozilla zur Verfügung gestellt werden, um die Canvas Funktion erweitert werden. [fulton11]

2.3 Cloud Computing

Die Cloud ist ein umschreibender Begriff für den Ansatz IT Infrastruktur, wie zum Beispiel Rechenkapazität, Speicherbedarf oder auch Software über ein Netzwerk zur Verfügung zu stellen. Durch die Aufteilung der einzelnen Ressourcen können diese dynamisch und an den Bedarf angepasst werden.

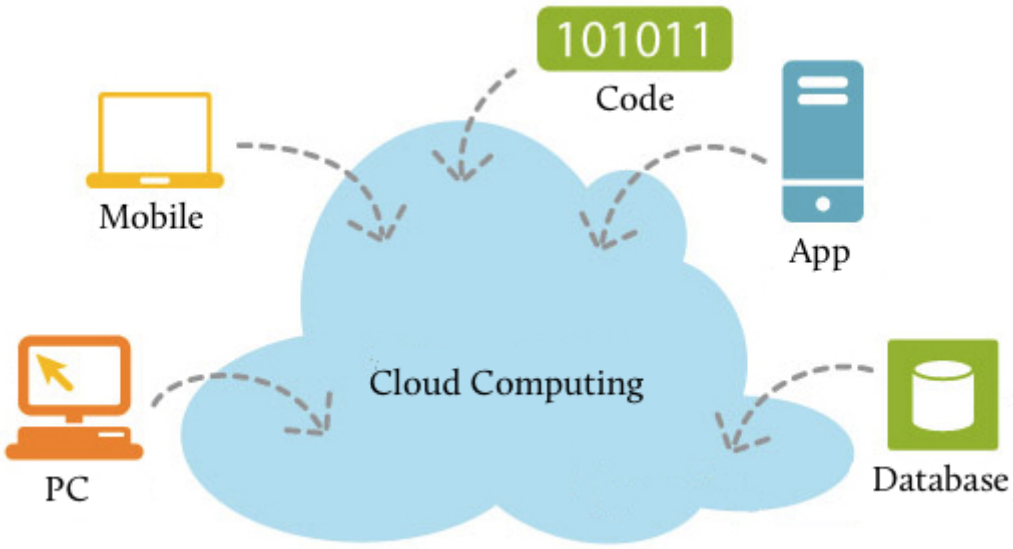

Abbildung 3: Elemente des Cloud-Computing

Vereinfacht kann man sich die Cloud wie folgt vorstellen: Man besitzt ein Online-Kleidergeschäft, mit zwei Mitarbeitern. Diese Mitarbeiter pflegen die Seite, verpacken und versenden die Ware. Während dem Jahr können die zwei Mitarbeiter alle Kunden jederzeit bedienen und es kommt zu keinen Engpässen. Über die Weihnachtszeit ist der Andrang aber so hoch, dass Kunden ihre Kleidung nicht rechtzeitig erhalten, da dass verpacken viel zu lange dauert.

38 Internet Explorer Plugin – Canvas: http://code.google.com/p/explorercanvas/

Nun gibt es für den Besitzer drei Möglichkeiten. Er stellt einen weiteren Mitarbeiter ein, den er aber ein ganzes Jahr beschäftigen müsste und zum Teil nicht ausgelastet ist. Die zweite Möglichkeit wäre, dass der Besitzer auf den Mehrumsatz verzichtet. Oder der Besitzer wendet sich für diese Zeit an einen Dienstleister der das verpacken und versenden für Ihn übernimmt.

Aus Sicht der Informatik, hat der Besitzer einen Online-Shop, dieser läuft auf einem Server. Für die üblichen Benutzerzahlen im Jahr ist dieser Server ausreichend, jedoch steigt die Zugriffszahlen über die Weihnachtszeit. Statt einen weiteren Server zu kaufen, der jährlich Geld kostet, mietet sich der Besitzer für eine geringe Zeitspanne weitere Rechnerressourcen in der Cloud.

2.3.1 Technische Realisierung

Cloud-Computing wird in drei technische Schichten, in einen so genannten Cloud-Stack, aufgeteilt. Jede Schicht stellt einen Grad der Abstraktion dar und wird in diesem Abschnitt näher erläutert. [barton08]

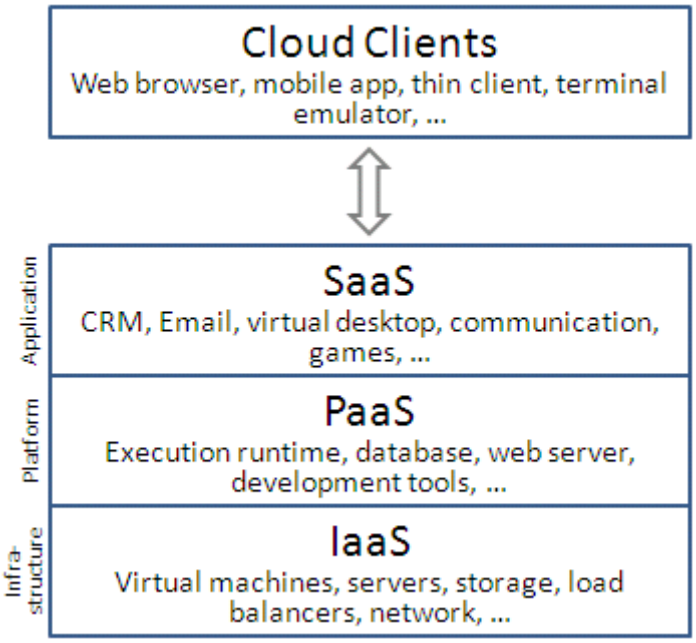

Abbildung 4: Cloud-Stack, die drei technischen Schichten

2.3.1.1 Infrastruktur (IaaS)

„Infrastructure as a Service“ befindet sich an der untersten Schicht im Cloud-Stack. Die Cloud-Provider bieten dem Benutzer Rechnerinstanzen, meist in Form von virtuellen Server, an. Diese kann der Benutzer selbst verwalten. Der Cloud-Dienst ist skalierbar ausgelegt, jedoch nicht zwingend die Programme die der Benutzer auf den Rechnerinstanzen installiert.

Je nach Anforderung kann der Cloud-Dienst um Rechnerinstanzen erweitert oder verkleinert werden. Der Benutzer ist hierbei ab der Betriebssystemebene für seine Instanz verantwortlich.

2.3.1.2 Plattform (PaaS)

Bei „Platform as a Service“ steht die Anwendung des Entwicklers im Vordergrund. Der Entwickler stellt seine Anwendung über die Cloud zur Verfügung. Die Cloud selbst kümmert sich um die Aufteilung auf die Rechnungsinstanzen.

Da der Entwickler nur seine Anwendung liefert, kann die Cloud die Anzahl der tatsächlichen Instanzen jederzeit erhöhen oder reduzieren. In der Cloud werden vom Entwickler gelieferte Daten verarbeitet, das umliegende System ist für ihn nicht einsehbar.

2.3.1.3 Anwendung (SaaS)

Der Benutzer verwendet bei „Software as a Service“ eine bereits bestehende Anwendung in der Cloud. Für ihn sind die Plattform und die Infrastruktur nicht sichtbar.

Beispiele für Cloud-Anwendungen sind unter anderem Google Drive[39], DropBox[40] und Microsoft Office Communication Online[41].

39 Google Drive: http://drive.google.com/
40 DropBox: http://www.dropbox.com/
41 Microsoft Communication Online: http://www.microsoft.com/online/de-de/prodComm.aspx

2.3.2 Organisatorische Arbeiten

Die Cloud wird meist, abhängig vom Anwendungsfall, in drei Organisationsformen eingeordnet. Eine Beschreibung dieser, erfolgt in diesem Abschnitt. [ibm10]

2.3.2.1 Private Cloud

Die Anbieter und Nutzer der „Private Cloud“ stammen aus der selben Organisation oder dem selben Unternehmen. Daten innerhalb dieser Cloud sind nur der Organisation zugänglich und nicht außerhalb erreichbar, dies bietet einen großen Sicherheitsaspekt.

2.3.2.2 Public Cloud

Die „Public Cloud“ ist nicht für eine Organisation beschränkt, sie ist öffentlich erreichbar und für jeden zugänglich. Eine wichtige Rolle spielt hierbei die Datensicherheit. Jeder Benutzer muss selbst Entscheiden welche und wie viele Daten er in der Cloud speichert. Ein bekanntes Beispiel für die „Public Cloud“ sind die Amazon Web Services[42].

2.3.2.3 Hybrid Cloud

Bei der „Hybrid Cloud“ handelt es sich um eine Mischung aus „Private und Public Cloud“. Eine Organisation verwendet eine „Private Cloud“ und wechselt im Fehlerfall oder bei hoher Belastung zur „Public Cloud“, dies wird auch „Cloud-Bursting“[43] genannt.

2.4 Virtualisierung

Für die Virtualisierung gibt es viele verschiedene Methoden, abhängig davon ob eine Anwendung oder ein gesamtes Betriebssystem virtualisiert werden soll. Die Virtualisierung spielt eine wichtige Rolle für die Cloud Infrastruktur. Für die Bachelorabeit ist die

42 Amazon Web Services: http://aws.amazon.com/

43 Cloud-Bursting: http://aws.typepad.com/aws/2008/08/cloudbursting-.html

Bereitstellung ganzer Rechnerinstanzen relevant. Die beste Leistung wird mit der Paravirtualisierung erreicht.

2.4.1 Hardware Virtualisierung

Die virtuelle Maschine stellt dem Gastbetriebssystem Teilbereiche der Hardware in Form von virtueller Hardware zur Verfügung. Somit kann ein unverändertes Betriebssystem darauf in einer isolierten Umgebung laufen. Das Gastsystem muss hierbei für den gleichen CPU-Typ ausgelegt sein. [yao10] [yaocpu10] [yaodisk10]

2.4.2 Emulation

Im Gegensatz zur Hardware Virtualisierung wird dem Gastbetriebssystem die komplette Hardware simuliert. Es ist damit möglich beinahe jedes Betriebssystem in einer virtuellen Umgebung zu starten. Der CPU-Typ muss nicht dem des Hostsystems entsprechen.

2.4.3 Paravirtualisierung

Bei Paravirtualisierung wird zwar ein Betriebssystem virtuell gestartet, jedoch wir keine Hardware virtualisiert, sondern die Betriebssysteme verwenden eine abstrakte Verwaltungsschicht, um auf gemeinsame Ressourcen, wie Netzanbindung oder Festplattenspeicher, zuzugreifen. Damit das Betriebssystem auf der virtuellen Maschine ausgeführt werden kann muss es teilweise portiert werden. Durch diese Portierung kann sich die Leistung der virtuellen Maschine erhöhen. [degelas08]

2.4.4 Schnittstelle zur Virtualisierung

Durch die verschiedenen Virtualisierungstechnologien und Verfahren ist es schwierig für den Entwickler, eine Anwendung zu entwerfen, die mit allen Schnittstellen umgehen kann. Libvirt bietet eine einheitliche Schnittstelle (API) um verschieden Technologien wie Linux

KVM[44], Xen[45] und VMware ESX[46] zu verwalten.

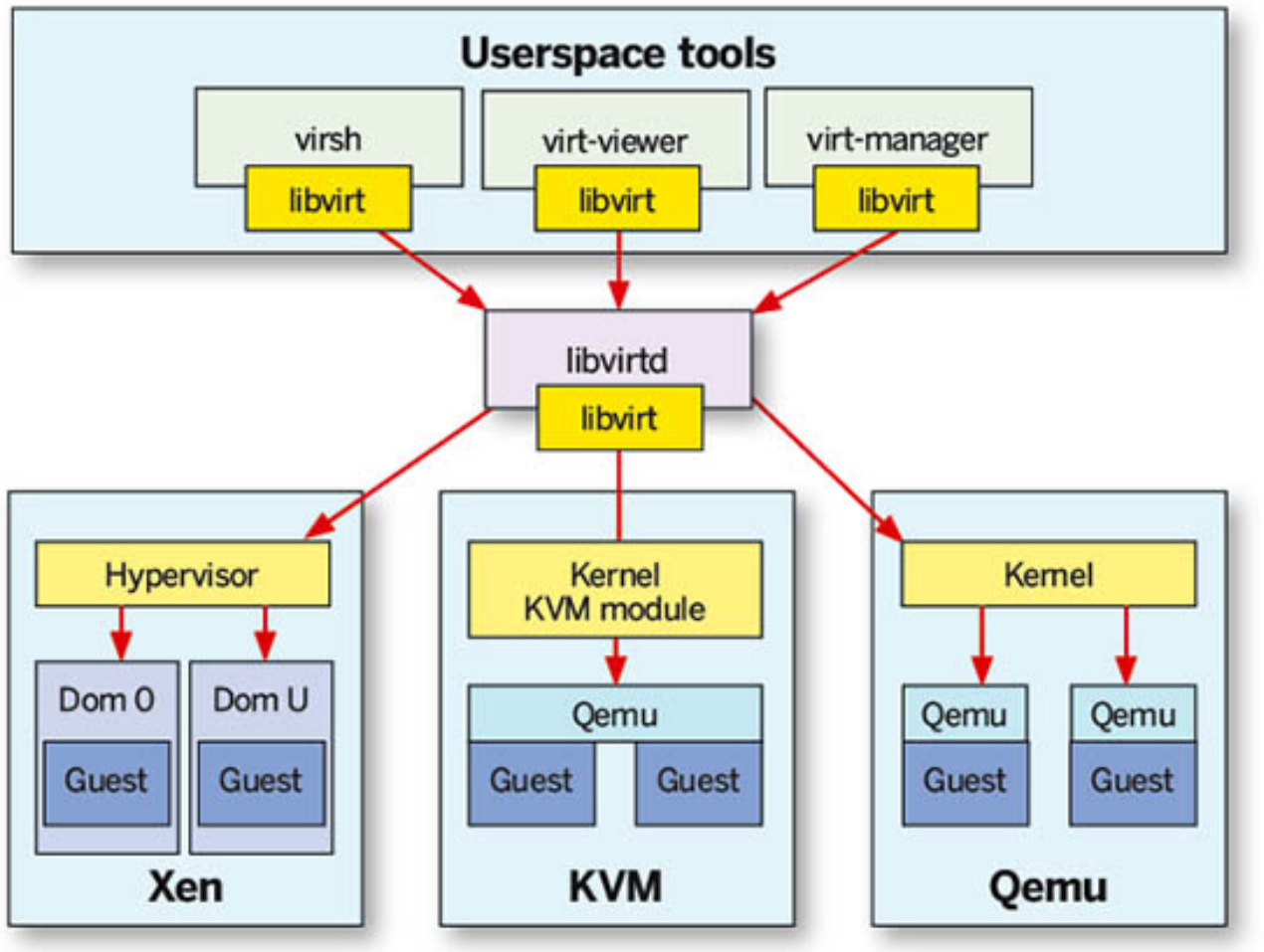

Abbildung 5: Einbindung libvirt Schnittstelle

2.5 Virtual Network Computing

Virtual Network Computing (VNC), ist eine Software um den Bildschirminhalt eines entfernten Rechners (Server) auf einem lokalen Rechner anzeigt und Tastatur- und Mausbewegungen des lokalen Rechners an den entfernten sendet.

