I8 - Erstellung und Verwaltung von Gruppen für Software Engineering Projekte - Anwenderdokumentation

Inhaltsverzeichnis

1. Einführung

Dieses Dokument beinhaltet detaillierte Informationen zum Aufbau der Software und ihrer Komponenten, um die Einarbeitung anderer Entwickler zu erleichtern. Dieses Dokument wird durch eine HTML Seite für die Dokumentation der Python Django Module erweitert (mehr dazu unter Punkt Django).

2. Frameworks

Folgende Frameworks und Bibliotheken wurden für die Java Entwicklung verwendet:

- Java Spring Boot
- Bootstrap
- MySQL Client

3. Architecture Notebook

3.1. Zweck

Das vorliegende Dokument umfasst eine Beschreibung der Philosophie, der getroffenen Entscheidungen, der Nebenbedingungen, der Begründungen sowie der wesentlichen Elemente und anderer übergreifender Aspekte des Systems, welche den Entwurf und die Implementierung beeinflussen.

3.2. Architekturziele und Philosophie

Das Ziel der Architektur besteht darin eine Webanwendung für Studierende und Professoren schaffen, die im Umfang des Software Engineering Modul die eingetragenen Studenten auf verschiedene Teams verteilt. Die dabei entstehende Webanwendung basiert in ihrer Struktur auf der bereits bestehenden Website, wird jedoch zu großen Teilen neu erstellt und soll nach Abschluss deshalb die bestehende Webanwendung vollständig ersetzen. Die Schwerpunkte der Architektur liegen hauptsächlich darauf, die Funktionalitäten der bereits vorhanden Webanwendung zu replizieren, jedoch dabei bereits neue Funktionalitäten von Grund auf hinzuzufügen. Die hohe Nutzerfreundlichkeit und intuitive Benutzeroberfläche sollen ebenso wieder hergestellt und erweitert werden.

Ein wichtiger Teil der Webanwendung besteht aus einem Login für die Studierenden, welches eine Verbindung mit den LDAP Server der HTW - Dresden herstellt, um die Daten zu validieren. Deswegen ist es besonders wichtig, dass das System vertraulich mit den Studentendaten umgeht und die Verbindung zu dem LDAP Server verschlüsselt ist.

Die Architektur soll in Struktur und Verhalten eine hohe Nutzungsdauer gewährleisten, so dass die Webanwendung über einen langen Zeithorizont genutzt werden kann. Ein weiterer Schwerpunkt besteht darin, das Backend strukturiert und dynamisch zu entwickeln, damit es eine hohe Erweiterbarkeit aufweist, wodurch neue Funktionen oder Module problemlos hinzugefügt werden können. Das Ziel besteht darin, eine solide Basis für eine Weiterentwicklung zu schaffen, die eine schnelle Einarbeitung und hohe Verständlichkeit ermöglicht.

3.3. Annahmen und Abhängigkeiten

Annahmen:

- Jeder Benutzer hat einen Zugriff auf das HTW-Netzwerk.
- Alle Benutzer haben eine stabile Verbindung zu unserem Webserver mit ausreichend Bandbreite.
- Die von uns verwendeten Technologien und Frameworks funktionieren ohne schwerwiegende Fehler und werden zukünftig gewissenhaft gepflegt.
- Alle Benutzer besitzen einen Webbrowser, welcher HTML 5 fähig ist.
- Jeder Student besitzt einen gültigen HTW-Login.
- Für die Studenten liegt eine Einschreibung im Modul Software Engineering I vor.

Abhängigkeiten:

- Die Webanwendung ist abhängig von dem Rechenzentrum der HTW Dresden, da dieses den Server bereitstellt, auf dem die fertige Webanwendung installiert wird.
- Unser Projekt ist abhängig von dem LDAP Login Server der HTW Dresden, da unsere Website nur mittels des HTW Logins zu erreichen ist.
- Sollten sich Daten für die Kommunikation mit dem LDAP Server der HTW - Dresden ändern, werden diese Änderungen in das System eingepflegt.
- Wir verlassen uns darauf, dass die von uns verwendeten Technologien und Frameworks auch in Zukunft gepflegt und gewartet werden.

3.4. Architektur-relevante Anforderungen

Funktionalität (Functional)

- NFAF-1 Die Webanwendung bietet den Dozenten die Funktion Teams nach bestimmten Faktoren automatisiert mittels eines Algorithmus erstellen zu lassen.
- NFAF-2 Die Webanwendung muss die Studentendaten inklusive der Fragebogenantworten persistent speichern.
- NFAF-3 Die Verbindung zu dem LDAP Server der HTW - Dresden ist verschlüsselt, um die Sicherheit der Studentendaten zu gewährleisten
- NFAF-4 Ein Login für Studenten ist ausschließlich möglich, wenn diese vorher in das System importiert wurden.

Effizienz (Performance)

- NFAP-1 Das System soll robust gegen einen Ausfall bei einer gleichzeitigen Nutzung durch zehn Personen sein.
- NFAP-2 Der Algorithmus sollte innerhalb zehn Minuten das bestmögliche Ergebnis berechnen.
- NFAP-3 Das Laden von Seiten sowie anzuzeigender Informationen ist zeiteffizient.

