# IT Umgebung

## Hosting und Unterhalt

Hosting und Unterhalt wird in zwei teile eingeteilt. Entwicklung und Produktion. Im Entwicklungsteil wird die Entwicklungsumgebung beschrieben. Im Produktionsteil wird das Thema Wartung und Unterhalt beschrieben, sowie die Produktionsumgebung.

### Entwicklung

Die Entwicklungsumgebung besteht aus einer Linux VM. Auf dieser ist das LAMP packet Installiert. Entwickelt wird mit der Entwicklungsumgebung Eclipse. Eclipse ist auf meinem Hostcomputer Installiert. Die Entwicklung geschieht über SFTP mithilfe eines Eclipse-plugins. Der ganze Code wird auf Github abgelegt.

### Produktion

Die Produktionsumgebung besteht aus einem von Rock-Server gehosteten vServer. Dieser wird aus dem Internet erreichbar sein. Auf dem vServer wird Linux sowie LAMP Installiert sein.

## Technologie

Die Website wird mit PHP, MySQL, Html und CSS umgesetzt. Als Frontend Framework wird Bootstrap oder Foundation verwendet. Die Website wird auf einem Apache Web Server laufen.

Als Datenbank Provider wird PDO verwendet. Dies ist dynamischer und ansprechender aufgebaut als mysqli. PDO ist nicht von MySql abhängig und kann auch andere Datenbanken ansprechen.

# Projekt

## Ziele

Das Haupt Ziel von diesem Projekt ist es PHP besser kennen zu Lernen und ein Erweiterbares Gästebuch zu erstellen. Dieses Gästebuch soll als Grundlage für weitere Projekte verwendet werden können.

In diesem Projekt wird vor allem auf Sicherheit aber auch auf Browserkompatibilität geachtet.

## Änderungen zur Vorgabe

Ich habe den Usernamen entfernt, da der Username/Benutzername Nirgends verwendet wird. Für das Login wird eine E-Mail Adresse verwenden und angezeigt wird der Fullname.

Ich werde den Webserver anstatt auf den DMZ Server auf einen bei Rock Server gehosteteten Linux vServer migrieren.

## Anforderungen

Um die Anforderungen zu definieren werde ich die aus Scrum Bekannten User Stories verwenden. Diese werden in Funktionale und nicht Funktionale Anforderungen eingeteilt. Die Funktionalen sind für den Benutzer sichtbar und nachvollziehbar. Diese werden durch die Testcases abgedeckt. Die nicht Funktionalen werden anders getestet.

## User Stories

### Funktionale Anforderungen

1. Als Benutzer will ich alle Gästebuch Einträge lesen können.
2. Als Benutzer will ich mich registrieren können. Ich will mich mit einem Benutzernamen, einer E-Mail Adresse, meinem Namen und einem Passwort registrieren können.
3. Als Benutzer will ich mich mit meiner E-Mail Adresse und meinem Passwort Einloggen können.
4. Als eingeloggter Benutzer will ich mich ausloggen können.
5. Als eingeloggter Benutzer will ich Gästebucheintrage erfassen können.
6. Als eingeloggter Benutzer will ich meine Gästebucheintrage bearbeiten können, nur die Eigenen.
7. Als eingeloggter Benutzer will ich meine Gästebucheintrage löschen können, nur die Eigenen.
8. Als eingeloggter Benutzer will ich mein Passwort ändern können.

### Nicht Funktionale Anforderungen ( nicht durch UI Testbar)

1. Als eingeloggter Benutzer will ich, dass mein Passwort sicher in der Datenbank gespeichert wird.
2. Als eingeloggter Benutzer will ich, dass mein Account/meine Identität nicht missbraucht wird.
3. Als eingeloggter Benutzer will ich nicht dass meine Einträge in einer Suchmaschine gefunden werden.

# Test Cases

Die Testcases/Testfälle sind im Testprotokoll.docx beschrieben.

# Integration

Das Gästebuch wird auf einem Ubuntu Server Installiert. Das LAMP Packet muss auf dem Server installiert sein.

LAMP Installations-Anleitung: <https://www.howtoforge.com/ubuntu_lamp_for_newbies>

Zusätzlich muss noch das Packet mcrypt Installiert und aktiviert werden:

sudo apt-get install php5-mcrypt

sudo php5enmod mcrypt

sudo service apache2 restart

Nun geht es zu Installation und Konfiguration des Webservices.

Die Applikationsdateien können via git clone oder mit einem manuellen Download ins www Verzeichnis des servers geladen werden.

Nun muss im lib Ordner noch die passwd.php Datei erstellt und beschrieben werden.

Die Datei sollte in etwa so aussehen:

<?php

define("DB\_NAME", "homepageanwendungen");

define("DB\_USER", "phpuser");

define("DB\_PASSWORD", "<passwort>");

define("DB\_HOST","localhost");

DB User und Passwort müssen noch angepasst werden.

Anschliessend kann der vHost konfiguriert werden.

Dazu muss im /etc/apache2/sites-available/ eine neue Datei mit dem Namen gaestebuch.conf erstellt werden. Die Datei muss gegebenenfalls noch angepasst werden.

<VirtualHost \*:80>

ServerName devserver.local

ServerAdmin webmaster@localhost

DocumentRoot /var/www/gibb-modul-151

<Directory /var/www/gibb-modul-151>

AllowOverride None

RewriteEngine on

#Um auf files zuzugreifen...

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule ^(.\*)$ index.php [L,NC]

Options Indexes FollowSymLinks MultiViews

Order allow,deny

allow from all

DirectoryIndex index.php

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Dieser vHost muss nun mit dem Befehl a2ensite gaestebuch aktiviert werden.

