Entwurf und Realisierung eines Capture the Flag Core Systems

Bachelorarbeit
zur Erlangung des Grades *Bachelor of Science*

an der
Hochschule Niederrhein
Fachbereich Elektrotechnik und Informatik
Studiengang *Informatik*

vorgelegt von
Robert Hartings
Matrikelnummer: 1164453

Datum: 4. August 2020

Prüfer: Prof. Dr. Jürgen Quade
Zweitprüfer: Prof. Dr. Peter Davids

Eidesstattliche Erklärung

Name:	Robert Hartings
Matrikelnr.:	1164453
Titel:	Entwurf und Realisierung eines Capture the Flag Core Systems
Englischer Titel:	Design and Implementation of a Capture the Flag Core System

Ich versichere durch meine Unterschrift, dass die vorliegende Arbeit ausschließlich von mir verfasst wurde. Es wurden keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt.

Die Arbeit besteht aus _____ Seiten.

_____	_____
Ort, Datum	Robert Hartings

Inhaltsverzeichnis

1 Einleitung

Das Thema IT Sicherheit ist besonders in den letzten Jahren relevant geworden. Viele Firmen suchen Fachkundige [it-19], die die bestehenden und neu designten Systeme und Programme auf Sicherheitslücken prüfen und Lösungsvorschläge zu deren Behebung präsentieren. Auch werden Experten gesucht, die die im Unternehmen bestehenden Prozesse prüfen und neue zum Umgang mit Sicherheitslücken entwerfen.

Einen Mangel an IT-Sicherheit in privaten und öffentlichen Unternehmen, beziehungsweise ein fehlendes Konzept zur Vorbeugung, Erkennung und Abwendung von Sicherheitslücken sieht man in jüngster Vergangenheit deutlich, nachdem beispielsweise diverse Universitäten wie Gießen [Sch20], Maastricht [WDR19] und Bochum [Ruh20] Ende 2019 Ziele von Hackerangriffen geworden sind. Aber nicht nur Universitäten sind betroffen, auch sind neben Gerichten [HHH20], Stadtverwaltungen [BH20] und Krankenhäusern [Wel19] bereits der Deutsche Bundestag von Hackern angegriffen und kompromittiert worden. [FM20]

In der Studie „Wirtschaftsschutz in der digitalen Welt“ vom 06. November 2019 des Bundesverbandes Informationswirtschaft, Telekommunikation und neue Medien e.V. Bitkom wird die aktuelle Bedrohungslage durch Spionage und Sabotage für deutsche Unternehmen untersucht. Grundlage der Studie ist eine Befragung von 1070 (2019) bzw. 1074 (2015) Unternehmen.

Waren 2015 noch 28% *vermutlich betroffen* und 51% *betroffen* haben 2019 schon 13% *vermutlich betroffen* und 75% *betroffen* angegeben. Die Studie zeigt, dass die Zahl von Datendiebstahl, Industriespionage oder Sabotage steigend ist. Der Schaden wird durch die Unternehmen auf 102,9 Milliarden Euro pro Jahr beziffert. [BN19]

Dass diese Thematik auch im Lehrbetrieb angekommen ist, sieht man an neu startenden Studiengängen wie dem Bachelorstudiengang Cyber Security Management der Hochschule Niederrhein, der zum Wintersemester 2020/21 startet. [Hoc20]

Es ist zu erwähnen, dass die Hochschulen sich bereits mit dem Thema auseinandersetzen. So beschäftigt sich an der Hochschule Niederrhein das Institut für Informationssicherheit Clavis besonders mit Themen rund um das Informationssicherheitsmanagement, gestaltet aber auch Inhalte zur Vulnerabilität von (kritischer) Infrastruktur und Hacking. Das Ziel von Clavis ist die Erhöhung der Informationssicherheit von Organisationen im regionalen Umfeld der Hochschule. [Hoc]

Auch hat die Hochschule Niederrhein das Thema IT-Sicherheit bereits in ihren Lehrplan für die Studiengänge Informatik und Elektrotechnik am Fachbereich 03 Elektrotechnik und Informatik aufgenommen. So werden dort im fünften Semester in der Veranstaltung IT-Security

grundlegende Kompetenzen zum Thema IT-Sicherheit vermittelt, welche einem allgemeinen Anspruch genügen. [Hoc19]

Begleitend zu dieser Veranstaltung werden drei Versuche im Rahmen des Praktikums durchgeführt. Diese sollen den Studierenden praktische Erfahrungen ermöglichen und ein breites Bewusstsein schaffen, indem die Studierenden sowohl in die Rolle des Angreifers als auch die des Verteidigers schlüpfen.