VNC wurde im Olivetti Research Laboratory[47] (ORL) entwickelt und später von einer

44 Linux KVM: http://www.linux-kvm.org/
45 Xen: http://www.xen.org/
46 VMware ESX: http://www.vmware.com/
47 Olivetti Research Laboratory: http://www.cl.cam.ac.uk/research/dtg/attarchive/

Reihe von Entwicklern unter der Firma RealVNC[48] weiterentwickelt. Mit dieser Weiterentwicklung kam auch erstmals eine Open-Source Variante von VNC heraus, die unter GNU General Public License[49] veröffentlicht wurde.

Für alle gängigen Betriebssysteme existieren mittlerweile VNC Implementierungen, sodass VNC als plattformunabhängig gilt. Die Software implementiert das Remote Framebuffer Protocol (RFB)[50]. [roebuck11]

2.5.1 RFB

Das Remote Framebuffer Protocol ist ein einfaches Protokoll für den entfernten Zugriff auf eine grafische Benutzeroberfläche. Es arbeitet auf der Ebene des Framebuffers, der grob gerasterten Bildschirmdarstellung entspricht. Daher kann das Protokoll für alle Fenstersysteme verwendet werden. Das Programm dass das Protokoll implementiert, wird als Client bezeichnet, das Programm mit dem Framebuffer wird als Server bezeichnet.

Für die Übertragung eines Bildes des Framebuffers benötigt man eine erhebliche Übertragungsrate, daher sollte RFB nur in Netzwerken mit ausreichender Bandbreite eingesetzt werden. [cambrige02]

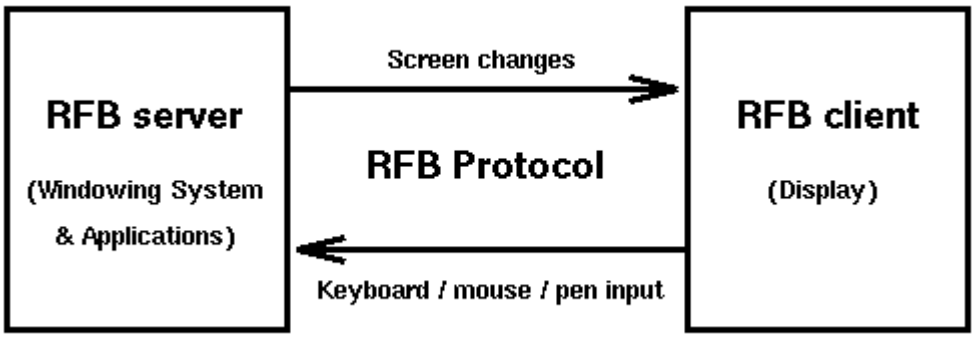

Abbildung 6: Remote Framebuffer Protocol

48 RealVNC: http://www.realvnc.com/
49 GNU Licenses: http://www.gnu.org/licenses/
50 Remote Framebuffer Protocol: https://tools.ietf.org/html/rfc6143

3 Analyse

In diesem Kapital erfolgt die Analyse zur Entwicklung der webbasierten Anwendung.

> *„Plan your work and work your plan.“*
>
> *– Vince Lombardi*

Die ersten Abschnitte beziehen sich auf die Projektplanung. Zum einen auf welche *Zielgruppe* die Anwendung fällt, und eine *Ist-Zustandsanalyse* der bisherigen Implementierung vor Beginn des Projekts.

Die weiteren Abschnitte des Kapitels beziehen sich auf die verschieden Möglichkeiten die mit modernen Webtechnologien realisierbar sind. Hier werden die Grundlegenden Bibliotheken besprochen. Anschließen wird auch auf alternative Anwendungen eingegangen.

Im letzten Abschnitt folgt das Design der Anwendung. Mittels UML Diagrammen wird hier verdeutlicht wie die Implementierung der Webapplikation umgesetzt wird.

3.1 Zielgruppe

Die Definition der Zielgruppe und die Abfragen der Interessen der Anwender ist entscheidend für die Realisierung einer guten Anwendung.

3.1.1 Brainstorming

Um die Zielgruppe und deren Anforderungen besser spezifizieren zu können, wurde ein Brainstorming abgehalten. Durch dieses Verfahren war es einfacher möglich eine genaue Ziel-Definition zu erstellen.

Mit der in *Abbildung 7* erstellten Mindmap konnte eine Definition abgeleitet werden.

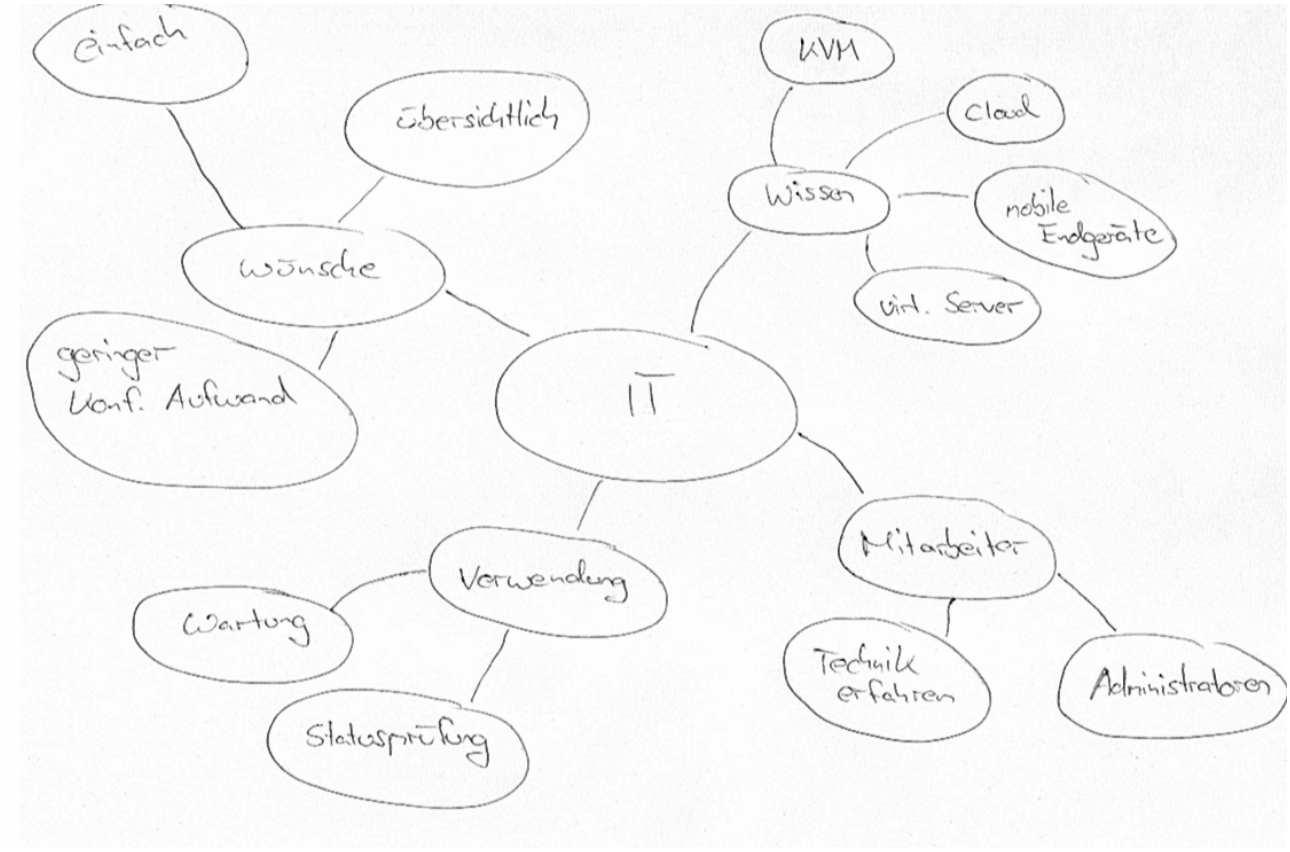

Abbildung 7: Brainstorming über die Zielgruppe

3.1.2 Definition

Die Zielgruppe für das Projekt umfasst Systemadministratoren. Diese haben Erfahrung mit den Betriebssystem Linux. Ebenso sind sie vertraut mit der Handhabung von mobilen Endgeräten.

Den Benutzern ist die Verwendung von virtuellen Server in einer Cloud-Landschaft bekannt. Die Technologie Linux KVM im Zusammenhang mit Hardware-Virtualisierung ist ein Begriff.

Sie verwenden die Anwendung vor allem für Wartung und Statusüberprüfung und wünschen geringen Konfigurationsaufwand und einfache Handhabung.

3.2 Ist-Zustandsanalyse

Zum Zeitpunkt des Projektbeginns existiert noch keine Anwendung zur Verwaltung von Cloudinstanzen im Unternehmen. Ein erster Prototyp zur Administration und Verwaltung von Hosting Instanzen ist bereits in Django erstellt. Hierauf wird bei der Entwicklung aufgesetzt.

Dieser Prototyp kann bereits Mailboxen und Email-Redirects anlegen und verwalten. Erstellt und legt FTP Benutzer, Domains und Backups an. Ebenso können MySQL Datenbanken on-the-fly angelegt und verwaltet werden.

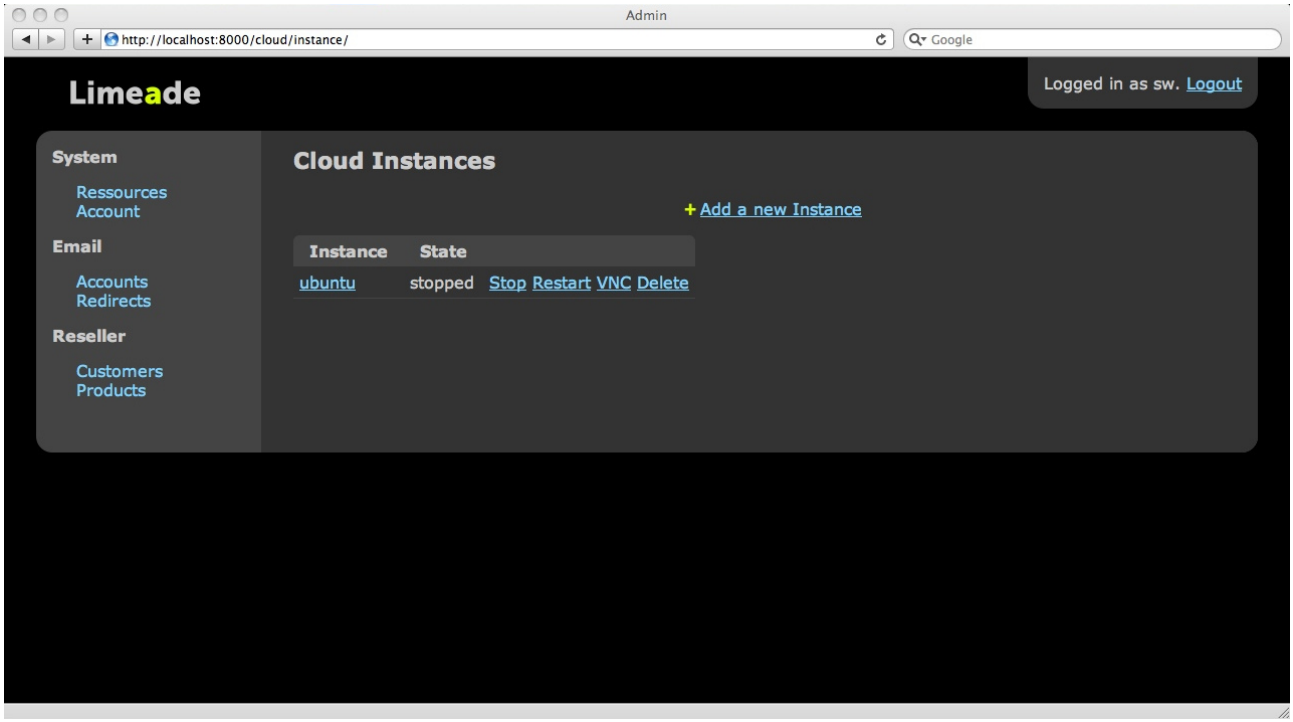

Abbildung 8: Webportal zu Beginn der Arbeit

Die Webanwendung setzt dabei auf die Bibliotheken RabbitMQ[51] und django-celery[52] um das Anlegen der verschiedenen Dienste asynchron ausführen zu können und es so zu keiner Verzögerung in der Benutzung kommt. RabbitMQ nimmt dabei die Anweisungen auf und erstellt eine Nachrichtenwarteschlangen die von django-celery asynchron abgearbeitet wird.