Wartbarkeit (Supportability)

- NFAS-1 Das System soll für einen Administrator ausführlich dokumentiert sein, um eine einfache Wartung zu ermöglichen.
- NFAS-2 Die Möglichkeit die Webanwendung um verschiedene Funktionalitäten zu erweitern, sollte durch eine entsprechende Struktur und geeignete Schnittstellen geben sein.

3.5. Entscheidungen, Nebenbedingungen und Begründungen

Das ursprüngliche Ziel bestand darin die vorhandene Webanwendung der Gruppe I7 weiterzuentwickeln und um verschiedene Funktionen zu erweitern. Allerdings zeigte sich nach genauerer Betrachtung des vorhandenen Systems, sowie nach mehreren Konsultationen mit Teammitgliedern der Gruppe I7, dass die vorhandene Architektur und der Quellcode sich nicht für eine Erweiterung eignen. Aus diesem Grund wurde sich einheitlich für eine Neuentwicklung entschieden, welche auf der Struktur der vorhandenen Webanwendung basiert. Aus diesem Grund werden folgende Entscheidungen der Vorgängergruppe weitergeführt:

1. Die Datenbanken werden weiterhin mit MySQl erstellt und verwaltet, da MySQL einen vielfältigen Anwendungsbereich besitzt und gut dokumentiert ist.

2. Wir werden weiterhin Docker verwenden, da Docker eine einfache Bereitstellung, sowie hohe Portabilität und Konsistenz bietet.

3. HTML und CSS stellen weiterhin die Grundlage für unser Frontend dar, weil sie grundlegende Technologien für das Erstellen von Webseiten und Benutzeroberflächen sind und eine hohe Flexibilität und Anpassungsfähigkeit bieten.

4. Als Webserver verwenden wir Nginx, da er eine hohe Leistung und Skalierbarkeit bietet.

5. Wir werden weiterhin das Framework OR-Tools für den Constraint Solver nutzen, da dieser gut dokumentiert ist und ein einfaches Hinzufügen von Restriktionen bietet.

Im Folgenden werden Entscheidungen gelistet, die sich in Bezug auf die Vorgängergruppe geändert haben:

1. Für das Backend werden wir die Programmiersprache Java anstatt Python verwenden, da wir bereits Kenntnisse mit der Entwicklung in Java besitzen und eine Einarbeitung in Python einen großen zusätzlichen Zeitaufwand bedeuten würde.

2. Als Framework verwenden wir Java Spring Boot, da es eine effiziente und produktive Entwicklungsumgebung bietet, die speziell für die Erstellung von skalierbaren und robusten Java-Anwendungen entwickelt wurde.

3. Das Frontend wird mit Bootstrap designt, da es ein einsteigerfreundliches und flexibles Frontend-Framework ist, welches es uns ermöglicht, schnell und effizient, ansprechende und benutzerfreundliche Webseiten zu erstellen.

3.6. Architekturmechanismen

Doku "Concept: Architectural Mechanism"

1. Zur dauerhaften Speicherung der Fragebogen- und Zuordnungsdaten greifen wir auf die relationale Datenbank MySQL zurück. Diese Entscheidung haben wir getroffen, da sich die zu sichernden Daten gut in einem relationalen Schema darstellen lassen. MySQL bietet dabei eine weit verbreitete Lösung zur zuverlässigen und sicheren Speicherung der Daten. Außerdem kann durch das individuell einstellbare Rechtesystem in der Datenbank der Zugriff vor Unbefugten geschützt werden (siehe nicht funktionale Anforderungen).
2. Wir haben die Entscheidung getroffen, dass die Funktionalität der Buttons einheitlich visuell dargestellt werden soll, da dies die Usability für die Nutzer erhöht. Dabei hatten wir entweder die Möglichkeit das User Interface manuell per CSS von Grund auf neu zu entwickeln oder ein Framework mit vorgefertigten Design Elementen zu wählen. Das manuelle Designen mittels CSS bietet zwar sehr großen Freiraum und nahezu unbegrenzte Gestaltungsmöglichkeit, jedoch kostet dieses Vorgehen auch sehr viel Zeit. Deshalb haben wir uns für die zweite Möglichkeit entschieden und das Design mit Bootstrap umgesetzt. Dies schränkt zwar die Gestaltungsmöglichkeiten ein, bietet für unsere Anforderungen jedoch ausreichend Freiraum und erspart uns sehr viel Zeit.
3. Für die Teamzusammenstellung werden folgende Methodiken in Betracht gezogen: Zuordnen & Vertauschen Algorithmus, genetischer Algorithmus und Algorithmus mittels Constraint Solver. Letztendlich haben wir uns für einen Algorithmus mittels Constraint Solver entschieden, da dieser es uns ermöglicht schnell neue Restriktionen (Constraints) hinzuzufügen. Dies erleichtert die Entwicklung und auch eine eventuelle Weiterentwicklung unseres Projekts in Zukunft. Außerdem schnitt der Algorithmus mit Constraint Solver im Vergleich zu den anderen Varianten in Sachen Lösungsqualität als auch Lösungszeit am besten ab.
4. Die neuentwickelte Webanwendung wird weiterhin auf Linux basieren, wodurch diese auch in Zukunft im HTW-Netzwerk über Linux laufen wird. Die Möglichkeit die Website aufzurufen ist für jeden Browser auf jedem Betriebssystem möglich, erfordert jedoch, dass der entsprechende Benutzer sich im HTW-Netzwerk befinden.
5. Der Zugang zur Dozentenansicht ist für alle Benutzer möglich, die die entsprechenden festen Benutzerdaten besitzen. Der Fragebogen ist für alle Studierenden zugänglich, welche einen gültigen HTW-Account besitzen und deren Daten in einer CSV Datei in das System importiert wurden. Voraussetzung dafür ist, dass sich die Studenten vorher im OPAL für den Beleg eingeschrieben haben. Zusätzlich ist das Erreichen der Website ausschließlich im HTW-Netzwerk möglich. Benutzer von außen benötigen somit eine VPN Verbindung in das Netzwerk der HTW, welche ihre eigenen Sicherheitsmaßnahmen besitzt.