Falls das neustarten des apache2 fehlschlägt muss noch das Rewrite Modul Aktiviert werden (a2enmod rewrite).

Soweit so gut, nun kann das Gästebuch getestet werden. Dazu kann im Browser die IP des Servers aufgerufen werden.

Falls irgendwelche Fehler auftreten oder der Server mit Fehlercode 500 antwortet, steht meistens im Apache Fehler log eine Meldung. /var/log/apache2/error.log

# Sicherheitsgedanken

Beim erstellen von diesem Projekt habe ich mir einige Sicherheitsgedanken gemacht.

## Risiken

Die Website steht im Internet und ist von überall erreichbar, deshalb bestehen auch Risiken.

In diesem Abschnitt werden die Risiken beschrieben und nach den Kriterien Wahrscheinlichkeit und Auswirkung bewertet.

### 1. DDOS

DDOS ist ein Angriff für den die Meisten Webserver verwundbar sind. Ein DDOS geschieht meistens dadurch, dass sehr viele oder unerwartete Anfragen an den Server gestellt werden und die ganzen Ressourcen aufbrauchen. In diesem Fall ist es normalen Nutzern nicht mehr möglich den Dienst normal zu nutzen.

DDOS Ist bei diesem Projekt kein grosses Problem, da keine hohe Verfügbarkeit nötig ist, und die Firewall des Hosters einige Angriffe abwehrt.

### 2. SQL Injection

SQL Injection ist ein Angriff bei dem versucht wird Daten aus der Datenbank auszulesen oder diese zu überschreiben. Mit dieser Attacke können Logins umgangen und Daten gestohlen werden.

Gegen SQL Injection werden Prepared Statements verwendet. Ausserdem wurde ein Test mit SqlMap durchgeführt und es wurden keine einfachen Sicherheitslücken gefunden.

### 3. XSS

Cross Site Scripting oder XSS ist ein angriff bei dem sogenannte Scripts meist Javascript auf einer Website platziert werden, dass diese auf einem anderen Client ausgeführt werden. Mit dieser Attacke kann zum einen das Aussehen der Seite manipuliert werden, andererseits ist es auch möglich Benutzerdaten und Sitzungen von anderen Benutzern zu ergattern und sich somit Unerlaubten zugriff zu verschaffen.

### 4. Passwort-/Datenklau

Ein Passwortklau ist nicht direkt ein Angriff, ist aber auch ein Risiko. Meist geschieht ein Passwortklau durch eine SQL Injection, durch Unsichere Admin Passwörter oder durch Fehler in der Software. Es werden meist Nutzerdaten und somit auch Passwörter gestohlen. Da einige Nutzer ihre Passwörter für verschiedene Dienste wiederverwenden kann so auch Zugriff zu anderen Seiten oder Diensten erlangt werden.

### 5. Spam

Spam Angriffe geschehen meist dadurch, dass Nutzer Verbotene, Unerwünschte oder übertrieben viele Einträge auf der Seite erstellen.

Es ist oft schwierig Spam von Richtigen Einträgen zu unterscheiden.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Auswirkung | | |
| Wahrscheinlichkeit |  |  |  |
|  |  |  |
|  |  |  |

## Sichereheitsgedanken

## Zugriff von Aussen

Um die Seite vor unerlaubten zugriffen zu schützen, wird eine Basic Authentication verwendet.

Der zugriff zum Server ist nur via SSH möglich. Der SSH Zugang ist mit einem starken Passwort(mind. 24 zufälligen Zeichen) gesichert.

## Passwörter / Logins

Um alle Logins gegen Dictionary Attacke zu schützen werden sichere zufällig generierte Passwörter für alle kritischen dienste verwendet. Kritische dienste wie SQL Logins und auch Server Logins. Die zufällig generierten Passwörter haben eine mindestlänge von 20 Zeichen und müssen Zahlen, Buchstaben und Sonderzeichen enthalten.

Auch von den Seitennutzern werden starke Passwörter verlangt. Die Passwort Policys sind die selben wie beim vorherigen Web Projekt: mind. 8 Zeichen lang, gross und Kleinschreibung, mindestens eine Zahl und ein Sonderzeichen.

## SQL Injection

Um einer SQL Injection vor zu beugen werden Querys nicht Dynamisch generiert. Es werden ausschliesslich Prepared Statements verwendet. Als Datenbank Provider wird anstatt mysqli PDO verwendet, da PDO nicht von MySQL abhängig ist.

## XSS

Um XSS vorzubeugen werden alle Requests auf Verbotene Tags, wie das <script> Tag gefiltert. Diese Tags werden anschliessend entfernt.

## Password Hashing

Um auch die Sicherheit für die Nutzer des Gästebuchs zu erhöhen werden die Benutzerpasswörter mit einer 32 Bit langen Zufallszahl ergänzt. Dieser Zufallszahl wird auch Salt genannt. Anschliessend werden Passwort und Salt zusammen mit sha256 gehasht.

## Weiterleitungs Validierung

Während der Implementation des Gästebuchs ist mir aufgefallen, dass Backurls sinnvoll wären. Damit mithilfe dieser Backurls keine Verbotenen Requests ausgeführt werden, werden die Backurls validiert. Backurls müssen mit einem „/“ beginnen. Somit werden zugriffe auf andere Seiten verhindert. Ausserdem ist das „?“ ebenfalls ein Verbotener Charakter.

Bei Bedarf können einzelne Externe Urls freigeschaltet werden.

## Robots.txt und Basic Auth

Um auf die Seite zu zugreifen, wird eine Basic Authentication verwendet. So werden unerwünschte zugriffe während der Testphase verboten.

Damit die Seiteneinträge der Nutzer nicht von Suchmaschinen gefunden werden ist ein robots.txt File vorhanden, welches Suchmaschinen den Zugriff verbietet.