51 RabbitMQ: http://www.rabbitmq.com/
52 django-celery: http://celeryproject.org/

Für die Verwaltung von SSL-Zertifikaten kommt pyOpenSSL[53] zum Einsatz. Die Bibliothek ist dabei ein kleines Modul das auf die OpenSSL Bibliothek aufbaut.

Zwischen der Weboberfläche und der Virtualiserungstechnologie befindet sich die libvirt-Schnittstelle. Durch die Python-API die libvirt mitbringt, ist es möglich in der Webanwendung selbst Cloudinstanzen anzulegen, zu löschen oder Informationen der Instanz abzurufen. Die ganze Logik lässt sich somit ohne Probleme mit Django verwenden.

Da das Webportal bereits auf Django und libvirt aufsetzt, ist es sinnvoll diese auch bei der Entwicklung des In-Browser VNC Viewers im Backend einzusetzen und sauber zu integrieren.

3.3 Anwendungsfunktionen

Ziel der Anwendung ist eine Kommunikation zwischen Webseite und VNC Server herzustellen und den VNC-Stream in den Browser zu zeichnen. Um die Aufgabe der Anwendung besser definieren zu können folgt eine Aufstellung der Funktionen die geboten werden sollen.

- **Virtueller Server**

  Virtuelle Server sind Gast-Systeme auf einem Cloud Server. Der virtuelle Server erhält Rechnerressourcen vom Cloud Server. Um diese zu verwalten werden folgende Funktionen benötigt:

  - Starten
  - Stoppen
  - Neustarten
  - Löschen

53 pyOpenSSL: https://launchpad.net/pyopenssl

- **VNC**

  Um mit dem virtuellen Server zu kommunizieren wird VNC eingesetzt. Der VNC Stream soll folgendes bieten:

  - Echtzeit-Stream in den Browser
  - Maus, Tastatur Benutzung im VNC Fenster
  - Zwischenablage für Host (Webanwendung) und virtualisierten Rechner
  - Bedienbar mit mobilen Endgeräten

- **Sicherheit**

  Keine Funktion im eigentlichen Sinne, jedoch muss die von Anfang an bedacht werden. Die Anwendung soll nicht von außen erreichbar sein und nur den Anwendern zur Verfügung stehen, die auch wirklich Zugriff auf den VNC-Server haben sollen.

  - Benutzerauthentifizierung
  - Session Management
  - Verhinderung bzw. Reduzierung von möglichen Sicherheitslücken
  - Absicherung mittels SSL

Beispielsweise befindet sich der Administrator des virtuellen Servers gerade nicht an seinem Arbeitsplatz, er hat aber sein Tablet-Computer dabei.

Durch die zu entwickelnde Anwendung hat er trotzdem Zugriff auf die virtuelle Maschine über ein Webinterface und kann eine Statusüberprüfung oder dringende Wartungsaufgaben durchführen.

3.4 Möglichkeiten moderner Webtechnologien

Mit HTML5 kamen und kommen immer mehr Möglichkeiten native Webanwendungen zu entwickeln, die keine Plugins wie Flash oder Java für den Browser benötigen. Durch den Wegfall dieser Plugins reduziert sich auch die Code-Dichte auf dem Computer, und somit auch die Angriffsmöglichkeiten und Sicherheitslücken. Gleichzeitig sind moderne Webtechnologien meist nur in aktuellen Browsern implementiert, was zu bedeuten hat dass es meist einen Fallback für ältere Browsergenerationen geben sollte.

Aufgrund der schnellen Entwicklung im Web sind auch neue Herausforderungen an HTML gekommen. Diese sind unter anderem: [comsolit09]

- Grafik (3D/2D)
- Geschwindigkeit (muss sich wie echtes Desktop-Programm anfühlen)
- Speicher / Caching
- Livecontent, Real-Time-Applikation
- JavaScript Performance

HTML5 bietet zur Lösung dieser Herausforderungen neue APIs, Multimedia-Unterstützung und neue Elemente für das Document Object Model (DOM), sowie eine Offline Unterstützung.

Zur Veranschaulichung (*Listing 5*) ein einfaches Beispiel um Videos in eine Webseite einzubinden. Früher ist dies nur mit Hilfe von Flash oder anderen Plugins möglich gewesen.

```
<object width="425" height="344">
    <param
        name="movie"
        value="http://www.youtube.com/v/4guksqag5xi&hl"></param>
    <param
        name="allowfullscreen"
        value="true"></param>
    <param
        name="allowscriptaccess"
        value="always"></param>
    <embed
        src="http://www.youtube.com/v/4guksqag5xi&hl=en&fs=1"
        type="Application/X-Shockwave-Flash"
        allowscriptaccess="always"
        allowfullscreen="true"
        width="425"
        height="344"></embed>
</object>
```

Listing 5: Einbindung eines Videos in HTML mit Flash

Mit HTML5 jedoch ist das Einbingen von Videos um einiges einfacher gehalten.

```
<video src="html5test.ogg" width="320" height="240" controls>
    Anzeige falls der Browser kein HTML5 Video unterstützt.
</video>
```

Listing 6: HTML5 Video

Wie in Listing 6 deutlich zu sehen ist müssen viel weniger Angaben gemacht werden. Die Steuerung ist Browserabhängig und nicht mehr von einem Plugin, das sich teilweise wie ein Fremdkörper auf der Webseite bemerkbar macht, vorgegeben. Um den Code aus Listing 5 als Fallback für ältere Browser anzubieten, genügt es den Code innerhalb des `<video>`-Tags einzubinden.

3.4.1 APIs für Kommunikation

Websites setzen sich heute oft aus Daten verschiedener Quellen zusammen – Ajax macht es möglich, aber auch ältere Techniken wie Iframes oder eingebundene externe Skripte gehen in diese Richtung. Eine nahtlose und zugleich sichere Kommunikation erlaubt jedoch keine dieser Techniken: Eingebundene Skripte sind unsicher, Iframes zu abgeschottet und Ajax kann nur Inhalte von der gleichen Domain ansprechen.

Web-Messaging löst dieses Problem. Mit diesem bereits gut implementiertem Standard können Skripte Nachrichten an eine URL verschicken. Dort muss eine Gegenstelle lauschen, damit etwas passiert.

Push-Anwendungen können zeitkritische Informationen durchs Netz schicken, etwa Börsenticker oder Online-Spieldaten. Hierfür gibt es die Server-sent Events, die allerdings bisher nur in Browsern implementiert sind, die Webkit aufsetzen. Bei Server-sent Events hält der Browser die Verbindung und wartet auf Aktualisierungen.

WebSockets gar brechen komplett aus dem HTTP-Protokoll aus und setzen direkt auf TCP auf, das unterhalb der HTTP Internetverbindungen herstellt. Dabei lassen WebSockets Server-Push zu, beispielsweise für gestreamte Medien. Die weitreichende Implementierung in den Browsern ist ein gutes Argument um auf WebSockets zu setzen, da Server-sent Events diese nicht bieten können und weniger für gestreamte Inhalte gedacht sind. [braun11]

Mittlerweile hat auch das W3C den letzten Entwurf der WebSocket-API in den Status des Last Call erhoben. Dies stellt die letzte Möglichkeit dar Änderungsvorschläge an der Spezifikation einzubringen.

In der Vergangenheit jedoch zeigte sich bei bereits anderen APIs von HTML5 das sich im Status des Last Call keine gravierenden Änderungen mehr ergeben und die Spezifikation somit als quasi Standard angesehen werden kann.

3.4.2 Grafik und Animation

Zeichnen im Web war bisher nicht möglich. Vektorgrafiken wie SVG wurden von keinen Browser richtig unterstützt. Daher war es auch hier nötig auf Dritthersteller Plugins wie Flash oder Silverlight von Microsoft zu setzen. Mit der fortschreitenden Implementierung von HTML5 hat sich dies aber geändert. Alle aktuellen Browser unterstützen das Zeichnen im Web.

Vor knapp 6 Jahren berichtete der in der Webstandard-Szene bekannte Blogger „Isolani“, dass die skalierbare Vektorgrafik (SVG) keine relevante Webtechnologie mehr ist und es nie war. [isolani06]

Mittlerweile ist die SVG-Unterstützung in den meisten aktuellen Browsern sehr gut[54] und somit steht auch der Verwendung nichts im Wege. SVG ist die vom World Wide Web Consortium empfohlene Spezifikation zur Beschreibung zweidimensionaler Vektorgrafiken. SVG basiert dabei auf XML. Manipulationen an dem SVG-DOM sind mit Hilfe eingebetteter Funktionen via Skriptsprachen möglich. Die Vektorgrafik unterstützt auch Animationen mittels SMIL[55], eine ebenfalls auf XML basierende Auszeichnungssprache für zeitsynchronisierte multimediale Inhalte.

Mit Hilfe von JavaScript können in SVG Programme geschrieben werden. Mit dem DOM von SVG kann man die XML-Struktur der Grafik manipulieren.

SVG ist durch seine Implementierung in XML eher dafür gedacht Animationen zu ermöglichen (siehe *Abbildung 9*) und skalierbare Grafiken auf der Webseite einzubinden.

54 SVG-Unterstützung: http://de.wikipedia.org/w/index.php?title=Scalable_Vector_Graphics&oldid=106874878#SVG-Unterst.C3.BCtzung_in_Browsern

55 SMIL: http://www.w3.org/AudioVideo/

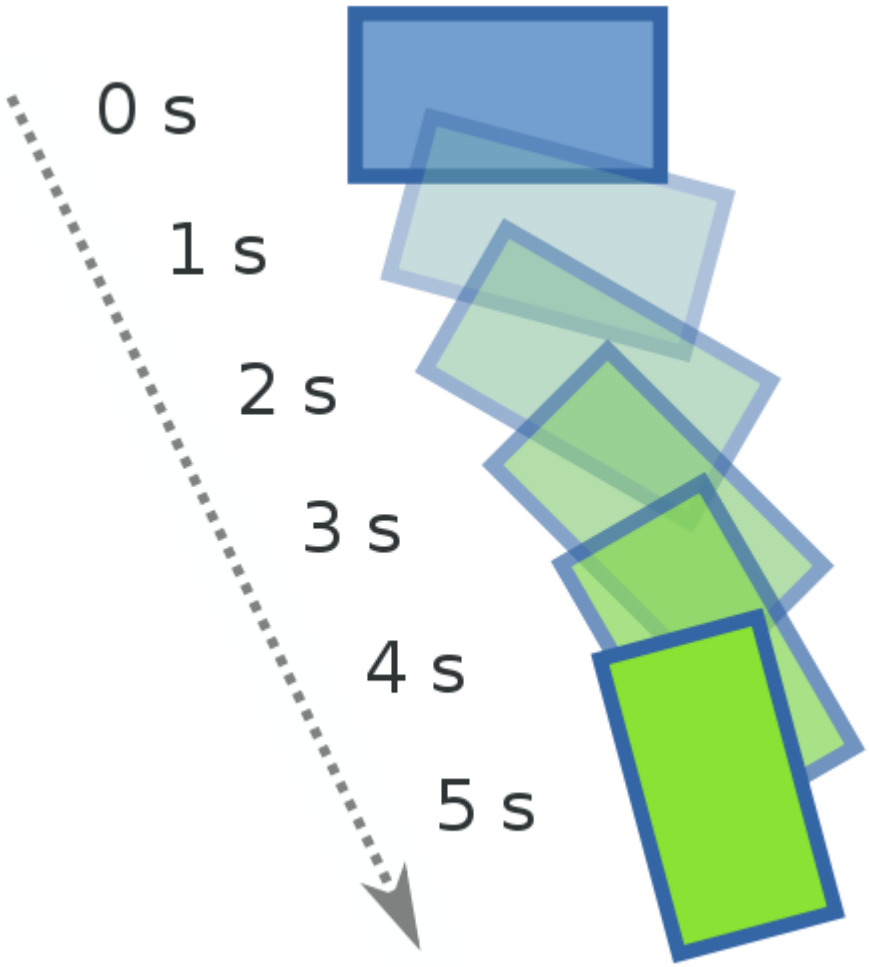

Abbildung 9: Schematisches Beispiel einer Animation in SVG.

WebGL dagegen ist eine relativ neue API. WebGL steht für Web Graphics Library (Web-Grafik-Bibliothek) und ist ein Bestandteil von Webbrowsern, mit dessen Hilfe hardwarebeschleunigte 3D-Grafiken direkt im Browser dargestellt werden können. Die Bibliothek setzt direkt auf OpenGL auf und ist somit plattformunabhängig.

