Bachelorthesis

Development of a Web-Application
for executing tasks on an IBM-Mainframe
in the context of Education

Bachelorarbeit gemäß § 17 der Allgemeinen Prüfungsordnung vom 01.08.2008
im Bachelorstudiengang Informationsmanagement und Unternehmenskommunikation
an der Hochschule für angewandte Wissenschaften Neu-Ulm

Erstkorrektor	Prof. Dr. Phillipp Brune
Betreuer	Dr. Kevin Henrichs
Verfasser	Marc Morschhauser (Matr.-Nr.: 204041)

Thema erhalten:	01.01.2020
Arbeit abgegeben	01.05.2020

Unterschrift des Studierenden

Unterschrift und Firmenstempel der Ausbildungsstelle

In association with:

1 Abstract

Hier kommt der/die/das Abstract.

Abbildungsverzeichnis

Inhaltsverzeichnis

2 Introduction

Big Data, Cloud, Blockchain, IoT - Digitalization is steadily proceeding and with it the amount of data that has to be processed every day. Many web-services at this point are dependent on flexibility in first place. Big server farms facilitate an unprecedented agility to operators of online-businesses referring to their resource-planning. Urgent situations with high data traffic can be handled by adding physical or also virtual servers within minutes. However there are services where minutes make the difference between millions of euros. Those services demand for information technology, which not only has a high data throughput but also ensures the highest availability. The IBM Mainframe in its newest construction provides an I/O of over 800GB per second while being available 99,999% of the time running. This makes the Mainframe a cornerstone in present finance-sector's IT-management. Due to it's security and it's high speed it also finds it's use in other big industries - thus also insurances, the aerospace industry or big retail companies benefit from this supercomputer.
But since the IT improved considerably over the last decades, the Mainframe was presumed dead and many companies discontinued training their staff to maintain this technology. Contrary to their expectations they are dependent on it till today and they will presumably be in the remote future. What is left is a big gap in the division of skilled professionals that is hard to close.
The Goethe-University in Frankfurt, Germany, attends to participate in closing this gap. Therefore the university procured one of the aforementioned IBM supercomputers to train their students dealing with it.

This Bachelor Thesis approaches a Front-End-Application, which will be running on a Linux Virtual-Machine but executing tasks on the mainframe's operating system called z/OS. By building this bridge between the Linux VM and the z/OS, students will be able to get in touch with the Mainframe located in Frankfurt University through a web-browser and as a consequence experience how the system is working.

The resources for this activity are allocated by the *Talentschmiede AG* - a Frankfurt located IT consultancy which is in close contact to the finance sector and knows and cares about the gap of skilled professionals. Further it is supported by the *Academic Mainframe Consortium e.V.* - an association founded by Mainframe Experts which are also willing to acquire new educated staff in the division of Mainframe.

3 Related Work

4 Forschungsfrage

5 Anforderungen

6 Development Environment

In this chapter, a development environment will be created. An Oracle Virtual-Machine called VirtualBox will be installed on MacOSX. In this virtual machine the open and free Linux-System Ubuntu 16.04 LTS will be running, to make it possible to implement an Apache HTTP-Server. The Apache Server makes it possible to develop a simple dynamic website offline on a localhost that can be accessed through any web-browser as long as the virtual machine is running.

Schematische Darstellung!!

7 Die Entwicklungsumgebung

In diesem Kapitel soll eine optimale Entwicklungsumgebung für das Projekt geschaffen werden. Hierfür wird ein Apache Web-Server innerhalb einer virtuellen Linux-Partition implementiert. Sowohl Linux als auch bei Apache handelt es sich um freie open-source Projekte.
-Kostenersparnis durch open-source projekte?!-
-Gründe für Linux / Apache ???!-

7.1 Einrichten einer Linux-VM

Passend für den Apache Web-Server ist die Ubuntu Server GUI.
Installiert wird diese in der Virtual-Box von Oracle.

Für dieses Vorhaben wird die Ubuntu Version Server 16.04.3 LTS herangezogen.