3.7. Wesentliche Abstraktionen

Student: Enthält Stammdaten (s-Nummer, Name, Studiengang und weiteres) zu den belegbearbeitenden Studenten und verweist auf deren Fragebogenergebnisse. In Verbindung mit der s-Nummer können die Logindaten der Studenten abgeglichen werden.

Dozent: In der Rolle des Admins, welcher in der Dozentenansicht die Website verwaltet. Dazu zählt das Einstellen verschiedener Ansichten des Fragebogens, sowie das Verwalten gespeicherter Daten. Zusätzlich besitzt er die Möglichkeit Projekte und Teams zu erstellen, bearbeiten und zu löschen.

Fragebogen: Je nach Einstellung der Website durch die Dozenten, wird den Studenten unterschiedlicher Inhalt angezeigt. Der Fragebogen besteht aus einer Ansicht der Projekte, sowie drei Blöcken in denen unterschiedliche Fragen bezüglich der Projekte gestellt werden. Die Antworten werden gespeichert und beim erneuten Aufruf der Website dem entsprechenden Studenten angezeigt.

Teamübersicht: Sollten noch keine Teams erstellt wurden sein, wird der Benutzer darauf hingewiesen und er erhält die Möglichkeit die Teams zu erstellen. Wenn Teams erstellt wurden, werden diese als Blöcke bestehend aus ID, Thema, Teamleiter und Teammitglieder für den Dozent angezeigt. Dieser besitzt die Möglichkeit die fertigen Teams für die Studenten sichtbar zu schalten.

Projektübersicht: Zeigt eine Liste der vorhandenen Projekte bestehend aus ID, Name, Beschreibung und Ansprechpartner an. Die Projekte sind nach ihrer ID sortiert und es besteht die Möglichkeit vorhandene Projekt zu bearbeiten und zu löschen. Weiterhin besteht die Möglichkeit weitere Projekte hinzuzufügen, indem sich eine Eingabemaske öffnen lässt.

3.8. Architektursichten (Views)

3.8.1. Model-View-Controller(MVC)

ergibt sich aus:

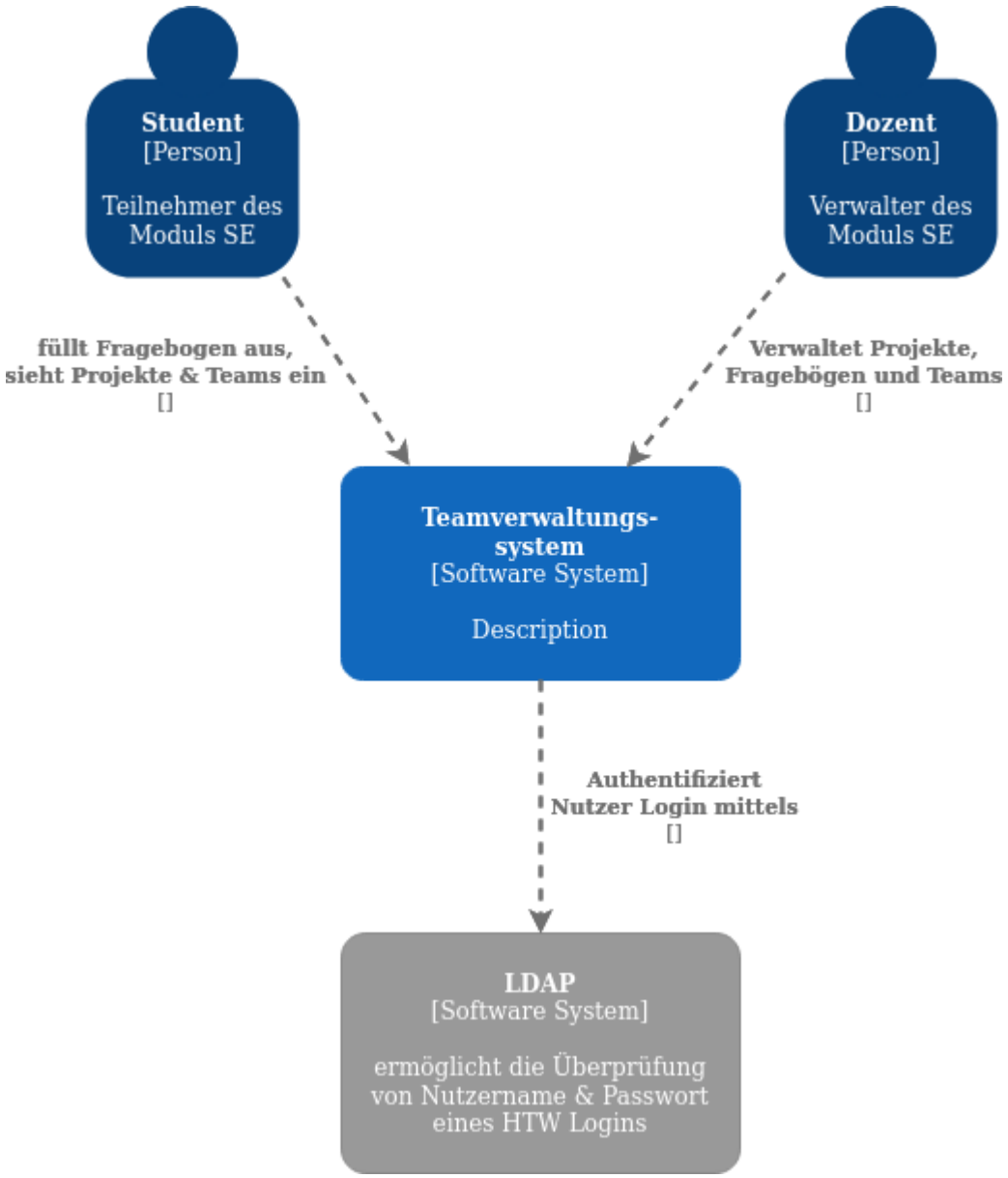
- Model: Docker als Open SourceTool zur Bereitstellung der Container in denen die verschiedenen Elemente des Backends vorhanden sind.
- View: Oberfläche aus HTML und CSS, implementierte Vorlagen von Bootstrap
- Controller: Java Spring Boot Controller mit HTTP und JPA Anfragen

Im Nachfolgenden werden die Architektursichten anhand der ersten beiden Levels des C4 Modells erläutert.

3.8.2. Logische Sicht

Unser System, welches für eine Nutzung im HTW Netzwerk vorgesehen ist, besteht weitestgehend aus internen Verbindungen. Wie im unten anliegenden C4 Model zu erkennen, ist die einzige Schnittstelle nach außen die Verbindung zu dem LDAP Server der HTW Dresden. Jener gibt uns die Möglichkeit, die Authentifizierung für unsere Website über den bereits bestehenden HTW Login zu gestalten. Diese Entscheidung ist auch wieder mit der Nutzerfreundlichkeit, aber auch der Sicherheit zu begründen. Einerseits muss keine externe Seite verwendet werden und zum anderen muss kein neuer Login angelegt werden. Dies lässt den Nutzer die Website sofort verwenden, ohne auf eine Bestätigungsemail oder ähnliches zu warten.

C4 Modell - Level 1:



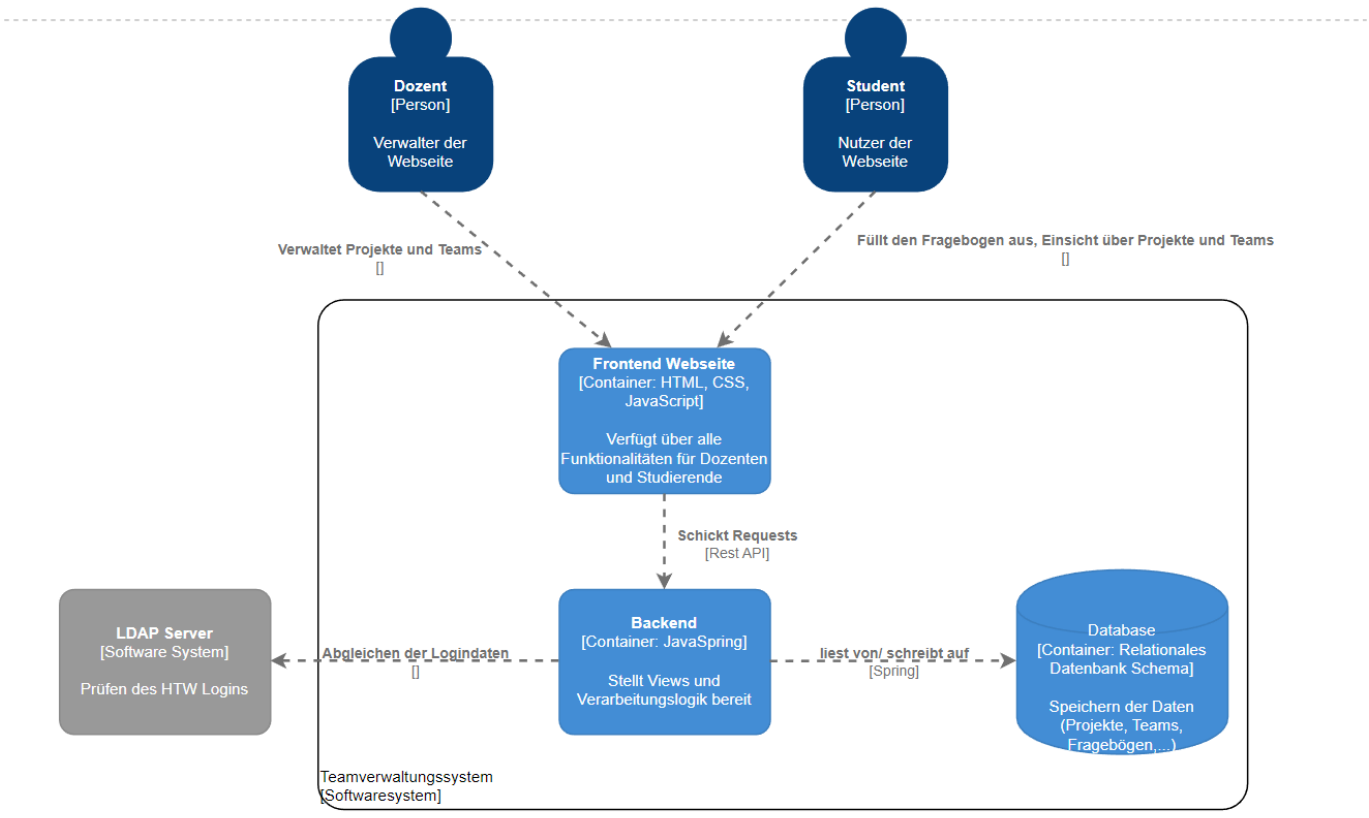
3.8.3. Physische Sicht

Im zweiten Level unseres C4-Modeles wird die Grundstruktur und der Aufbau unseres Projektes deutlich. Auf unserem Server für das Programm laufen mittels Docker mehrere Container:

- Webserver Nginx: Verantwortlich für die Darstellung der Webseite auf den Endgeräten der Nutzer.
- Java Spring Boot: greift auf die Daten aus der Datenbank zu und verwaltet diese mit abstrahierten Funktionen. Verwaltet zudem die Oberfläche für die Benutzer und zeigt entsprechende Daten an.
- MySQL Datenbank: speichert alle erhalten Daten persistent.

Als optimales Tool für die Verwaltung und Kommunikation zwischen diesen Elementen bietet sich Docker an. Docker ermöglicht das Betreiben von der Datenbank, Spring und der Oberfläche als Server unter Verwendung von Containern. Hierbei laufen die einzelnen Komponenten in separaten Prozessen. Die Container können als virtuelle Knoten betrachtet werden, die miteinander kommunizieren. Die Kommunikation erfolgt zwischen Spring und dem Webserver, sowie Spring und der Datenbank.

C4 Modell - Level 2 (Teamverwaltungssystem)



3.8.4. Use Cases

4. Server

In diesem Abschnitt wird jede Komponente des Servers und ihre Rolle erläutert.

Im Beispiel läuft der Server unter Ubuntu, aber dieses Dokument kann auch analog für andere Distro (Arch Linux, OpenSuse, Fedora, usw.) verwendet werden. Für die Installation des benötigen Pakets folgen Sie bitte den Anweisungen der Distribution. Außerdem kann alles, was in Docker läuft, auch ohne Docker laufen.

4.1. Bestandteile des Servers

4.1.1. Docker

Um Docker und Docker Compose aufzubauen und auszuführen, werden die Docker-Engine, Docker CLI und Docker Compose benötigt. Unter Windows und MacOS gibt es eine gemeinsame Methode namens Docker Desktop, welche eine proprietäre Anwendung ist. Docker Desktop enthält alle erforderlichen Komponenten und führt sie innerhalb einer virtuellen Linux-Maschine aus. Dies ermöglicht Windows-Nutzern die Verwendung von Docker über das Windows Subsystem for Linux (WSL), das Docker unter WSL ausführbar macht.
Unter Linux kann das Paketmanagement verwendet werden, um die Pakete "docker" und "docker-compose" einfach zu installieren. Nach der Installation starten Sie den Docker-Dienst "docker.service" und aktivieren ihn, damit er beim Booten automatisch gestartet wird.

Docker ermöglicht eine schnelle Bereitstellung und einfache Testbarkeit von Anwendungen. Wenn Docker Compose lokal erfolgreich ausgeführt wird, kann davon ausgegangen werden, dass es auch auf dem Server fehlerfrei funktioniert.

```
systemctl enable --now docker.service
```

Normalerweise ist es dem Root-Benutzer vorbehalten, Docker auszuführen. Um Ihrem aktuellen Benutzer die Ausführung von Docker zu ermöglichen, müssen Sie ihn der "docker"-Benutzergruppe hinzufügen. Nachdem Sie dies getan haben, melden Sie sich erneut an und starten Sie den Docker-Dienst neu. Dadurch erhält Ihr Benutzer die erforderlichen Berechtigungen, um Docker ohne Root-Rechte auszuführen.

```
usermod -aG docker username systemctl restart docker.service
```