Im zweiten Versuch namens „Catch me, if you can“ findet ein Vergleichswettbewerb zwischen den teilnehmenden Studierendenteams statt.

Die Aufgabe der Teams besteht darin, festgelegte Programme bzw. Dienste abgesichert bereitzustellen, geheime Informationen sowohl auf dem eigenen Rechner als auch auf den Rechnern der anderen Teams zu finden und Schwachstellen abzusichern, um so zu verhindern, dass andere Teams an die eigenen geheimen Informationen gelangen. [Sos10, S. 2]

Die geheimen Informationen sind normalerweise Passwörter oder private Bilder und werden im Versuch durch sogenannte Flags repräsentiert. Eine Flag ist eine generierte Zeichenfolge mit fester Länge.

1.1 Motivation

Diversen Meldungen über erfolgreiche Angriffen auf Unternehmen und öffentliche Körperschaften sowie die Veranstaltung IT-Sicherheit im fünften Semester, besonders herauszuheben ist hier das Praktikum[1], haben mich auf das Thema IT-Sicherheit aufmerksam gemacht.

Die zunehmenden Vorfälle zeigen, dass ein breites Bewusstsein für IT-Sicherheit geschaffen werden muss. Das Praktikum übernimmt hierbei eine wichtige Aufgabe.

Das Programm, das das Praktikum überwacht, ist bereits 10 Jahre alt und fordert Modernisierung, Überarbeitung und Erweiterung. So gibt es beispielsweise heute bessere Möglichkeiten, die Darstellung (Web-Oberfläche) zu realisieren.

1.2 Aufgabenstellung

Das Praktikum wird durch ein Auswertungs- und Überwachungssystem begleitet, das eine objektiv nachvollziehbare Bewertung vornehmen kann und die in den Bewertungsprozess eingeflossenen Parameter dokumentiert. [Sos10, S. 2]

Ziel meiner Arbeit ist die Modernisierung und Verbesserung dieses Auswertungs- und Überwachungssystems. Anforderung hierbei war, dass das neue System die gleichen Basisfunktionalitäten wie das Altsystem erfüllen muss.

[1] Praktikum ist hierbei mit einer Pflichtübung vergleichbar

In der einführenden Betrachtung (Kapitel 2) wird der aktuelle Stand des Systems, Schnittstellen zwischen Server und Client sowie der Begründung für die Veränderung dargelegt. Aus dieser einführenden Betrachtung werden dann Anforderungen abgeleitet.

Im folgenden Kapitel 3 werden Entwürfe für die verschiedenen, neu designten Komponenten des Servers erstellt.

Anhand der abgeleiteten Anforderungen und des Entwurfs der verschiedenen Komponenten werden im Kapitel 4 diverse Technologien diskutiert und passende ausgewählt.

Die Implementierung des Entwurfs mit den gewählten Technologien wird im Kapitel 5 beschrieben.

Eine kritische Auseinandersetzung mit dem Ergebnis dieser Arbeit folgt und es werden Aussichten für mögliche Veränderungen und Erweiterungen gegeben.

Aus diesem Aufbau der Arbeit ergeben sich die folgenden Aufgaben:

- Analyse des vorliegenden Systems
- Analyse der Voraussetzungen
- Ableitung von Anforderungen
- Entwurf der Architektur
- Entwurf der einzelnen Komponenten
- Diskussion einzusetzender Technologien
- Implementierung des Entwurfs
- Bereitstellung von Installations- und Bedienungsanleitung.

2 Analyse

In diesem Kapitel werden die Voraussetzungen im Labor vorgestellt, die derzeitige Implementierung des Auswertungs- und Überwachungssystems beleuchtet und kurz auf einen überwachten Client sowie dessen Schnittstellen zum System eingegangen.