Allerdings ist Unterstützung in eher gering, was zum Teil auch den Sicherheitsbedenken der Browserhersteller geschuldet ist. Diese bieten zwar eine experimentelle Unterstützung für WebGL an, jedoch läuft die neue API ohne direktes aktivieren in den Browsereinstellungen weder im Internet Explorer, Opera oder Mozilla Firefox. WebGL

kann direkt in JavaScript geschrieben werden und ermöglicht somit Hardwarezugriff von einer Webseite aus.

Die 3D-Grafiken werden in das HTML5 Canvas Element gezeichnet und sind somit über den HTML DOM-Baum erreichbar.

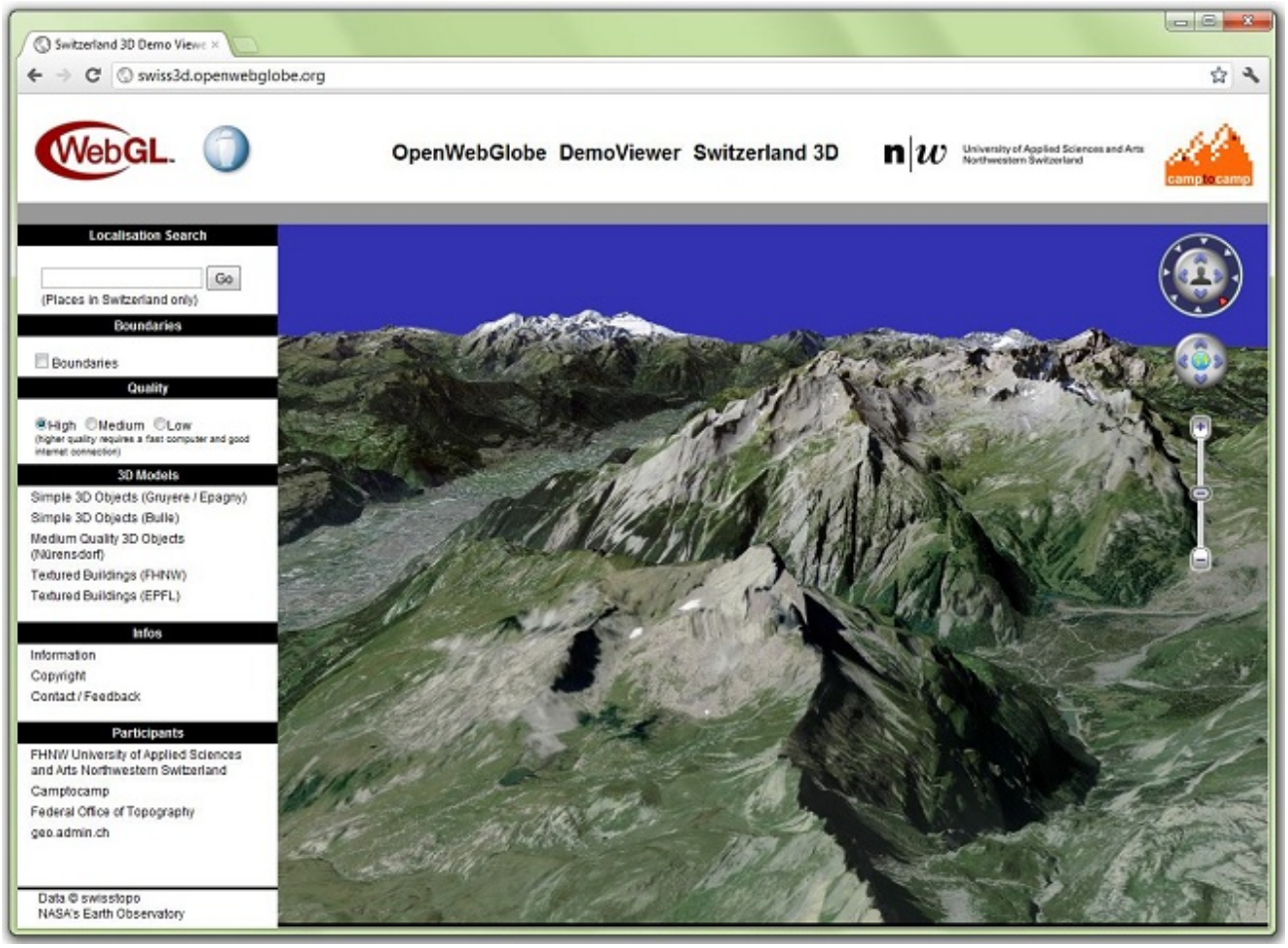

Abbildung 10: WebGL Beispiel: OpenWebGlobe

Da WebGL direkt in das Canvas-Element zeichnet, bietet sich dieses Element als Zeichenregion am Besten an, da so der Stream der virtualisierten Cloudinstanz direkt in den Browser geladen werden kann. SVG oder WebGL ist hierfür weniger geeignet, da dies zum einen mehr Rechenleistung benötigt und zum anderen diese Bibliotheken für andere Zwecke, wie Animationen und In-Browser Games entwickelt wurden.

3.5 Bibliotheken für VNC

Bereits fertige Bibliotheken die das RFB-Protokoll implementiert haben gibt es für native Webanwendungen wenige. Dagegen gibt es einige Applikationen die im Hintergrund Gebrauch von Java machen, diese werden im Abschnitt *Alternative Anwendungen* behandelt.

Die einzige aktiv entwickelte Anwendung die unter Open Source Lizenz steht ist von Joel Martin[56]. Seine Bibliothek noVNC[57] ist unter LGPLv3 lizenziert und kann somit frei verwendet werden.

3.5.1 noVNC

Die HTML5 Client Anwendung noVNC implementiert das RFB Protokoll direkt in JavaScript und stellt den Stream des Servers im Canvas-Element in der Webseite zur Verfügung. Die Applikation unterstützt WebSockets, und die meisten VNC Encodings (raw, copyrect, hextile, tight, tightPNG).

Weitere Funktionen von noVNC sind unter anderem:

- Unterstützung von mobilen Endgeräten
- 24-bit True Color und 8-bit Color Mapping
- Remote und lokaler Browser
- Zwischenablage

Der VNC-Server selbst muss dabei WebSockets unterstützen, was derzeit nur libvncserver und PocketVNC anbieten. Hier muss angesetzt und ein Proxy geschrieben werden, der die Kommunikation von Websockets zu TCP ermöglicht.

56 Joel Martin: https://github.com/kanaka/

57 noVNC: http://kanaka.github.com/noVNC/noVNC/vnc.html

NoVNC unterstützt zwar Verschlüsselung per SSL und somit eine sichere WebSoket-Verbindung, jedoch ist keine Authentifzierungsmöglichkeit innerhalb der Webanwendung gegeben.

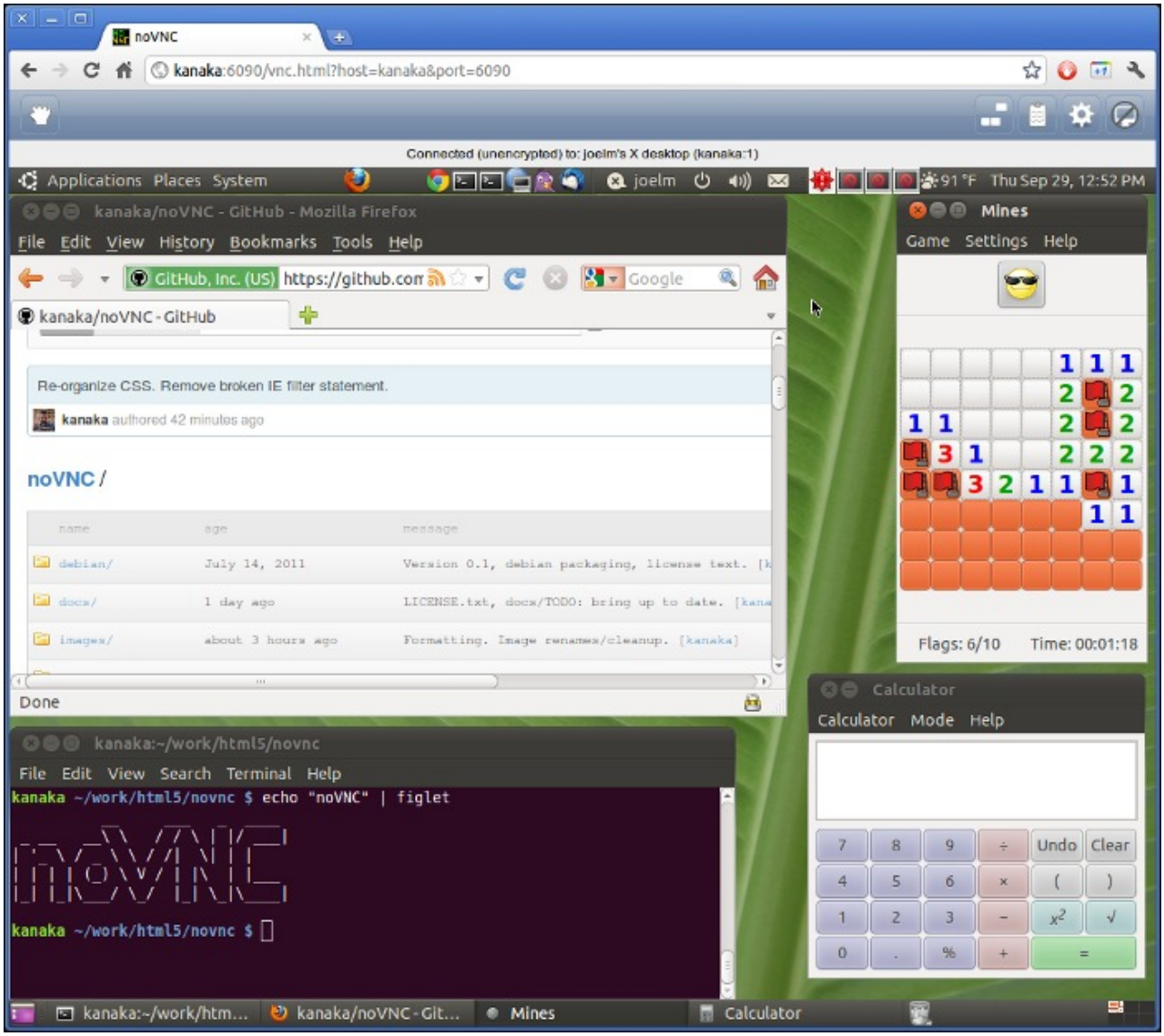

Abbildung 11: Screenshot von noVNC in Google Chrome

Die Anforderungen an den Browser sind relativ gering. Lediglich Unterstützung von HTML5 Canvas und WebSockets sind dringend notwendig. Für ältere Browser bietet es sich hier an einen Flash Fallback zu implementieren, der WebSockets emuliert.

Die Browser sollten jedoch eine schnelle JavaScript Engine besitzen um Verzögerungen in der Darstellung und Kommunikation zu vermeiden.

Ein großer Nachteil von noVNC, bzw. der reinen Implementierung in JavaScript ist das internationale Tastaturlayouts nicht unterstützt werden. Das heißt alle Eingaben werden beispielsweise von einem deutschen Layout gemacht und kommen als die Englische Variante an.

Aufgrund der fortgeschrittenen Entwicklung von noVNC, aktuelle Version ist 0.3 und Verbreitung der Anwendung (unter anderem gibt es fertige Debian Pakete[58]) wurde entschieden die Bibliothek zu integrieren und um benötigte Funktionen zu erweitern. Darunter fallen unter anderem der Proxy und Sicherheitsfunktionen.

3.6 Alternative Anwendungen

In diesem Abschnitt werden kurz alternative Anwendungen betrachtet und beschrieben, die jedoch wegen fehlender Funktionen oder Abhängigkeiten von Drittherstellern nicht in Betracht gezogen wurden.

3.6.1 Guacamole

Guacamole[59] ist ein HTML5 und JavaScript (Ajax) Viewer für VNC, der allerdings die Verwendung eines Server-seitigen Proxys in Java voraussetzt. Die aktuellste Version läuft in jedem Browser der das HTML5 Canvas Element unterstützt.

Ein großer Vorteil ist jedoch durch den Java Proxy gegeben: Es wird Unterstützung für internationale Keyboard Layouts geboten, was durch reine Webanwendungen schwer umsetzbar ist, da jedes Keypress-Event abgefangen und auf Belegung geprüft werden muss.

58 noVNC Debian Paket: http://packages.debian.org/sid/novnc
59 Guacamole: http://guac-dev.org/

Abbildung 12: Screenshot von Guacamole

Nachteile von Guacamole sind unter anderem die Voraussetzung eines Java Server, wie Tomcat, und dass die Anwendung keine ständige Verbindung zwischen VNC und Webinterface bietet. Als Echtzeitanwendung kann Guacamole somit nicht bezeichnet werden.

3.6.2 TightVNC Java Viewer

TightVNC[60] ist eine kostenlose Remote-Control Software. Das Paket läuft Unabhängig vom Betriebssystem, aufgrund der Verwendung von Java und bringt ein Browserplugin, ein Java-Applet, mit.

Das gesamte Softwarepaket bringt auch einen kleinen Webserver mit. Mit diesem kann dann per Webbrowser das Java-Applet aufgerufen und der Remote Desktop benutzt werden.

Vorteile sind auch hier die breite Unterstützung von Tastaturlayouts, da diese über das Java-Applet geregelt werden.