Docker Compose ist ein alternatives CLI-Frontend für die Docker-Engine, das sich besonders für die komplexe Konfiguration von Docker-Containern eignet. Es bietet eine benutzerfreundliche Möglichkeit, mehrere Container und deren Abhängigkeiten zu verwalten und zu orchestrieren.

```
docker-compose up --build -d
```

Mit diesem Befehl können Container gestartet und bei Bedarf erstellt werden.

```
```

Docker und Docker-Compose ermöglichen daher eine schnelle Bereitstellung komplexer Systeme. Entwickler können Docker auch nutzen, um zu überprüfen, ob die neue Version des Quellcodes in der Produktionsumgebung funktioniert, bevor sie diesen auf den Server übertragen. Zudem ist es einfach, Module (Services) auszutauschen, hinzuzufügen oder zu entfernen. Dies bietet Flexibilität bei der Systemkonfiguration und erleichtert die Anpassung des Systems an sich ändernde Anforderungen.

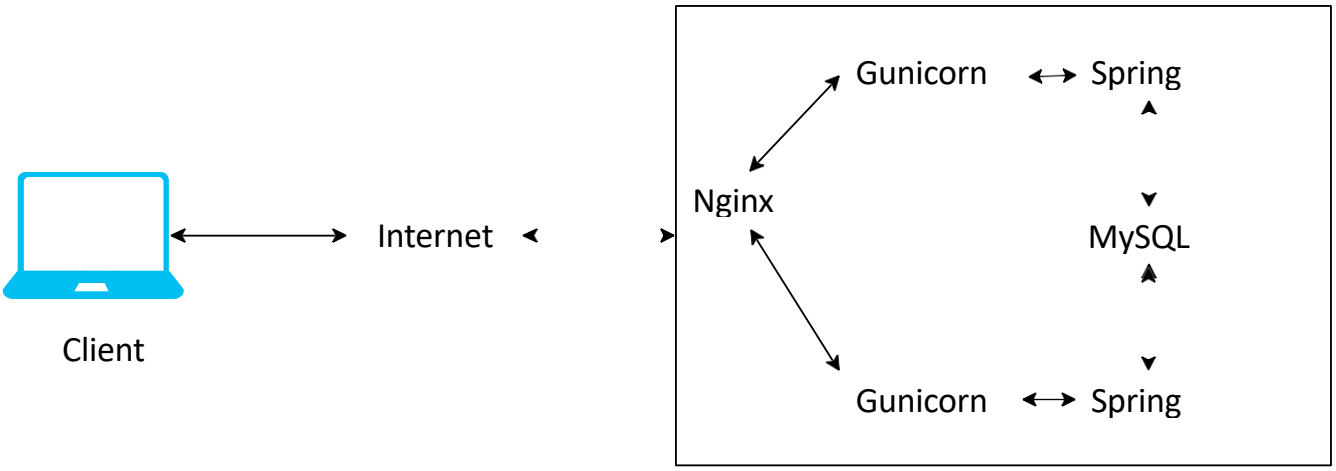


Abbildung 3. Docker Compose für die Webseite Teamverwaltung

```
version: "3.9"
services:
 db:
  image: mysql:8
  restart: always
  environment:
   MYSQL_ROOT_PASSWORD: root_password
   MYSQL_DATABASE: studentdb
   MYSQL_USER: Nuha
   MYSQL_PASSWORD: 123
  ports:
   - "3306:3306"
  volumes:
   - ./database/db.sql:/docker-entrypoint-initdb.d/db.sql
 web:
  image: nginx:alpine
  ports:
   - 80:80
   - 443:443
  volumes:
   - ./Website:/usr/share/nginx/html/Website
   - ./nginx/nginx.conf:/etc/nginx/nginx.conf
   - ./nginxm/mime.types:/etc/nginx/mime.types:ro
   - ./ssl:/etc/nginx/ssl
 spring:
  image: openjdk:20-jdk
  restart: always
  ports:
   - 8080:8080
  environment:
   - SPRING_DATASOURCE_URL=jdbc:mysql://db:3306/studentdb
   - SPRING_DATASOURCE_USERNAME=Nuha
   - SPRING_DATASOURCE_PASSWORD=123
  volumes:

   - ./spring/spring.jar:/app/spring.jar
  command: java -jar /app/spring.jar
  depends_on:
   - db
```

Der Client kann nur mit Nginx auf Port 80 und 443 kommunizieren, wenn er Nginx einen Request schickt, leitet Nginx ihn an Gunicorn weiter, der der aktuelle Webserver ist. Gunicorn übersetzt die Anfrage in etwas, das Spring verstehen kann. Spring arbeitet mit MySQL und sendet Gunicorn die Antwort, der diese weiter an Nginx befördert. Danach schickt Nginx dem Client die Antwort.