2.1 Lehrveranstaltung IT-Sicherheit

Das Pflichtmodul IT-Sicherheit (ITS) ist in drei Veranstaltungen gegliedert. [Hoc19, S.30]

- Vorlesung (2 Semesterwochenstunden)
- Übung (1 Semesterwochenstunde - freiwillig)
- Praktikum (1 Semesterwochenstunde - verbindlich)

Vorlesung

Die Vorlesung wird im wöchentlichen Turnus angeboten und behandelt grundlegendes Wissen zur IT-Sicherheit unter anderem in den Bereichen Gefährdung, Gegenmaßnahmen, aber auch im Bereich rechtliche Gegebenheiten. Es werden Beispiele aufgezeigt, bei denen die angesprochenen Themen gar nicht oder in einem ungenügenden Zustand umgesetzt worden sind. Die Vorlesung wird von den Veranstaltungen *Übung* und *Praktikum* ergänzt.

Übung

Die Übungen sind freiwillig und werden im zweiwöchentlichen Turnus á 2 Stunden angeboten. Diese ermöglichen es den Studierenden, den durch die Vorlesung und das Selbststudium vermittelten Stoff zu vertiefen und zu festigen. Auch können dort praktische Erfahrungen gesammelt werden, von denen die Studierenden unter anderem im Praktikum profitieren können.

Praktikum

Das Praktikum findet im monatlichen Turnus (3x im Semester á 4 Stunden) statt. Es ist in drei Laborversuche untergliedert. Nach Bestehen aller Versuche erhalten die Studierenden ihre Klausurzulassung. Jeder Versuch des Praktikums muss vorbereitet werden, dazu erhalten die Studierenden vor dem Versuch ein Hackit[1]. Nur mit erfolgreichem Absolvieren des Hackits ist es möglich, am nächsten Versuch teilzunehmen. [Qua17]

2.2 Ausstattung Labor

Das Praktikum wird im Labor für Echtzeitsysteme (EZS Labor) der Hochschule Niederrhein durchgeführt.

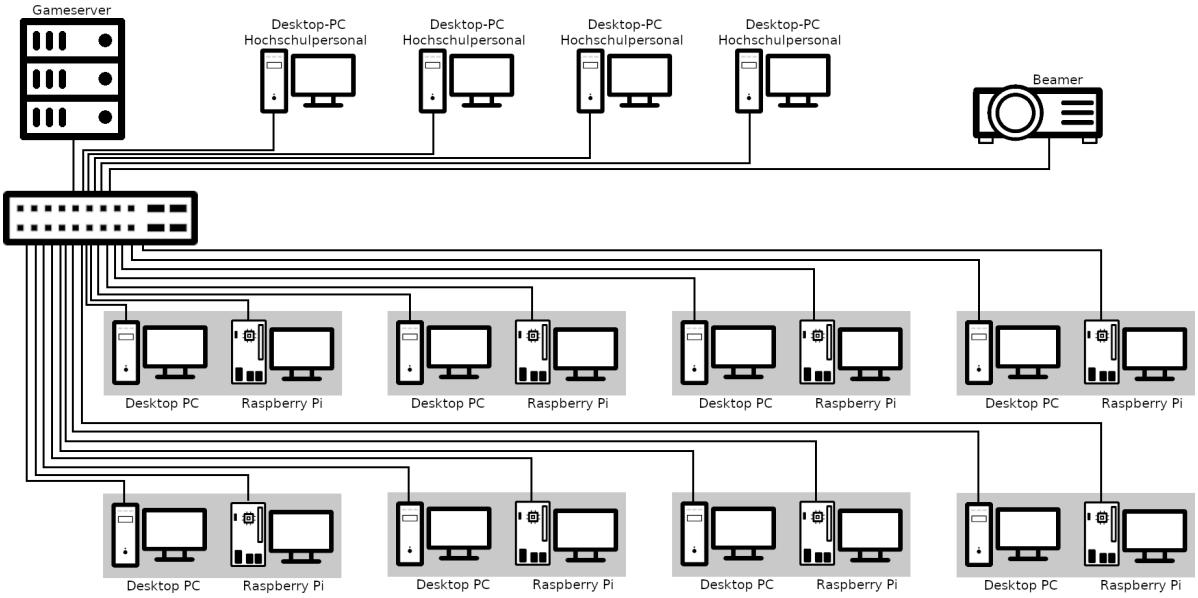

Abbildung 2.1: Übersicht über die Laborausstattung (Netzwerktopologie)

Wie in der Abbildung 2.1 dargestellt, ist das Labor mit acht Gruppenarbeitsplätzen für Studierende sowie Arbeitsplätzen für die Mitarbeitenden ausgestattet. Ein Arbeitsplatz kann zu einem neunten Gruppenarbeitsplatz umfunktioniert werden.