60 TightVNC: http://www.tightvnc.com/

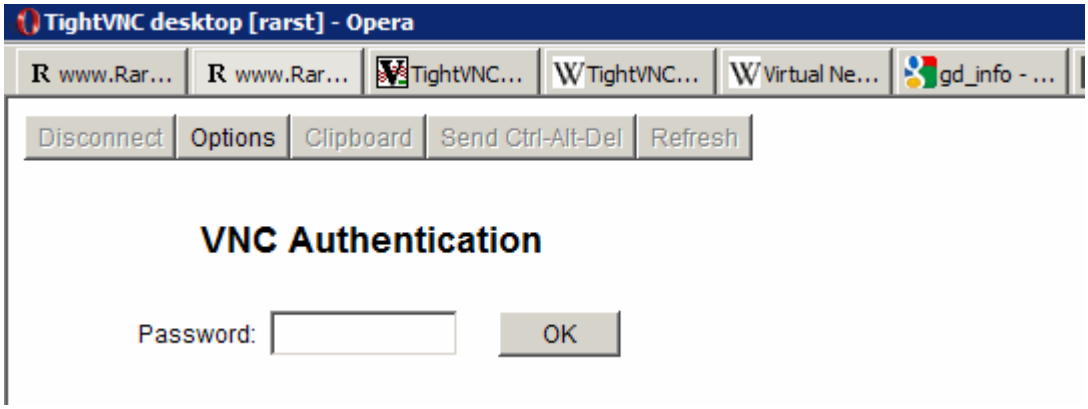

Abbildung 13: TightVNC mit Java Applet

Allerdings hat die Lösung von TightVNC auch einige Nachteile:

- komplette Abhängigkeit von Java
- TightVNC muss auf Client sowie Server laufen
- Der Webserver kann weder verschlüsselt werden noch kann das Java Applet normale HTTP-Authentifizierung

3.6.3 ThinVNC

Die von Cybele Software[61] entwickelte Anwendung ermöglicht den Zugriff auf Windows Remote Desktops, als HTML5 Applikation. ThinVNC[62] ermöglicht den Zugriff mittels SSL und Websockets. Auch Authentifizierung ist mit der Webanwendung möglich.

Abbildung 14: Funktionsweise von ThinVNC

61 Cybele Software: http://www.cybelesoft.com/
62 ThinVNC: http://www.cybelesoft.com/thinvnc/web-vnc.aspx

ThinVNC erfüllt alle Voraussetzungen die benötigt werden und kann sogar noch mehr, wie Dateitransfer und Screen-Sharing. Für Entwickler steht auch ein SDK bereit, um eventuelle Anpassungen vornehmen zu können. Allerdings laufen auf dem virtuellen Servern nur Windows Betriebssysteme ab Windows XP.

Zudem ist ThinVNC kein Open-Source Produkt. Zwar bietet Cybele Software eine 30-tägige Testversion an, allerdings fallen danach Gebühren von mindestens 35$ pro Jahr an, je nach gewählter Version.

3.6.4 OnlineVNC

OnlineVNC[63] läuft wie auch *ThinVNC* nur auf Windows Systemen ab Windows Vista. Die Software ist allerdings keine HTML5 Anwendung, sondern setzt auf Adobe Flash auf. Die Hostmaschine wird komfortabel über eine Software konfiguriert, unterstützt Authentifizierung und bietet Schnittstellen für Drittanbieter wie TightVNC oder RealVNC.

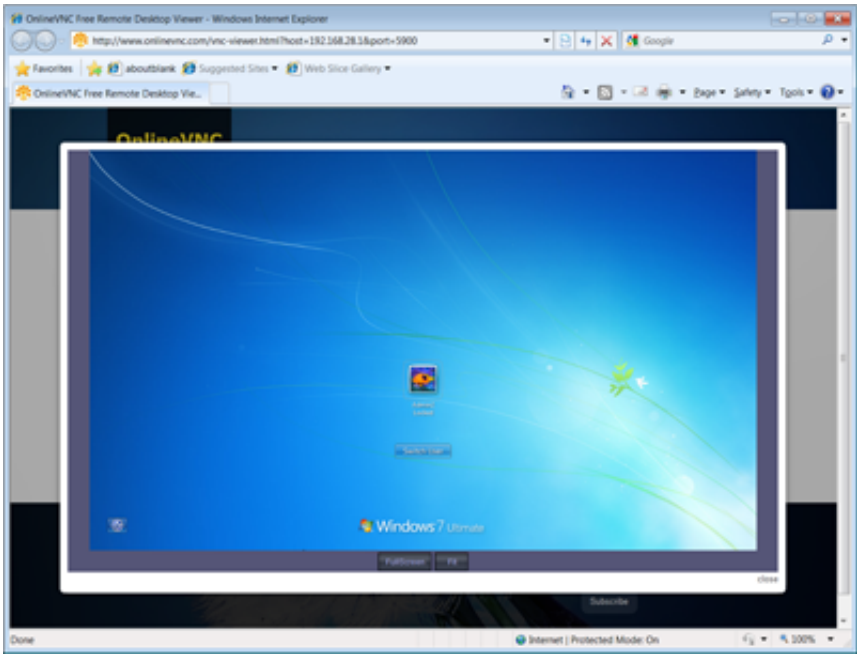

Abbildung 15: Screenshot von OnlineVNC

63 OnlineVNC: http://www.onlinevnc.com/

Durch die Verwendung von Flash, funktioniert der VNC Viewer in jedem Browser. Der Stream der zu steuernden Maschine wird ebenfalls in eine Canvas Fläche gezeichnet.

Nachteile sind neben der Verwendung von Flash auch das blocken der Applikation. Es lässt nur eine Verbindung zu. So ist es, zum Beispiel für Lernzwecke, nicht möglich mehrere Verbindungen zum selben Rechner herzustellen. Zudem ist der Ressourcenverbrauch größer als unter Linux KVM, aufgrund des Flash-Servers.

Ein Vorteil von OnlineVNC ist jedoch die völlig freie Verwendung. Es werden keine Gebühren oder ähnliches verlangt.

3.7 Konzeption und Design der Anwendung

Wie bereits in den *Anwendungsfunktionen* beschrieben, ist es das Ziel den Stream des VNC-Servers auf die Webseite zu zeichnen. Dies setzt eine Überprüfung des Benutzers voraus. Der VNC Server selbst muss einen offenen TCP-Port haben über den kommuniziert wird. Der Webserver, auf dem Django läuft, soll dabei unberührt bleiben und nur das Template bereitstellen sowie die Benutzerauthentifizierung durchführen. Um die Anwendung möglichst skalierbar zu lassen kommt als Bindeglied zwischen VNC-Server und der HTML5 Anwendung Node.js zum Einsatz.

3.7.1 Konzept

Abbildung 16 zeigt schematisch den Verbindungsaufbau der Anwendung und wie die Kommunikation zwischen Client (Webbrowser), Django, Node.js und dem VNC-Server funktioniert.

1. Der Client gibt eine Anfrage zum Öffnen seines VNC-Servers #2 ab.
2. Django prüft ob der VNC-Server zum Benutzer gehört und liefert als Antwort das Template zurück, mit der benötigten Websocket-Adresse.

3. Nach dem Laden des Templates öffnet sich direkt eine Websocket-Verbindung (`ws://`) zum Node.js Proxy. Hierbei werden Benutzer-Cookies mitgesendet (im WebSocket-Header).

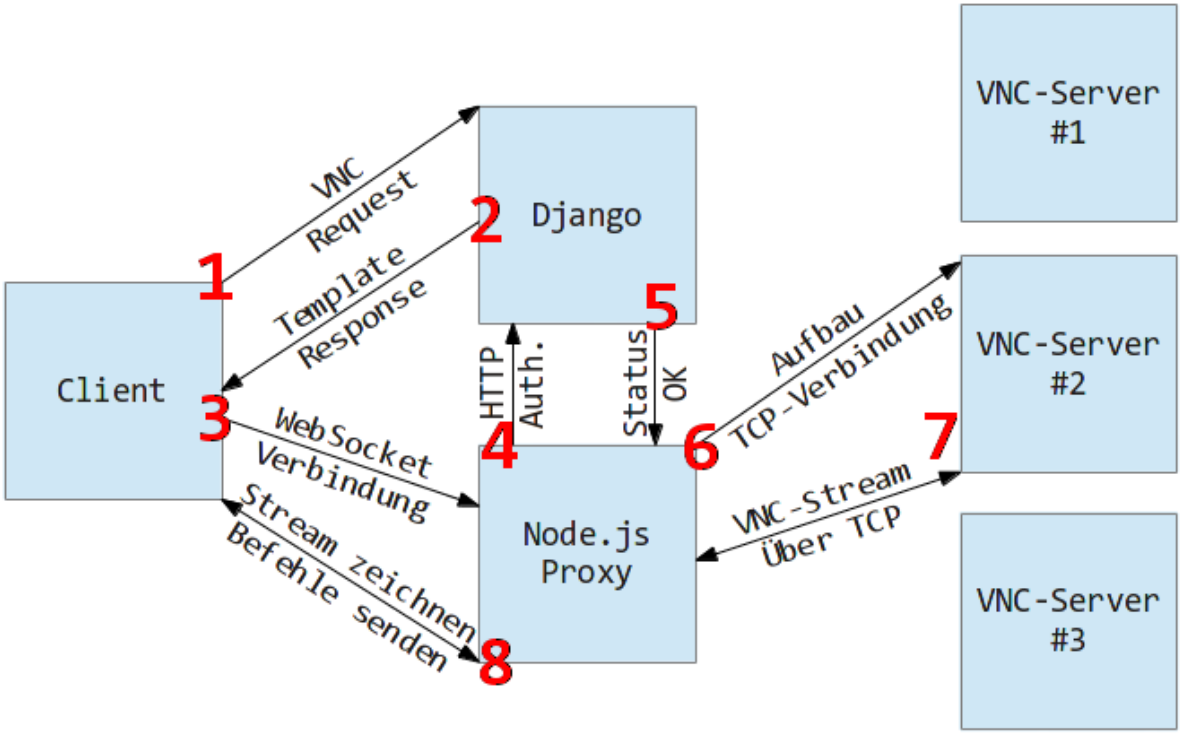

Abbildung 16: Schematische Darstellung der Anwendung

4. Der Node.js Proxy führt mit dem Cookie eine HTTP-Benutzerauthentifizierung durch.

5. Sollte diese korrekt sein, liefert Django, die benötigten VNC-Daten zum Aufbau der TCP-Verbindung an den Proxy zurück. Bei falschen Ergebnis gibt der Proxy dem Client eine Fehlermeldung zurück und die Verbindung wird geschlossen.

6. Es wird zum VNC-Server verbunden und zwischen TCP und WebSocket Binärdaten übertragen.

7. Node.js empfängt den Stream des VNC-Servers #2 über TCP und wandelt diese zur Übertragung mit WebSockets in base64 codierte Zeichenketten um.

8. Der Client empfängt die kodierten Zeichenketten und zeichnet diese in das Canvas-Element des Templates. Der Client kann nun Befehle mittels Tastatur oder Maus über die WebSocket-Verbindung absetzen, die über Node.js an den VNC-Server übertragen werden.

Dieses Konzept verdeutlicht die Schwerpunkte der Implementierung. *Abbildung 17* zeigt das dazugehörige Sequenzdiagramm um den Programmablauf besser verstehen zu können.

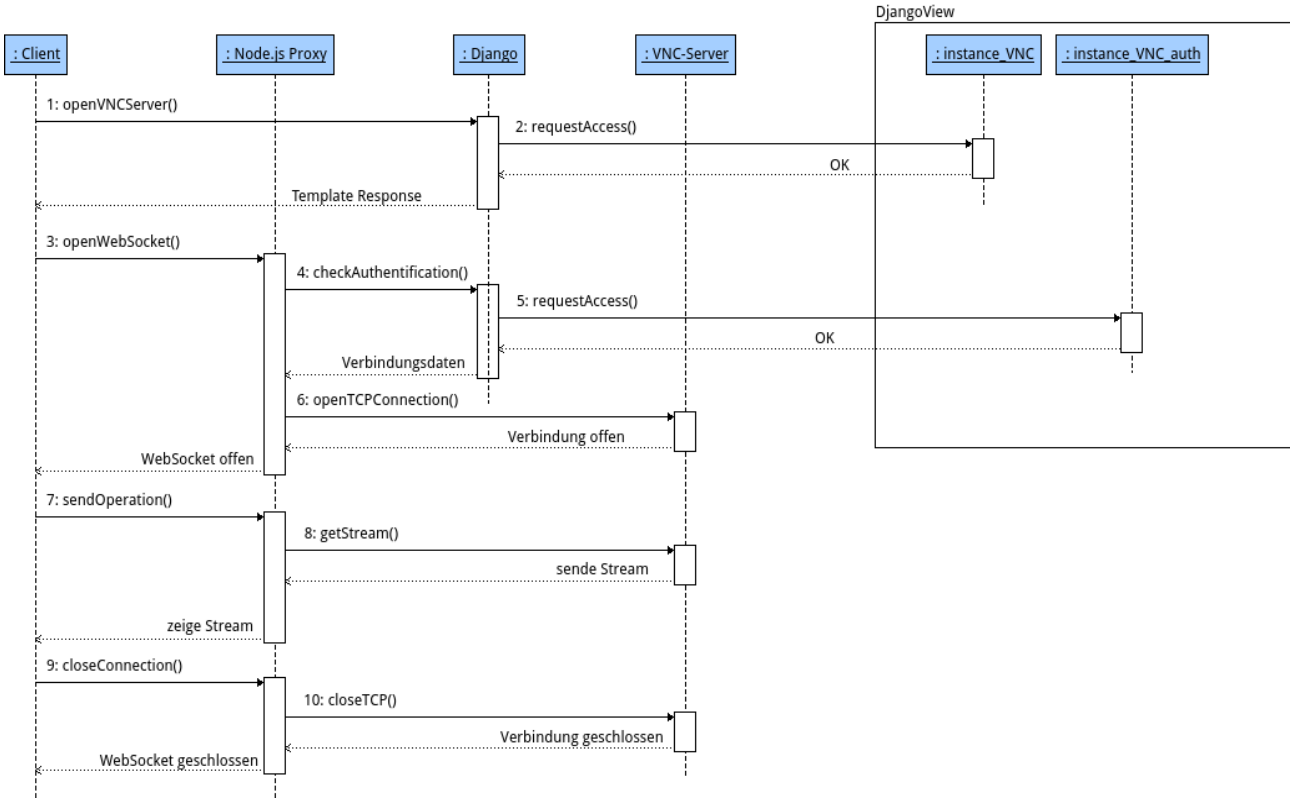

Abbildung 17: Sequenzdiagramm

Es wird nun versucht diese Schwerpunkte in der Implementierung kurz zusammenzufassen und diese zu erläutern. Im Kapitel *Realisierung* wird anschließend gezeigt wie dieses Konzept in das Webportal integriert wurde.