Warum muss das System so kompliziert sein? Spring ist lediglich ein Framework, das den Entwicklern einen Großteil der Webentwicklung abnimmt und ihnen ermöglicht, sich auf die eigentliche Anwendungsentwicklung zu konzentrieren, ohne das Rad neu erfinden zu müssen. Es bietet jedoch keinen aktuellen Webserver und weist in Bezug auf Sicherheit gewisse Schwächen auf, weshalb ein aktueller Webserver wie Gunicorn erforderlich ist, um das System für den produktiven Einsatz geeignet zu machen. Die schnelle Entwicklung mit Spring hat allerdings ihren Preis: Manchmal kann Spring nicht schnell genug sein, was bedeutet, dass Benutzer möglicherweise einige Minuten warten müssen, bis eine Seite vollständig geladen ist. Um diesem Performance-Problem entgegenzuwirken, kommen Threads und Lastverteilung ins Spiel. Durch den Einsatz von Threads können Anwendungen, die Multicore- oder Multi-CPU-Systeme nutzen möchten, Daten und Aufgaben in parallele Teilaufgaben aufteilen. Dabei übernimmt die zugrunde liegende Architektur die Verwaltung und Ausführung der Threads, entweder simultan auf einem Kern oder parallel auf mehreren Kernen. Lastverteilung (Load Balancing) wird eingesetzt, um große Mengen von Anfragenauf mehrere parallel arbeitenden Systemee zu verteilen und somit die Gesamtverarbeitung effizienter zu gestalten. Nginx ist eine einfache Antwort auf die Herausforderungen der Lastverteilung und unterstützt Gunicorn bei der effektiven Verteilung der Anfragen. Durch den Einsatz von Threads und Lastverteilung kann das System eine bessere Leistung und Skalierbarkeit erreichen, um den Anforderungen großer Anwendungen gerecht zu werden. So wird gewährleistet, dass das System effizient arbeitet und eine optimale Benutzererfahrung bietet.

4.1.2. Nginx

Nginx ist ein Reverse-Proxy, der vor den Backend-Anwendungen platziert wird und Client-Anfragen an diese weiterleitet. Eine Aufgabe kann ein Reverse-Proxy erfüllen, der die Anfragen entgegennimmt, um die Geschwindigkeit zu verbessern und funktionell zu erweitern. Die an den Client zurückgegebenen Antworten sehen aus, als kämen sie vom Webserver selbst. Damit kann Reverse-Proxy die Existenz und die Eigenschaften von Ursprungsservern verbergen. Außerdem verwendet die Webseite ihn zusammen mit anderen Techniken, um die Last auf die internen Server zu verteilen. Reverse-Proxy kann statische Inhalte auf dem Cache speichern, um die Belastung dieser internen Server und des internen Netzes weiter zu verringern. Es ist auch üblich, dass Komprimierung oder TLS-Verschlüsselung zwischen dem Client und dem Reverse-Proxy eingebaut werden. Bei sicheren Webseiten kann es vorkommen, dass die Webseite die TLS-Verschlüsselung nicht selbst durchführt, sondern diese Aufgabe an einen Reverse-Proxy überträgt. Es gibt eine Technik, die "Spoon feeding" heißt, deswegen kann eine dynamische Seite (HTML, CSS, JS, usw.) einmal generiert und an den Proxy Server geschickt werden. Danach sendet der Proxy Server sie an den Client zurück und muss die aktuelle Webseite nicht öffnen, das heißt die Serverressourcen

werden entlastet. Mittels Reverse-Proxy kann ein Webserver, der über keine Authentifizierung verfügt, eine Zugangsauthentifizierung hinzufügen.

Um nginx nutzen zu können, müssen Sie das Paket nginx installieren und den nginx.service starten. Anschließend müssen Sie die Konfigurationsdatei /etc/nginx/nginx.conf anpassen. Ein Beispiel für diese Datei finden Sie im Repository unter docker/nginx/nginx.conf.

```
# Source: https://docs.nginx.com/nginx/admin-guide/web-server/ worker_processes auto;
worker_cpu_affinity auto;
```

```
events {
        multi_accept on;
        accept_mutex on;
        worker_connections
        1024;
}

http {
        sendfile on;
        sendfile_max_chunk
        1m; ssl_ciphers
"EECDH+CHACHA20:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:EECDH+3DES:RSA+3DES:!M D5";
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
        add_header Strict-Transport-Security "max-age=15768000; includeSubDomains" always; include mime.types;
        default_type application/octet-stream; # fallback in case we can't determine a
type
        client_max_body_size 5G;
        error_log /var/log/nginx/error.log warn; access_log
        /var/log/nginx/access.log combined; proxy_cache_path
        /tmp/cache keys_zone=mycache:10m;
        #limit_conn_zone $binary_remote_addr zone=ip_addr:10m; # Let's prevent DoS

        upstream
                gunicorn.djang
                o { least_conn;
                server web:8000;
                #server i7_teamverwaltung-web-3:8000 backup;
        }

        server {
                # if no Host match, close the connection to prevent host spoofing listen 80
                default_server;
                return 444;
        }

        server {
                listen 80;
                listen [::]:80;
                server_name iseproject01.informatik.htw-dresden.de; keepalive_timeout 5;
                return 301 https://$host$request_uri;
        }

        server {
                listen 443 ssl http2;
                listen [::]:443 ssl
                http2;
                server_name iseproject01.informatik.htw-dresden.de; proxy_cache
                mycache;
                keepalive_timeout 5; ssl_certificate
                ssl/server.crt;
```

```
ssl_certificate_key ssl/server.key;
#limit_conn ip_addr 20;

location / {
proxy_set_headerHost $host; proxy_set_headerX-Forwarded-Proto $scheme;
proxy_set_headerX-Forwarded-For $proxy_add_x_forwarded_for; proxy_redirectoff;
proxy_connect_timeout300;
proxy_send_timeout300;
proxy_read_timeout300; proxy_cache_valid any 1m; proxy_cache_min_uses 3;
proxy_pass https://gunicorn.django;
}
location /static { autoindex on;
alias /var/www/static/;
}
}
}
```