VirtualBox wird als Maschine verwendet, um den Linux-Server zu beherbergen.
Hier wird eine neue Virtuelle Maschine erstellt.
Dem Projekt wird der Name sandbox verliehen, was eine isolierte, von der Öffentlichkeit -zunächst- abgeschottete Umgebung impliziert. Zusätzlich wird hier bereits von Anfang an festgelegt, dass es sich um einen Linux und genauer um eine 64-Bit VM mit dem Ubuntu OS handeln soll.
Eine dynamische HDD Zuweisung sorgt dafür, dass die Servergröße nicht um Vorhinein festgelegt werden muss, sondern mitwächst, sollte mehr Speicherplatz benötigt werden. Die Größe ist natürlich durch die lokale Festplatte begrenzt.
Erklärung localhost!?

Um eine lokale Entwicklung auf diesem Server zu ermöglichen, sind einige Konfigurationen nötig.
Neben der Bereitstellung der richtigen Linux-Version (Für dieses Vorhaben wird die Ubuntu Version Server 16.04.3 LTS herangezogen.) wird hier die Verteilung der Virtual Machine auf die verfügbaren CPUs und die damit verbundene Auslastung eingestellt. (2 CPU - 100% - wichtig????!)

Wichtige Einstellungen im Bereich Netzwerk:
NAT - Network Adress Translation - hier werden die Ports eingestellt über welche die VM kommuniziert.
Portweiterleitung??!?!?
Alle wichtig für lokal development

Rule1: Apache / TCP / -empty for any ip / Host: 8080 / Guest : 80 for default HTTP-port for web traffic
->Apache - Webbrowser
Rule 2: MySQL / TCP / Host: 9306 / Guest: 3306
->Verbindung zu MySQL server -> debugging uploading etc.
Rule 3: MailCatcher / / 1080 / 1080
Receive only Mail-server, dass nicht zufällig E-Mails versandt werden
Rule 4: SSH / 2222 / / 22
Hierüber kann die VM über die Kommandozeile gesteuert werden
- bidirectional - traffic I/O
Firewall erwähnen!!! - ist auf den meisten Systemen bereits vorhanden
Ports einstellen ist aber kein Ersatz für eine Firewall – siehe später in squid
Auf dem lokalen System -Mac- wird ein Ordner erstellt -sandbox- welcher als gemeinsamer Ordner automatisch eingebunden wird.

Ubuntu installieren:
sudo - superuser do - benötigt um root-befehle zu geben
LAMP server für apache / OpenSSH für SSH zugriff
localhost - der eigene Computer wird via Loopback zugegriffen!?
Ändern des Hostname am Mac für die VM zu sandbox.dev
->Aufrufen der hosts datei -> Terminal

```
$ sudo nano /etc/hosts
```

add 127.0.0.1 sandbox.dev
Nun kann man sich mit folgendem Befehl auf der VM einloggen.

```
$ ssh sandbox.dev
```

Um nicht jedes mal ein PW eingeben zu müssen und um man-in-the-middle-Attacken zu verhindern, werden nun keys generiert.

```
$ ssh-keygen -t rsa -C "username@example.com"
```

Nun kann noch ein extra pw eingegeben werden, aber da es eine lokale Installation ist, wird das nicht gemacht.

Wenn der Private key generiert wurde, wird noch ein public key generiert.

```
$ ssh -p2222 marc@sandbox.dev mkdir -p .ssh
```

Um sich nun direkt auf den Server einloggen zu können:

```
$ cat ~/.ssh/id_rsa.pub | ssh -p2222 marc@sandbox.dev 'cat >> .ssh/
    authorized_keys'
```

Da nun immer noch ssh -p2222 marc@sandbox.dev geschrieben werden muss - wird die config datei verändert.

```
$ nano ~/.ssh/config
```

Edit:
Host sandbox.dev Port 2222 User marc
Nun kann man sich einfach mit ssh sandbox.dev auf den Server einloggen (lokal).

7.2 Einrichten eines Apache Web-Servers

Um den Web-Server zu konfigurieren wird dieser zuerst auf den neuesten Stand gebracht. apt - advanced packages tool (packages erklären?!)

ssh sandbox.dev soll nicht jedes mal erwähnt werden.. Abgrenzen mit Farben wann man eingeloggt ist und wann nicht!?