Django

Im Webportal müssen zwei neue Views integriert werden, die beide eine Überprüfung des Benutzers durchführen und die benötigten Informationen wie Verbindungsdaten zurückgeben.

Bei falschem, bzw. nicht autorisiertem Zugriff, muss eine entsprechende Fehlermeldung/Fehlerseite zurück geliefert werden.

Client / Template

Im Template muss das noVNC Paket integriert und sauber in das Design eingepasst werden. Zudem müssen Funktionen angepasst und die WebSocket-Verbindung direkt nach dem Laden der Seite geöffnet werden.

Das Template selbst wird von Django ausgeliefert, der Inhalt wird aber über WebSockets gesteuert.

Bei falschen Daten oder fehlerhafter Benutzerauthentifizierung muss die Verbindung geschlossen werden und dem Benutzer eine entsprechende Information anzeigen.

Node.js Proxy

Dieser Proxy muss komplett erstellt werden, und dabei auf die Eigenheiten von Node.js geachtet werden. Der Proxy muss nicht blockend und Ereignis-basierend entwickelt werden um diesen skalierbar zu halten. Zudem muss das WebSocket-API implementiert werden und die Abwicklung der Benutzerauthentifizierung mittels Cookies abgebildet werden.

Zur Analyse soll ein Logger implementiert werden, der verschiedene Logging-Level auf der Konsole ausgibt.

4 Realisierung

In diesem Kapital werden technische Details zur Realisierung erklärt und einige Quelltext und API Beispiele aufgeführt. Die Code Beispiele stellen Ausschnitte der Implementierung dar. Quelltextkommentare und `import`-Statements sind dabei ausgelassen. Für die vollständige Beschreibung der Realisierung sollte die API-Dokumentation, sowie der Quelltext selbst, verwendet werden.

4.1 Allgemein

Dieser Abschnitt befasst sich mit Allgemeinen Änderungen am Webportal, die außerhalb der Implementierung des VNC-Viewers statt gefunden haben.

4.1.1 Django Static

Da bei Projektbeginn die Portalsoftware noch mit Django 1.2 entwickelt wurde, mussten ein paar Anpassungen vorgenommen werden um die Vorteile von Django 1.3 und Django 1.4 verwenden zu können. Darunter fällt auch die Verwendung von Django Static[64].

Der Vorteil von der Verwendung Django Static ist nicht nur Zukunftssicherheit bei Django Updates, sondern auch die Zentrale Verwaltung von JavaScript-, CSS-, und Bilddateien. Das Modul `django.contrib.staticfiles` sammelt alle statischen Dateien von jeder Applikation innerhalb des Projekts und speichert diese an einer zentral festgelegten Stelle. Ein weiterer Vorteil ist die einfache Handhabung im Livebetrieb der Webseite um die statischen Dateien auszuliefern.

Um diese Methodik in Django verwenden zu können, müssen Anpassungen an der `settings.py` vorgenommen werden, die in *Listing 7* gezeigt werden.

64 Django staticfile app: https://docs.djangoproject.com/en/1.4/ref/contrib/staticfiles/

```
STATIC_ROOT = os.path.join(os.path.dirname(os.path.dirname(__file__)),
'htdocs', 'static')


STATIC_URL = '/static/'


STATICFILES_DIRS = (
  os.path.join(os.path.dirname(os.path.dirname(__file__)), 'static'),
)


STATICFILES_FINDERS = (
  'django.contrib.staticfiles.finders.FileSystemFinder',
  'django.contrib.staticfiles.finders.AppDirectoriesFinder',
)
```

Listing 7: settings.py - Einbindung von staticfiles

`STATIC_ROOT`

Legt den zentralen Speicherort der statischen Dateien fest.

`STATIC_URL`

Die URL unter welche diese Dateien ausgeliefert werden.

`STATICFILES_DIR`

Ort der neben den Django Apps nach Dateien durchsucht werden. Meistens genutzt für allgemeine JavaScript und CSS Dateien, die auf der gesamten Seite verwendet werden.

`STATICFILES_FINDERS`

Methoden die für das Suchen der Dateien verantwortlich sind. Hier wird das Dateisystem (`STATICFILES_DIR`) durchsucht und die App-Ordner. In den Applikation müssen diese Dateien aber zwingend in einem `static`-Ordner liegen.

Um in den Templates nun den Vorteil der staticfiles nutzen zu können, reicht es bei jeder URL zu einer statischen Datei ein `{{ STATIC_URL }}` einzufügen. Django ersetzt die Adresse mit der in der `settings.py` festgelegten.

```
<link
    rel="stylesheet" type="text/css" media="all"
    href="{{ STATIC_URL }}lib/css/layout.css" />
```

Listing 8: Verwendung von staticfiles in den Templates

Sollte das Portal noch im Debug Modus laufen, sind keine weiteren Schritte notwendig. Im Livebetrieb jedoch muss Django erst noch gesagt werden das die statischen gesammelt werden sollen. Dies lässt sich durch durch folgenden Management-Command erledigen:

```
$ python manage.py collectstatic
```

Dieses Kommando durchsucht alle statischen Dateien in den Apps und in den festgelegten Ordnern in `STATICFILES_DIR` und speichert diese an der festgelegten Zentrale Stelle im Dateisystem (`STATIC_ROOT`).

4.1.2 CSS Dateien splitten

Um effektiver Arbeiten zu können und später die Wartbarkeit zu erhöhen ist es notwendig geworden die CSS Dateien zu splitten und diese sinnvoll aufzuteilen.

Der Nachteil dieser Methode jedoch ist das bei einem Seitenaufruf mehrere Request abgesetzt werden, anstatt nur einen. Dieses Problem lässt sich jedoch mit dem Python Modul webassets[65] umgehen.

Aufteilung der CSS Dateien:

- `normalize.css`

  Normalize.css[66] ist eine Projekt von Nicolas Gallagher and Jonathan Neal. Die CSS-Datei versucht ein möglichst einheitliches Aussehen verschiedener HTML-Elemente in allen Browsern bereit zu stellen.

65 Webassets: http://elsdoerfer.name/docs/webassets/
66 Normalize.css http://necolas.github.com/normalize.css/

- `layout.css`

  Alle Elemente die für die Aufteilung der HTML-Struktur wichtig sind werden in dieser Datei bearbeitet.

- `forms.css`

  Das Styling von Form Elementen wie `<input />` und `<button />` befindet sich hier.

- `helper.css`

  Stellt Hilfsklassen dar, zum Beispiel für Image Replacement[67], clearfix[68] oder ein Print-Layout.

Nach der Aufteilung dieser Dateien ist es im Livebetrieb jedoch wichtig alle Dateien wieder in eine komprimierten Datei zusammen zu kommen lassen. Wie erwähnt hilft uns hierbei das Python Modul webassets. Die Anbindung ist äußerst einfach, da webassets direkt eine Django App mitliefert.

Nach der Installation von webassets muss lediglich in der `settings.py` unter `INSTALLED_APPS` die Applikation `'django_assets'` eingetragen werden.

```
from django_assets import Bundle, register
css = Bundle(
    'lib/css/normalize.css',
    'lib/css/layout.css',
    'lib/css/forms.css',
    'lib/css/helper.css',
    filters='cssmin',
    output='lib/css/_packed.css',
)
register('css_base', css)
```

Listing 9: Komprimieren von mehreren CSS Dateien mittels webassets

67 Image Replacement: http://css-tricks.com/css-image-replacement/
68 CSS Clearfix: http://www.positioniseverything.net/easyclearing.html

Anschließend muss eine `assets.py` in einem Applikationsverzeichnis erstellt werden, die ähnlich wie die `admin.py` automatisch geladen wird. In dieser Datei wird ein so genanntes „Asset“ festgelegt, das alle CSS (oder JavaScript) Dateien enthält und einen Ort in dem die komprimierte neue Datei gespeichert werden soll. *Listing 9* zeigt die Verwendung der assets.py. Dabei entsteht die neue, komprimierte CSS Datei _packed.css, die wieder rum von den staticfiles gelesen und verwendet wird.

```
{% load assets %}

<link
    rel="stylesheet" type="text/css" media="all"
    href="{{ ASSET_URL }}" />
```

Listing 10: Benutzung von Assets im Template

Um die neu entstandene Datei nutzen zu können kann das Codebeispiel aus Listing 8 wie folgt umgeschrieben werden.

4.2 Änderungen in den Views

Wie im Sequenzdiagramm (*Abbildung 17*) bereits ersichtlich müssen zwei neue Views innerhalb von der Django Applikation erstellt werden, die erstens eine Überprüfung der Benutzerzugehörigkeit zur virtuellen Maschine macht und zweitens benötigte Verbindungsdaten zum Proxy sowie zur VM zurück gibt. Die Applikation im Projekt wurde „`cloud`“ genannt, die dazugehörige Views findeen sich in „`views/vnc.py`“.

4.2.1 Funktion zum Zugriff auf den Proxy

Listing 11 zeigt die Implementierung für den Zugriff auf die Instanz der virtuellen Maschinen. Der View wird mit dem Parameter der Instanz-ID aufgerufen (URL). Der Dekorator `login_required` prüft bereits beim Request ob der Benutzer angemeldet ist,

wenn nicht leitet er auf Login-Seite weiter. Bei angemeldeten Benutzer jedoch wird die Instanz mittels der ID geladen und direkt geprüft ob diese zum Benutzer gehört. Im Erfolgsfall werden die Verbindungsdaten (`node_host` & `node_port`) ans Template geliefert. Wenn die Instanz jedoch nicht dem Benutzer gehört wird diesem eine 404-Fehlerseite zurückgeliefert. Die Prüfung ist notwendig um nicht per direkten URL Aufruf auf andere Instanzen zugreifen zu können, die dem Benutzer gar nicht gehören.

Der Token (`token`) der mit ans Template geliefert wird, enthält den Session Key. Dies stellt eine Fallback Lösung dar, falls ein Webbrowser das Cookie mit dem Session Key nicht mit der WebSocket-Verbindung mitliefern sollte.

```
@login_required
def instance_vnc(request, slug):
    try:
        i = Instance.objects.get(pk=slug, owner=request.user).pk
    except Instance.DoesNotExist:
        raise Http404
    node_host = settings.NODE_HOST
    node_port = settings.NODE_PORT
    token = request.session.session_key
    return render_to_response('limeade_cloud/instance_vnc.html', {
        'id': i, 'host': node_host,
        'port': node_port, 'token': token
    }, context_instance=RequestContext(request))
```

Listing 11: Ausschnitt der Implementierung

Das Mitliefern des Tokens muss sein, da ältere Browsergenerationen noch nicht das neueste WebSocket-API unterstützen. Um diesen jedoch keine Einschränkungen zu liefern, wurde der Token als Fallback Lösung umgesetzt.

4.2.2 API-Call zur VM

Der API-Call aus *Listing 12* wird verwendet um den Benutzer über Websockets zu

Authentifizieren und die benötigten Verbindungsdaten zum Aufbau der TCP-Verbindung an die virtuelle Maschine zurück zu liefern.

Zuerst wird die Session des User überprüft (`token`) und ob dieser auch Berichtigung hat die VM zu öffnen (`slug`). Falls dies nicht der Fall sein sollte wird als Status Code 403 („Permission Denied“) zurückgeliefert und der Node.js Proxy wird dies auch so ans Template zurückleiten.

```
def instance_vnc_auth(request, slug, token):
    status_code = 200 # default status code
    vnc_port = '5900' # default port
    host = None
    try:
        s = Session.objects.get(pk=token)
        user_id = s.get_decoded().get('_auth_user_id')
        user = User.objects.get(pk=user_id)
        i = Instance.objects.get(pk=slug, owner=user)
    except ObjectDoesNotExist:
        status_code = 403
    if i:
        host = urlparse(i.node.uri).netloc
    try:
        c = libvirt.open(i.node.uri)
        dom = c.lookupByName(i.domain)
        xml = dom.XMLDesc(0)
        vnc_port = fromstring(xml).xpath(
            '../domain/devices/graphics[@type="vnc"]'
        )[0].attrib['port']
    except:
        status_code = 500
    data = {'host': host, 'port': int(vnc_port)}
    return HttpResponse(simplejson.dumps(data),
        mimetype='application/json', status=status_code)
```

Listing 12: Implementierung des API Calls

Bei Erfolg wird der Hostname der VM aus der Datenbank gelesen (`host`) und libvirt fragt die Daten der virtuellen Maschine ab. Um den Port des virtualisierten Rechners zu bekommen, muss allerdings die XML-Datei von QEMU/KVM eingelesen werden.