An einem Gruppenarbeitsplatz (in der Abbildung 2.1 grau hinterlegt) können 2 Studierende gleichzeitig arbeiten, da dieser mit einem leistungsfähigen Desktop-PC und einem Raspberry Pi[2] sowie den dazugehörigen Peripheriegeräten (Maus, Tastatur & Monitor) ausgestattet ist.

[1] Knobelaufgabe aus dem Bereich IT-Sicherheit / Hacking
[2] Einplatinencomputer mit der Größe einer Kreditkarte

Als Betriebssystem wird auf den Desktop-PCs Ubuntu[3] und auf den Raspberry Pis Raspbian[4] verwendet.

Auf den Desktop-PCs ist die Software VirtualBox der Firma Oracle installiert. Diese ermöglicht das Virtualisieren eines weiteren Rechners. Diese virtuelle Maschine wird Gast genannt und beheimatet die Dienste und Anwendungen, die während des Versuchs benötigt werden. Durch die Nutzung der Virtualisierung muss die Software nicht auf dem Wirtsystem installiert werden. So ist dieses auch für andere Versuche im Rahmen der Lehrveranstaltung nutzbar, ohne dass auf Inkompatibilitäten von Softwares der verschiedenen Versuche geachtet werden muss. Auch bietet VirtualBox die Möglichkeit, sogenannte Snapshots anzulegen. Ein Snapshot ist eine Momentaufnahme eines aktuellen Systemzustandes. Dieser stellt eine Komplettsicherung des Systems dar und ermöglicht es, die Systeme auf den gleichen Stand zu bringen. [Ora20c] Dies wird benötigt, um eine Vergleichbarkeit zwischen den Studierenden zu gewährleisten.

Die teilnehmenden Studentengruppen werden in den Softwarekomponenten und in der Auswertung durch die IP-Adresse ihres Gast-Systems identifiziert.

Neben diesen Computern steht ein Linux Server zur Verfügung, auf dem das Auswertungs- und Überwachungssystem betrieben wird.

Alle Computer, auch die Gastsysteme der Studentengruppen, sind untereinander über ein Netzwerkswitch via Ethernet verbunden.

Außerdem steht ein Beamer zur Verfügung, auf dem die aktuelle Spielübersicht dargestellt werden kann.

2.3 Versuch: Catch me, if you can

Der zweite der drei Versuche „Catch me, if you can“ wird im Rahmen eines Wettbewerbs zwischen den teilnehmenden Studierendenteams ausgetragen. Der Wettbewerb ist an ein Capture the Flag (CTF) angelehnt. Bei einem klassischen CTF erhält der Spielende durch das Lösen von Aufgaben einen bestimmten Text. Dieser wird Flag genannt. Die Aufgaben können das Lösen einer Art Schnitzeljagd, eine einfache Programmierung, aber auch das Hacken mehrerer entfernter Rechner umfassen. Anders als beim klassischen CTF werden bei „Catch me, if you can“ die Flags auf allen teilnehmenden Systemen verteilt. [Tan20] Die Studierenden können diese durch das Analysieren ihres eigenen Gastsystems sowie durch den Angriff auf fremde Gastsysteme erhalten. Besonderheit hierbei sind die Strafpunkte für den Verlust einer Flag an gegnerische Studierendenteams. Wie beim klassischen CTF können die Studierenden Flags und Punkte durch zentrale Aufgaben erhalten.

[3] Ubuntu ist eine freie Linux Distribution auf Basis von Debian
[4] Abwandlung von Debian für den Raspberry Pi

Der Versuch ist in drei Phasen untergliedert.