Da die Installation-disc nicht alle updates beinhaltet: Für alle neuen Installationen und Fehlerbehebungen:
- Update -> update list of available packages

```
$ sudo apt-get update
```

Nun müssen alle erworbenen packages auf den neuesten Stand gebracht werden. - Upgrade -> upgrade currently installed software

```
$ sudo apt-get upgrade
```

Virtual-Box-Integration:
Virtual Box verfügt über ein Add-On -> Guest Additions.
-> Eine Ansammlung von Treibern und System Programmen -> diese optimieren das OS für Performance und Usability.
Dies wird zB beim Filesharing gebraucht - zwischen Host und Guest.

Man benötigt: build-essential –> tools for compiling
und virtual-box-dkms –> Dynamic Kernel Module Support
module-assistant –> handles module source packages

Additional Software?! s. Bilder

```
$ sudo apt-get install build-essential virtualbox-dkms nano zip unzip curl
  man-db acpid git module-assistant
```

–> sudo reboot

–> log back in ssh....

7.2.1 Installation Guest-Additions

Install Virtual-Box Guest Additions to share Data betw. Guest and hosts: the shared folder will be
Der mount-command wird benutzt um the device file system to the file tree zu attachen.
Man kann auf verschiedenen Content zugreifen - wie zb eine CD.
Es wird eine virtuelle CD in das virtuelle Laufwerk der virtualBox gelegt um die Guest Additions zu laden.
Zum mounten benötigt man ein device und ein directory

-> Device: dev cdrom (immer mit slashes)
-> directory: media cdrom

Erstmal die CD einlegen -> Devices -> Insert.... Screenshot!?

Durch ls -la /dev oder /media schauen ob die cdrom files da sind?!

Dann mounten:

```
$ sudo mount /dev/cdrom /media/cdrom
```

Das Terminal zeigt:

```
mount: /dev/sr0 is write-protected, mounting read-only
```

Das ist normal. Wie bei einer cd.

Nun werden die Guest Additions installiert.

```
$ sudo sh /media/cdrom/VBoxLinuxAdditions.run --nox11
```

–> sudo reboot und ssh back again.

Check die geladenen modules.

```
$ lsmod | grep vbox
```

grep erklären -> vboxsf muss vorhanden sein - shared folder.

Nun sollte der media folder überprüft werden:

```
$ ls -la /media
```

Hier sollte ein Verzeichnis cdrom und ein sf_sandbox vorhanden sein!!!!

sh-Command - built in command interpreter lsmod - lists (s. Bilder)

7.2.2 Zugang zu sf

Nun muss Zugang zu den shared Folders gewährleistet werden.

Da der Ordner sf_sandbox nicht root sondern Gruppe vboxsf ist hat man bisher keinen Zugriff auf diesen Ordner. Hier für muss der Zugang gewährleistet werden.
Für den user marc:

```
$ sudo usermod -a -G vboxsf marc
```

Der Zugang erfolgt nach einmaligem logout. -> logout -> ssh -> Zugang sollte da sein.
Der Apache server hat den usernamen www-data -> Dieser braucht ebenfalls zugang zum sf.

```
$ sudo usermod –a –G vboxsf www–data
```

7.2.3 Linux Konfiguration für den Apache Server

Der Server weis bisher noch nichts von sf.

command sudoedit - overwrite original file:
vboxsf - HTTP Apache configuration - copy paste from exercise files
wie soll das gehändelt werden??

```
$ sudoedit sites–available/vboxsf.conf
```

Als nächstes die Ports konfigurieren:
Apache läuft eigt auf 80 - es wurde aber auf 8080 weiter geleitet
ports.conf -> verändern

```
$ sudoedit ports.conf
```

Unter Listen 80 noch Listen 8080 hinzufügen.