In der XML-Datei stehen jegliche Informationen der VM zur Verfügung. Interessant für den Verbindungsaufbau ist aber wie erwähnt nur der VNC-Port. Falls die XML-Datei nicht geparst werden kann, oder andere Fehler in der libvirt-API auftreten wird als Status Code „Internal Server Error“ (500) zurück gegeben und vom Proxy weiter behandelt.

Die zurückgegebenen Daten sind im JSON Format[69], um die Verarbeitung in JavaScript einfacher zu halten.

Als zusätzliche Bibliotheken zur Verarbeitung des Request werden wie erwähnt `libvirt` und als XML Parser `fromstring` eingesetzt. Django selbst bringt zur Verarbeitung von JSON direkt eine Bibliothek mit (`simplejson`).

4.3 Templates

Die Anpassungen am Template gehört eigentlich zu Django selbst, da Django für die Auslieferung des Templates verantwortlich ist. Hier muss aber unterschieden werden zwischen den Daten die Django fürs Template bereithält und dem noVNC Paket, das hier integriert und angepasst werden muss.

4.3.1 Aufbau

Das Basis-Template ist in mehre Blöcke unterteilt, um keine wiederholende Stellen im Template zu haben und eine sinnvolle Struktur aufweisen zu können. Auch hier schlägt sich das Prinzip des „Don't repeat yourself“ durch.

69 JSON: http://www.json.org/

```
<!DOCTYPE html>
<head>
    <title>{% block title %}Limeade{% endblock %}</title>
    {% block metatags %}{% endblock %}
    {% block stylesheets %}{% endblock %}
    {% block extrahead %}{% endblock %}
</head>
<body class="{% block bodyclass %}{% endblock %}">
    <header id="branding" role="banner"></header>
    <div id="main" class="clearfix">
        <nav id="navigation" role="navigation"></nav>
        <div id="content" role="main">
            {% block content %}{% endblock %}
        </div>
    </div>
    {% block extrajs %}{% endblock %}
</body>
</html>
```

Listing 13: Auszug des Basis Templates (base.html)

Wichtige Blöcke sind:

- `{% block stylesheets %}`

  Hier finden alle Stylesheet-Dateien des Webportals Platz. Jedoch nur dann wenn diese auch im jeweiligen Template gebraucht werden. Diese Methode spart zusätzliche Requests.

- `{% block content %}`

  Der sich, von Template zu Template, wechselnde Inhalt wird in diesem Block notiert.

- `{% block extrajs %}`

  Alle weiteren, nicht auf dem gesamten Portal verwendeten JavaScript Dateien oder

Funktionen werden in diesem Block notiert. In diesem Bereich findet die Hauptsächliche HTML5 Applikation Platz.

Eine vereinfachte Darstellung des Basis Templates ist in *Listing 14* dargestellt. Hier fehlen jedoch alle Angaben zu den verwendeten Metatags, Stylesheets und JavaScript Dateien. Es zeigt jedoch den grundlegenden Aufbau der Webseite.

Um dieses Template als Grundlage für den VNC-Viewer zu nehmen, reicht es, im neuen Template die Basis mittels `{% extends "base.html" %}` einzubinden.

4.3.2 Integration von noVNC

Dieser Abschnitt befasst sich mit der Einbindung des noVNC HTML5 Clients ins Template und erläutert die Herangehensweise sowie nötige Anpassungen der Bibliothek ans Projekt.

```
{% extends "base.html" %}
{% load i18n %}
{% block content %}
    <h1>{% trans "VNC" %}</h1>
    ...
    <div id="screen">
        <div id="screen_pad"></div>
        <div id="container">
            <canvas id="canvas" width="640px" height="20px">
                {% trans "Canvas not supported." %}
            </canvas>
        </div>
    </div>
{% endblock %}
...
```

Listing 14: VNC Template Ausschnitt der den Canvas Bereich zeigt

Zunächst mal muss der Client von der Projektseite heruntergeladen und entpackt werden.

Der `include` Ordner von noVNC wird im `static` Verzeichnis des Projekts eingegliedert. Dieser Ordner enthält alle, für den Client notwendige JavaScript Dateien um einen über WebSockets den Stream von einer virtuellen Maschine zu empfangen und anzuzeigen. Im Template `instance_vnc.html` müssen nun ein paar Anpassungen gemacht werden und den Pfad des `include` Ordners angegeben werden.

Listing 14 zeigt den Canvas Bereich des Template in dem später der VNC-Stream gezeichnet wird. Die Größenangaben sind nur für den Internet Explorer interessant, da dieser den Canvas Bereich sonst nicht zeichnen würde. Die Größe des Bereichs passt sich bei aktiven Stream automatisch an die Auflösung der virtuellen Maschine an.

```
{% block extrajs %}
    <script type="text/javascript">
        INCLUDE_URI = '{{ STATIC_URL }}lib/js/novnc/include/';
        TOKEN = '{{ token }}';
        INSTANCE_ID = '{{ id }}';
        HOST = '{{ host }}';
        PORT = '{{ port }}';
    </script>
    <script type="text/javascript"
        src="{{ STATIC_URL }}lib/js/novnc/include/vnc.js">
    </script>
    <script type="text/javascript"
        src="{{ STATIC_URL }}lib/js/novnc/start.js">
    </script>
    <script type="text/javascript">
        window.onload = START.load;
    </script>
{% endblock %}
```

Listing 15: instance_ vnc.html Auszug der Einbindung des noVNC Pakets

Im Block `extrajs` wird, wie bereits im Aufbau des Templates erwähnt, alle weiteren JavaScript Dateien eingebettet. *Listing 15* Stellt diese Einbindung dar und zeigt die vom Django View aus Kapitel *4.2.1* zurückgegebenen Werte. Diese werden in den inkludierten

JavaScript Dateien weiter behandelt. Zu erwähnen ist, das diese globale Variablen darstellen und somit auch besonderes geprüft werden müssen.

`INCLUDE_URI`

Enthält den Pfad zum noVNC Verzeichnis.

`TOKEN`

Der Session Key des Users, wird als Fallback benutzt falls der Browser kein Cookie im WebSocket-Handshake mitliefert.

`INSTANCE_ID`

Die ID der Instanz auf die der User zugreifen möchte.

`HOST`

Hostname des Node.js Proxys.

`PORT`

Port des Node.js Proxys,

Die JavaScript Datei `vnc.js` lädt alle benötigten Bibliotheken aus dem `include` Verzeichnis. Die Datei `start.js` wurde neu entwickelt und stellt die Funktionen der Anwendung bereit, sowie den direkten Aufbau der WebSocket Verbindung.

```
load: function() {
    START.rfb = RFB({
        'target': $D('noVNC_canvas'),
        'onUpdateState': START.updateState,
        'onClipboard': START.clipReceive
    });
    START.connect();
}
```

Listing 16: Laden des Remote Framebuffer Protocols und Verbindungsaufbau

Nach dem Laden der Seite (`window.onload`) wird die Funktion `load` aufgerufen die das Remote Framebuffer Protocol initialisiert (*Listing 16*).

Dem Remote Framebuffer Protocol werden auch Funktionen mitgegeben die das Darstellen des Streams und die Einbindung der Zwischenablage regeln.

Weitere Aufgaben der Funktion `load`:

- Speichern der Standardwerte (Verbindungs-Timeout, True Color, usw.), diese Werte können über ein Einstellungsmenü aber jederzeit geändert werden.
- Sollte das zugreifende Gerät eine Touchbedienung haben (Smartphone/Tablet) werden spezielle Funktionen, wie Display Tastatur, geladen.

Die Methode `connect` hingegen regelt den WebSocket Verbindungsaufbau.

4.4 Node.js Proxy

Die Funktionen des Proxys wurden bereits im *Konzept* beschrieben. Dieser Abschnitt beschreibt die Implementierung der kompletten Node.js Anwendung. Die Anwendung selbst soll mit den Node.js Version 0.6 und 0.8 funktionieren.

4.4.1 Verwendete Module

Node.js bringt im Kern einige Module mit die in den jeweiligen Anwendungen verwendetet werden können. Um Programme klein zu halten müssen diese explizit geladen werden, genauso wie externe Module. Durch den Node Packet Manager (NPM) können diese speziell für eine Anwendung installiert werden.

Der Proxy benötigt neben einigen Core-Modulen noch ein Modul um WebSockets unterstützen zu können und, als Fallbacks für ältere Browser, `policyfile`, welches Flash Anwendungen absichert. Da noVNC einen Flash-Fallback mitbringt der WebSockets emuliert wird diese Modul benötigt.

```
var WebSocketServer = require('ws').Server;
var policyfile      = require('policyfile');
var Buffer          = require('buffer').Buffer;
var http            = require('http');
var url             = require('url');
var net             = require('net');
```

Listing 17: Laden der nötigen Module

Das installieren der externen Module geht, wie bereits erwähnt mittels NPM, hier reicht ein einfaches

```
$ npm install ws policyfile
```

dabei werden auch alle Abhängigkeiten aufgelöst und gegebenenfalls mit installiert.

- `ws`

  WebSocket Modul, das die aktuellste Protokoll Version implementiert hat und laut dem Autobahn Test die schnellste Implementierung darstellt.

- `policyfile`

  Sichert Flash Anwendungen ab, in dem diese sich durch eine einen bestimmten Port mit dem Policyfile authentifizieren müssen.

- `buffer`

  Kodiert und Dekodiert Strings, wird zur Übertragung des VNC-Streams verwendet.

- `http`

  Stellt einen HTTP-Server bereit.

- `url`

  Funktionen zur Manipulation und Verarbeitung von URLs.

- `net`

  Stellt Verbindungen zu anderen Servern her. Unterstützt werden TCP, HTTP und HTTPS.

4.4.2 Logging

Um Zugriffe, Fehler und Änderungen geschickt überwachen zu können, soll die Anwendung ein Logging-Modul enthalten. Diese soll verschiedene Logging Stufen unterstützen.

```
var Logging = {};
Logging._log_level = 'debug';
Logging.init_logging = function(level) {
    if (typeof level === 'undefined') {
        level = Logging._log_level;
    } else {
        Logging._log_level = level;
    }
    Logging.Debug = Logging.Info = Logging.Warn = Logging.Error =
function(msg) {};
    switch (level) {
        case 'debug':
            Logging.Debug = function(msg) { console.log(msg); };
        case 'info':
            Logging.Info = function(msg) { console.log(msg); };
        case 'warn':
            Logging.Warn = function(msg) { console.warn(msg); };
        case 'error':
            Logging.Error = function(msg) { console.error(msg); };
        case 'none':
            break;
        default:
            throw("invalid logging type '" + level + "'");
    }
};
Logging.get_logging = function() {
    return Logging._log_level;
};
Logging.init_logging();
```

Listing 18: Implementierung von verschieden Logging Stufen

- `Debug`

  Ausgabe aller Informationen, die bei jeder Funktion auftreten.

- `Info`

  Nur mitloggen wichtiger Statusänderungen

- `Warn`

  Ausgabe von Informationen die nicht im Programmablauf vorgesehen sind, jedoch keine Auswirkungen auf die Laufzeit haben.

- `Error`

  Mitloggen im Fehlerfall.

4.4.3 Proxy Handling

Dieser Abschnitt beschreibt das Handling zwischen der TCP und der WebSocket-Verbindung.

```
function connectTarget(host, port, callback) {
    try {
        var vncSocket = net.createConnection(port, host);
        Logging.Info('Client connected to
            VNC server on ' + host + ':' + port);
        Logging.Info('Start proxying from ' + serverOptions.host +
            ':' + serverOptions.port + ' to ' + host + ':' +
            port + "\n");
        callback(vncSocket);
    } catch (e) {
        Logging.Error('Could not create connection to VNC: ' +
            e.message);
    }
}
```

Listing 19: Herstellen der Verbindung zum VNC-Server

Es wird zuerst eine Verbindung zum VNC-Server hergestellt (*Listing 19*), und anschließend

die WebSockets-Events implementiert, die zur Kommunikation verwendet werden.

```
function handleProxy(ws, vncSocket) {
    vncSocket.on('begin', function() {
        Logging.Debug('Connected to target');
    });
    vncSocket.on('data', function(data) {
        ws.send(new Buffer(data).toString('base64'));
    });
    vncSocket.on('end', function() {
        ws.close();
        Logging.Warn('VNC Target disconnected');
    });
    ws.on('message', function(msg) {
        vncSocket.write(new Buffer(msg, 'base64').toString('binary'),
            'binary');
    });
    ws.on('close', function(code, reason) {
        vncSocket.end();
        Logging.Warn('WebSocket client disconnected: ' + code + ' [' +
            reason + ']');
    });
    ws.on('error', function(e) {
        Logging.Error('WebSocket client error: ' + e);
    });
}
```

Listing 20: Implementierung des Proxy Handlings

`vncSocket.on('begin')`

Event bei Herstellung einer neuen Verbindung zum VNC Server.

`vncSocket.on('data')`

Sendet Daten zum noVNC-Client.

`vncSocket.on('end')`

Ereignis bei Beenden der Verbindung zur WebSocket-Verbindung.

`ws.on('message')`

Sendet Daten zum VNC-Server.

`ws.on('close')`

Beendet die Verbdinung zum VNC-Server.

`ws.on('error')`

Falls Fehler während der Verbindung auftreten, werde diese geloggt.