1. Vorbereitung
2. Wettbewerb
3. Abschluss

Vorbereitung

Die Studierenden erhalten circa 30 Minuten Zeit, um ihr Gastsystem in Betrieb zu nehmen und sich mit diesem vertraut zu machen. Hierbei sollten die Schwachstellen in den vorhandenen Diensten abgesichert und der Zugriff durch andere Studierende verhindert werden. Während dieser Zeit dürfen die Studierenden andere Systeme nicht angreifen. Auch ist es möglich, in dieser Zeit Flags auf dem eigenen System der Studierenden zu suchen. Da der Ablageort der Flags auf allen Systemen gleich ist, kann diese Information im Spielverlauf bei einem eigens initiierten Angriff dazuführen, schneller Flags einzubringen.

Wettbewerb

Die Wettbewerbsphase selbst dauert circa 140 Minuten. In dieser Zeit sind Angriffe auf fremde Gastsysteme erlaubt und ausdrücklich erwünscht. Eine weitere Absicherung ist ebenfalls möglich. Das System sollte auf fremde Aktivitäten hin überwacht werden. Diese Aktivitäten sollten schnellstmöglich unterbunden werden, da die Angreifer Flags entwenden können und so dem Team Strafpunkte einbringen. Auch kann die Zeit für die Lösung von zur Verfügung stehenden Challenges sowie die Nutzung des Flagshops genutzt werden. Der Flagshop sowie die Challenges werden im nächsten Kapitel aufgegriffen.

Abschluss

Nach Ende der Wettbewerbsphase müssen die Studierenden ihre Angriffe einstellen und eine weitere Flagabgabe ist nicht mehr möglich. Die Studierenden erstellen für ihren anzufertigen Versuchsbericht einen Screenshot der Punkteübersicht. Eine Nachbesprechung ist optional und auf maximal 30 Minuten begrenzt.

Während des Wettbewerbs gelten die aufgelisteten Regeln. Es handelt sich hierbei nur um einen Auszug der für die Bachelorarbeit relevanten Regeln:

- Der Gameserver darf nicht angegriffen werden
- Es dürfen nur die Gastsysteme angegriffen werden
- Das Passwort des Logins *gamemaster* darf nicht zurückgesetzt werden
- Der SSH-Server muss für alle erreichbar sein

- Flags dürfen nicht modifiziert oder gelöscht werden
- Sämtliche Dienste müssen für den Gameserver erreichbar bleiben
- ICMP-Pakete (ping) dürfen nicht blockiert werden.

[Qua17, S.9] [Sos10, S.10-11]

2.4 Systemkomponenten

2.4.1 Komponenten des Servers

Im Folgenden werden die verschiedenen Komponenten des Auswertungs- und Überwachungssystems in der derzeitigen Implementierung untersucht. Dabei werden Rückschlüsse auf Anforderungen gezogen sowie Schwachstellen herausgearbeitet.

Scanner

Der Scanner prüft in regelmäßigen Abständen die auf den Gastsystemen der Studierenden installierten Dienste und speichert das Ergebnis ab. Die Abstände können beim Starten des Spieles eingestellt werden. Die folgenden Dienste werden pro Team geprüft.

ScanUp Die Aufgabe dieses Scans besteht darin zu prüfen, ob das Gastsystem noch für den Server erreichbar ist. Sollte das Gastsystem nicht erreichbar sein, wird hierfür ein Strafpunkt vergeben. Aus technischer Sicht wird das Linux-Kommando *ping* verwendet. Anhand des Rückgabewertes kann nachvollzogen werden, ob der Server das Gastsystem erreichen konnte.

ScanBubble Auf dem Gastsystem läuft ein selbst programmierter Bubble Server, welcher Flags unter Nutzung des Telnet Protokolls bereitstellt. Nachdem eine Flag abgeholt wurde, stellt der Dienst für eine bestimmte Zeit (Timeout) keine weiteren Flags bereit. Der Bubble Server nimmt Anfragen über den Port *12321* für unverschlüsselte Flags und Port *12322* für verschlüsselte Flags entgegen. Die Scan-Operation überprüft, ob eine Telnet-Verbindung zum Port *12321* möglich ist, indem die Operation eine Telnet-Verbindung öffnet und prüft, ob die Verbindung erfolgreich war.