Managing Apache Sites in Ubuntu s. Bilder

```
$ sudo a2ensite vboxsf
```

```
$ sudo a2dissite 000–default
```

Managing Apache Modules in Ubuntu s. Bilder

```
$ sudo a2enmod rewrite vhost_alias
```

Danach muss der Server neu gestartet werden. Weil Apache neue Group-permissions braucht außerdem wurden die server functionality gechanged durch die modules.

```
$ sudo service apache2 restart
```

Ob der Server funktioniert kann über den Browser herausgefunden werden

```
sandbox.dev:8080/server-status
```

s.Bilder

7.2.4 PHP Konfiguration

Da das Front-End in PHP geschrieben werden soll, müssen hier Vorkehrungen getroffen werden. PHP ist bereits installiert, muss aber für die Entwicklung noch konfiguriert werden.

```
$ sudoedit /etc/php/7.0/mods-available/phpcustom.ini
```

```
; Custom shared config
; priority=01
error_reporting=E_ALL
display_errors=On
display_startup_errors=On
error_log=/var/log/php_errors.log
log_errors_max_len=0
memory_limit=256M
post_max_size=100M
upload_max_filesize=100M
```

Apache weis nun wo er die errors während der Programmierung hinschreiben soll aber das File exisiert noch nicht.
File error_log erstellen
touch command erklären

```
$ sudo touch /var/log/php_errors.log
```

Permission to read and write to the server.
chown-command - change owner and group of files.

PHP-App Frameworks s-Bilder

```
$ sudo apt-get install php-mcrypt php-intl php-sqlite3 php-mbstring php-xml
    php-gd -y
```

Zwei frameworks müssen enabled werden.

```
$ sudo phpenmod mbstring simplexml
```

sudo service apache2 restart

7.2.5 MailCatcher

Benötigte Pakete

```
$ sudo apt-get install libsqlite3-dev ruby-dev -y
```

Mailcatcher selbst installieren

```
$ sudo gem install mailcatcher
```

Testmail

```
$ php -a
php > mail('target@example.com', 'Testmail', 'Was geht ab', 'From:
    source@example.com');
```

8 Design

Once the Server is up and running a framework for programming the website is needed to save time and make the application stable and safe in the end. For this Application the choice is a PHP-Framework named Laravel.

Laravel Begründung?!

At the beginning, the admin-section will be implemented to create users, which will be granted to have access to the Mainframe-Training-Units.

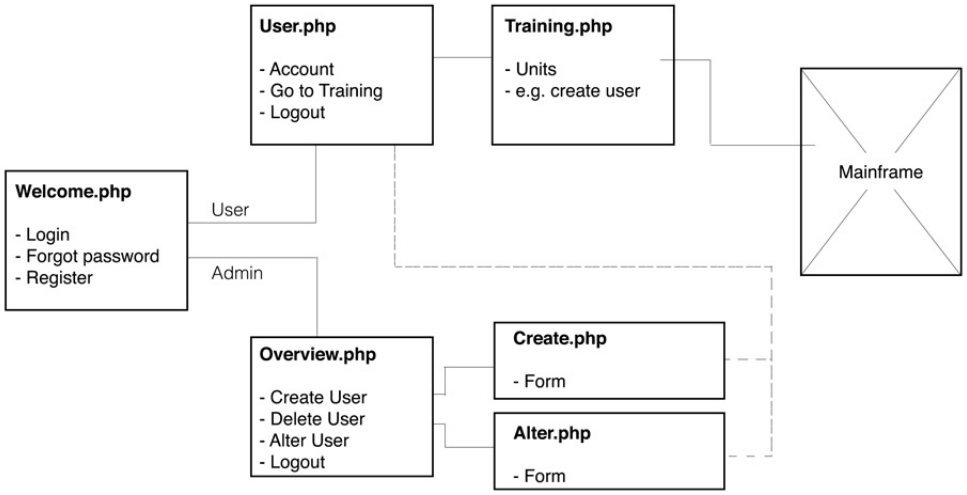

Abbildung 1: schematic presentation of the admin/user-section

8.1 MVC-Framework

Laravel is a framework that follows the MVC-Pattern. MVC stands for Model-View-Controller.