4.4.4 Authentifizierung

Bevor jedoch eine Verbindung hergestellt werden kann, muss der Benutzer überprüft und authentifiziert werden, dies gelingt durch das mitgelieferte Cookie, oder den Token. Dabei wird der API Call aus Abschnitt *4.2.2* aufgerufen.

```
function extractParams(header, callback) {
    var path        = header.url;
    var cookie      = header.headers.cookie.toString().split("; ");
    var params      = url.parse(path, true, false);
    var instance_id = params.query.instance_id;
    var s_id  = '';
    for (var i = 0; i < cookie.length; i++) {
        var loc = cookie[i].search(/session/);
        if (loc >= 0)
            s_id = cookie[i].toString().replace('sessionid=', '');
    }
    if (s_id == '') {
        Logging.Error('No Session ID found in cookie! Use token');
        s_id = params.query.token;
    }
    callback(instance_id, s_id);
}
```

Listing 21: Extrahieren der Session

Listing 21 extrahiert die Session ID aus dem mitgelieferten Cookie, oder nutzt den Token, falls das Cookie keine Session enthält. Anschließend wird die Instanz und die Session zurückgegeben.

```
function checkValidation(instance_id, session, callback) {
    var req = http.request(options, function(res) {
        if (res.statusCode == 200) {
            res.setEncoding('utf8');
            res.on('data', function(chunk) {
                var connectionData = JSON.parse(chunk);
                var host = connectionData.host;
                var port = connectionData.port;
                callback(host, port);
            });
        } else {
            Logging.Warn('Permission denied. Status Code: ' +
                res.statusCode);
            try {
                res.writeHead(res.statusCode, {
                    'Content-Type': 'text/plain',
                    'Access-Control-Allow-Origin': '*'
                });
                res.write('Permission denied.');
                res.end();
            } catch (e) {
                Logging.Error('Error: ' + e.message);
            }
        }
    });
    req.on('error', function(e) {
        Logging.Error('Problem with request: ' + e.message);
    });
    req.end();
}
```

Listing 22: API-Call im Proxy

Die zurückgelieferten Daten werden in der Funktion `checkValidation` (*Listing 22*) verarbeitet. Durch den API-Call zum Django View wird überprüft ob der Benutzer zugriff auf die VM hat und im Erfolgsfall der Host und Port des VNC-Servers zurückgeliefert. Falls der Zugriff verweigert wurde, gibt der Proxy eine Meldung zurück ans Template.

4.4.5 HTTP- und WebSocket Server

Der letzte Teil der Anwendung stellt einen HTTP-Server bereit der auf einem Port lauscht, in diesem wird ein WebSocket-Server gestartet. Sobald die Verbindung zum WebSocket Server aufgebaut ist (*Listing 23*), werden die einzelnen Funktion durchlaufen und somit eine Verbindung zum VNC-Server hergestellt. Ebenso wird das Flash Policyfile angehängt.

```
httpServer.listen(ops.port, ops.host, function() {
    Logging.Info('Server is running on ' + ops.host + ':' + ops.port);
    var wsServer = new WebSocketServer({
        server: httpServer
    });
    policyfile.createServer().listen(-1, httpServer, function() {
        Logging.Info('Flash policyfile attached.');
    });
    wsServer.on('connection', function(ws) {
        Logging.Info('New client tries to connect.');
        extractParams(ws.upgradeReq, function(path, session) {
            checkValidation(path, session, function(host, port) {
                connectTarget(host, port, function(vncSocket) {
                    handleProxy(ws, vncSocket);
                });
            });
        });
    });
});
```

Listing 23: Erstellen des WebSocket-Server

4.5 Ergebnis

Das Zusammenspiel zwischen Django, dem VNC-Server und Node.js ist nun implementiert und laut Test mittels Apache Benchmark (ab) schafft die Anwendung mehr als 10.000 gleichzeitige Verbindungen.

Beispiel:

```
$ ab -n 10000 -c 5 http://example.com/
```

- -n: Anzahl der Requests
- -c: Anzahl der Parallelen Requests

Die Anwendung läuft auf Smartphones, Tablets und Desktop Rechnern, durch die automatische Skalierung des View Ports. Für mobile Endgeräte wurde eine virtuelle Tastatur, sowie eine Maus geschaffen.

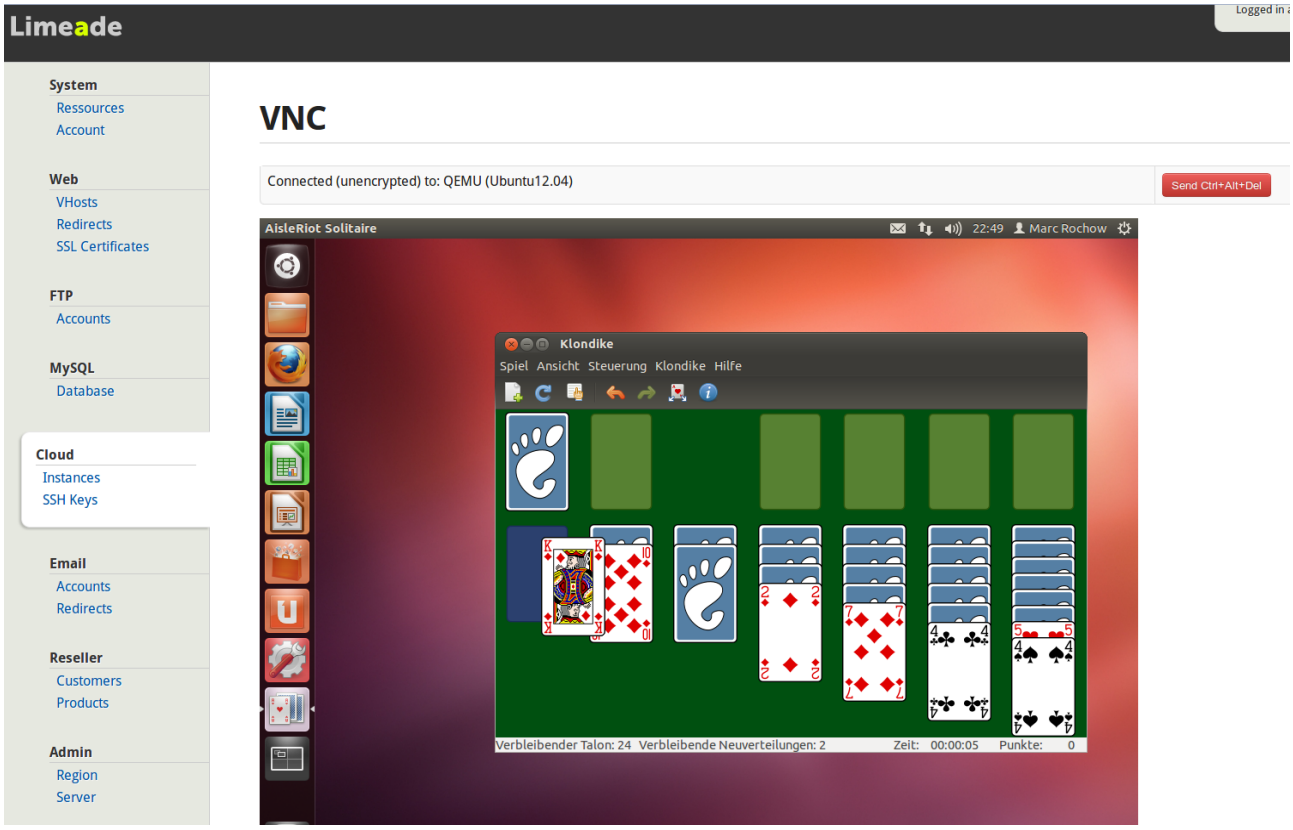

Abbildung 18: Screenshot der fertigen Anwendung

5 Fazit & Ausblick

6 Literaturverzeichnis

[marketshare12] **Marketshare**: Mobile Trend, 2012
http://marketshare.hitslink.com/mobile-market-share?qprid=61&qpct=5&qptimeframe=M&qpsp=139&qpnp=25,
aufgerufen am 22.06.2012

[ramsey05] **Ben Ramsey**: Framing the Frameworks - What Are They and Do I Need One? 2005. Seite 1-5
http://files.benramsey.com/talks/2005/ipcse/Frameworks.pdf,
aufgerufen am 05.06.2012

[ford09] Neal Ford: Produktiv programmieren. O-Reilly, 2009.
Ausgabe 1, ISBN: 978-3-89721-886-4. Seite 6

[django05] **Django Software Foundation**: FAQ: General, 2005
https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names,
aufgerufen am 20.05.2012

[holovatykaplanmoss07] **Adrian Holovaty, Jacob Kaplan-Moss**: The Definitive Guide to Django: Web Development Done Right. Apress, 2007.
Ausgabe 1, ISBN: 978-1590597255. Seite 59-82
http://www.djangobook.com/en/2.0/chapter05/,
aufgerufen am 20.05.2012

[heise10] **heise online**: Chromes "Kurbelwelle" optimiert JavaScript zur Laufzeit. Heise Verlag, 2010.
http://heise.de/-1149365,
aufgerufen am 17.06.2012

[ochs12] **Oliver Ochs**: Javascript auf dem Server mit Node. dpunkt Verlag, 2012. Ausgabe 1, ISBN: 978-3898647281. Seite 133 ff.

[geisendoerfer10] **Felix Geisendörfer**: Wie Node.js JavaScript auf dem Server revolutioniert: Schubrakete für JavaScript. T3N Magazin, 2010. http://t3n.de/magazin/nodejs-javascript-server-revolutioniert-schubrakete-226177/, aufgerufen am 06.06.2012

[herron11] **David Herron**: Node Web Development. Packet Publishing, 2011. ISBN: 978-1849515146. Seite 37 ff.

[bernerslee06] **Tim Berners-Lee**: Reinventing HTML, 2006. http://dig.csail.mit.edu/breadcrumbs/node/166, aufgerufen am 20.06.2012

[lilleybernerslee11] **Chris Lilley, Tim Berners-Lee**: HTML Working Group Charter, 2011. http://www.w3.org/2007/03/HTML-WG-charter#deliverables, aufgerufen am 20.06.2012

[weßendorf11] **Matthias Weßendorf**: WebSocket: Annäherung an Echtzeit im Web. heise Verlag, 2011. http://www.heise.de/developer/artikel/WebSocket-Annaeherung-an-Echtzeit-im-Web-1260189.html, aufgerufen am 22.06.2012

[ublkitamura12] **Malte Ubl, Eiji Kitamura**: Introducing WebSockets: Bringing sockets to the web, 2012. http://www.html5rocks.com/en/tutorials/websockets/basics/, aufgerufen am 23.06.2012

[lubbersgreco12] **Peter Lubbers, Frank Greco**: HTML5 Web Sockets: A Quantum Leap in Scalability for the Web, 2012. http://www.websocket.org/quantum.html, aufgerufen am 23.06.2012

[fulton11] **Steve Fulton**: HTML5 Canvas. O'Reilly Media, 2011. Ausgabe 1, ISBN: 978-1449393908. Seite 1-16

[barton08] **George Barton**: The Sassy part of the Cloud. Barton's Blog, 2008. http://bartongeorge.net/2008/12/18/the-sassy-part-of-the-cloud/, aufgerufen am 30.06.2012

[ibm10] **IBM**: Defining a framework for cloud adoption. IBM Corporation, 2010. ftp://ftp.software.ibm.com/common/ssi/pm/xb/n/cie03069usen/CIE03069USEN.PDF, aufgerufen am 30.06.2012

[yao10] **Yushu Yao**: Network Performance Test Xen/Kvm (VT-d and Para-virt drivers). Virtualization Studies, 2010. http://vmstudy.blogspot.de/2010/04/network-performance-test-xenkvm-vt-d.html, aufgerufen am 30.06.2012

[yaocpu10] **Yushu Yao**: CPU Performance Xen/Kvm. Virtualization Studies, 2010. http://vmstudy.blogspot.de/2010/04/cpu-performance-xenkvm.html, aufgerufen am 30.06.2012

[yaodisk10] **Yushu Yao**: Disk Performance Xen/Kvm with LVM and Para-virt drivers. Virtualization Studies, 2010. http://vmstudy.blogspot.de/2010/04/disk-performance-xenkvm-with-lvm-and.html, aufgerufen am 30.06.2012

[degelas08] **Johan De Gelas**: Hardware Virtualization: the Nuts and Bolts. AnandTech, 2008. http://www.anandtech.com/show/2480, aufgerufen am 01.07.2012

[roebuck11] **Kevin Roebuck**: Virtual Network Computing (Vnc): High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors. Emereo Pty Limited, 2011. ISBN: 978-1743047392. Seite 1-4

[cambrige02] **AT&T Laboratories Cambridge**: The RFB Protocol, 2002. http://www.cl.cam.ac.uk/research/dtg/attarchive/rfb.html, aufgerufen am 23.07.2012

[comsolit09] **comsolitAG**: HTML5 / CSS3, 2009. http://www.slideshare.net/Larz73/html5-css3, aufgerufen am 04.08.2012

[braun11] **Herbert Braun**: HTML5 - Der große Wurf. heise Verlag, 2011. Seite 6-11

[isolani06] **Isolani**: Tim Berners Lee and reinventing HTML , 2006. http://isolani.co.uk/blog/standards/TimBerners LeeAndReinventingHtml, aufgerufen am 04.08.2012

Anhang

A. Listings

B. Abbildungsverzeichnis

C. Abkürzungsverzeichnis