Nginx ist standardmäßig auf Port 80 konfiguriert. Wenn ein Client eine Anfrage an diesen Port sendet, gibt Nginx den Statuscode 301 zurück, was "Move Permanently" bedeutet. In diesem Fall befindet sich die Webseite auf Port 443 (HTTPS). Die Einstellung "worker_processes" legt fest, wie viele Verbindungen Nginx akzeptiert und wie viele Prozessoren es nutzen kann. Normalerweise übernimmt Nginx die Dateiübertragung selbst, indem es die Datei in den Puffer kopiert, bevor sie gesendet wird. Die Einstellung "sendfile" ermöglicht jedoch das direkte Kopieren, ohne dass die Daten zuerst in den Puffer kopiert werden müssen. Um sicherzustellen, dass eine schnelle Verbindung den Worker-Prozess nicht vollständig auslastet, kann die Einstellung "sendfile_max_chunk" verwendet werden, um die maximale Datenmenge zu begrenzen, die in einem einzigen "sendfile()"-Aufruf übertragen wird. Der Lastausgleich des Datenverkehrs auf eine Gruppe von Servern oder Diensten wird in der Konfiguration mit "upstream" definiert. Standardmäßig wird der Round-Robin-Algorithmus verwendet, der die Anfragen gleichmäßig auf die Server verteilt, wobei die Servergewichtung berücksichtigt wird. Im Beispiel wird jedoch die Methode "Least Connections" verwendet, bei der eine Anfrage an den Server mit der geringsten Anzahl aktiver Verbindungen gesendet wird, unter Berücksichtigung der Servergewichtung. Wenn ein Server vorübergehend aus dem Lastausgleich entfernt werden muss, kann er mit "down" markiert werden, z.B.: "server i7_teamverwaltung-web-2:8000 down;". Anfragen, die an diesen Server gerichtet werden, werden automatisch an den nächsten Server in der Gruppe weitergeleitet. Der Server "i7_teamverwaltung-web-3" ist als Backup-Server gekennzeichnet und erhält nur dann Anfragen, wenn beide anderen Server nicht verfügbar

sind. Die Einstellung "slow_start", z.B.: "slow_start=30s", ermöglicht einen langsamen Start des Servers nach einem Wiederherstellungsprozess, um zu verhindern, dass der Server durch eine Flut von Verbindungen überlastet wird. Dies könnte zu einer Zeitüberschreitung führen und dazu führen, dass der Server erneut als ausgefallen markiert wird. Die Optionen "proxy_cache_valid any 1m" und "proxy_cache_min_uses 3" bedeuten, dass Anfragen nur für eine Minute zwischengespeichert werden und erst dann zwischengespeichert werden, wenn die gleiche Anfrage dreimal gestellt wurde. Um einen HTTPS-Server einzurichten, fügen Sie in der nginx.conf den Parameter "ssl" hinzu und geben Sie dann die Speicherorte der Serverzertifikats- und privaten Schlüsseldateien an. Das Zertifikat kann von der HTW bezogen werden.

4.1.3. Java Spring Boot / Gunicorn

Java Spring ist ein Open-Source-Framework, das die Entwicklung von Java-Anwendungen vereinfacht. Es bietet eine umfassende Infrastruktur und Abstraktionen für die Implementierung von Geschäftslogik, Datenbankzugriff, Sicherheit und vielem mehr. Mit Spring können Entwickler effizientere und skalierbare Anwendungen erstellen.
Nginx ist die Schnittstelle zur Außenwelt. Es dient dazu, statische Dateien wie Bilder und CSS direkt aus dem Dateisystem und dem Cache bereitzustellen. Allerdings kann Nginx nicht direkt mit Spring kommunizieren. Es benötigt eine Komponente, die Spring ausführt, Anfragen von der Webanwendung entgegennimmt, Antworten generiert, auf mehrere Anfragen gleichzeitig reagiert und die Last verteilt. Hier kommt Gunicorn ins Spiel. Gunicorn übernimmt die Verarbeitung der Anfragen. Es betreibt einen Pool von Threads, die die Anfragen bearbeiten, und gibt die Antworten an Nginx zurück. Nginx fungiert als Vermittler zwischen dem Client und Gunicorn. Wenn eine Anfrage dynamischer Natur ist, leitet Nginx sie an Gunicorn weiter, der sie verarbeitet und die Antwort wiederum an Nginx zur Weiterleitung an den ursprünglichen Client zurückgibt.

```
import multiprocessing
 workers = multiprocessing.cpu_count() * 2 + 1
 worker_connection = 1000
```

4.2. Dokumente

- Docker (Linux)
- Docker Desktop
- Nginx
- Gunicorn
- SSL/TLS