8.1.1 Routing explained

Routes are established (unter anderem) in the web.php file. In the example we see the Routings that are triggering the AdminsController - the @ triggers a function that is

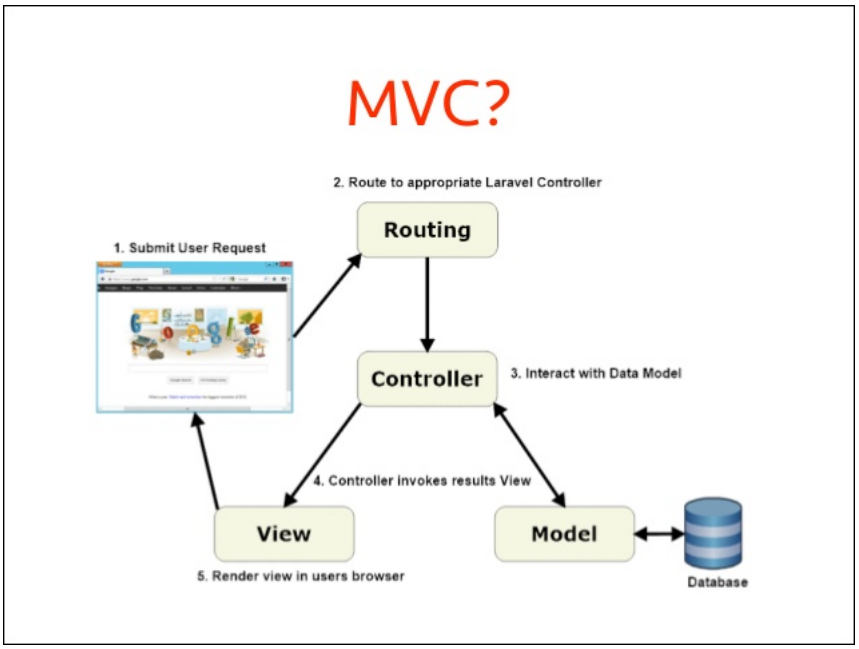

Abbildung 2: Model-View-Controller Pattern

listed in this controller. For example create.

```
/* Admin Routing */
Route::get('/admin', 'AdminsController@admin')->name('admin_view');
Route::get('/admin/alter', 'AdminsController@alter')->name('alter_user');
Route::get('/admin/create', 'AdminsController@newUser')->name('new_user');
Route::post('/admin/create', 'AdminsController@create')->name('create_user'
    );
Route::get('/admin/{user_id}', 'AdminsController@show')->name('show_user');
Route::post('/admin/{user_id}', 'AdminsController@modify')->name('
    update_user');
```

The names help to navigate to the pages in an easier manner. With blade you can just define the link to the named paged.

Blade kurz erklären.

```
<a class="hollow button" href="{{ route('show_user', ['user_id' => 1 ]) }}"
    >EDIT</a>
```

The create function does nothing but return the view / the file 'createuser.php' that is located in the folder 'admin'.

```
    public function create()
    {
      return view('admin/createuser');
    }
```

But within the Controller, also different types of data can be generated and passed to the view.
In this example hard coded user data is passed to the view 'overview'

```
    public function admin()
    {

            $data = [];                                 //data als leeres array
    definieren

            $obj = new \stdClass;
            $obj->id = 1;
            $obj->title = 'mr';
            $obj->name = 'john';
            $obj->last_name = 'doe';
            $obj->email = 'john@domain.com';

            $data['users'][] = $obj;            //data wird als obj definiert

      return view('admin/overview', $data);    //pass data to view
    }
```

While there will be a view to Add a new user and to modify an existing one, due to the controller there is no need to have two different pages / forms for that. There is just on file containing the form that can be used for both - editing and creating. The File here is form.php
In the controller there will be two functions.

```
/* 1st Function */
    public function newUser()
    {
      $data = [];
      $data['modify'] = 0;
      return view('admin/form', $data);
    }

/* 2nd Function */
    public function show($users_id)
    {
      $data = [];
      $data['modify'] = 1;
      return view('admin/form', $data);
    }
```

While they both show the same view 'admin/form', the passed data is different. The modify-sequence on line 5 and 13 enables the view to differentiate whether the passed data should be reorganized or added to the database. This becomes possible through an action that is allocated in the form.php file.

```
1         <form action="{{ $modify == 1 ? route('update_user', [ 'user_id' =>
    1]) : route('create_user') }}" method="post">
```

Here the status of $modify is queried and based on this decision the view routes to a particular function that is defined in the AdminsController. If the user is modified, a user-id has to be set.

9 Proof of Concept

10 Evaluation

Literatur