ScanWebUp Jedes Gastsystem stellt unter Zuhilfenahme eines Apache-Webservers und php-Dateien Webseiten und Daten bereit, die über einen Webclient abgerufen werden können. Dazu müssen auf Port *80* der HTTP- und auf Port *443* der HTTPS-Dienst laufen und erreichbar sein. Dieses verifiziert die Scan-Operation, indem sie eine Socket-Verbindung zu den Ports *80* und *443* öffnet und das Ergebnis prüft.

ScanSQLInjectUp Dieser Scan prüft, ob die Login-Seite des Teams, auf der die SQL-Injection-Schwachstelle implementiert ist, erreichbar und benutzbar ist. Die Operation sendet hierzu eine valide Kombination aus Nutzername und Passwort an den Webserver. Das Ergebnis wird dann mit dem erwarteten Resultat verglichen.

ScanSQLInjectSave Ähnlich wie bei der Operation ScanSQLInjectUp (2.4.1) wird geprüft, ob Ergebnis und Erwartung übereinstimmen. Besonderheit hierbei ist, dass statt einer validen Kombination aus Nutzernamen und Passwort eine sogenannte SQL-Injection (wird im Unterabschnitt 2.4.2 genauer erläutert) im Nutzernamen übergeben wird. Sollte die Anfrage alle gespeicherten Nutzerdaten zurückgeben, ohne dass eine Authentifizierung stattfindet, ist die SQL-Injection weiterhin möglich.

ScanXSSSave Diese Scan-Operation prüft, ob die auf dem Gastsystem implementierte XSS-Schwachstelle behoben wurde. Dazu wird die Webseite mit präpariertem Inhalt aufgerufen. Auf die Vorgehensweise wird im Unterabschnitt 2.4.2 eingegangen. In der Rückgabe wird geprüft, ob dieser ungefiltert auf der Webseite zu finden ist. Sollte dies der Fall sein, ist die XSS-Schwachstelle nicht oder unzureichend von den Studierenden abgesichert worden.

ScanSQLSave Bei diesem Scan wird kontrolliert, ob der Login mit dem auf allen Systemen voreingestellten Passwort *toor* für den SQL-Account *root* möglich ist. Ist dieser möglich, haben die Studierenden dieses unsichere Passwort nicht geändert. Des Weiteren wird geprüft, ob die Nutzerdaten in der htaccess-Datei, die die phpMyAdmin-Anwendung schützen soll, geändert worden sind.

ScanFTPSave Auf dem Client System läuft ein FTP-Server, der ohne Login (Nutzername & Passwort) Daten bereitstellt. Der Scan prüft, ob ein sogenannter anonymer Login möglich ist, indem eine FTP-Verbindung ohne Login aufgebaut wird. Sollte die Verbindung erfolgreich sein, ist der anonyme Login immer noch nicht deaktiviert worden.

ScanTelnetSave Ein Telnet-Server horcht auf Verbindungen auf Port *23*. Da dieser Dienst nicht benötigt wird, soll er durch die Studierenden abgeschaltet oder deinstalliert werden. Die Scan-Operation prüft, ob eine Verbindung mithilfe des Telnet-Protokolls auf Port *23* möglich ist. Dazu wird eine Verbindung zu Port *23* aufgebaut und das Resultat geprüft.

Generierung von Flags

Derzeitig erfolgt die Generierung der Flags sowohl auf den Gastsystemen als auch auf dem Auswertungs- und Überwachungssystem. Dies ist notwendig, da ansonsten eine Überprüfung der Gültigkeit der Flags und Verrechnung der Punkte nicht durchgeführt werden kann. Die Flags werden durch einen Algorithmus generiert. Dieser erzeugt pro Team eine bestimmte Anzahl an Flags.

Dazu wird die Flag mithilfe der Streuwertfunktion (Hashfunktion) MD5 und der Eingabe, einem sogenannten Seed, berechnet. Eine Hashfunktion bildet aus einer Eingabe variabler Länge eine Ausgabe mit einer festen Länge. Bei identischer Eingabe wird immer der gleiche Ausgabewert berechnet. Darüber hinaus ist es bei einer guten Hashfunktion nicht möglich, von der Ausgabe auf den Eingabewert zu schließen. [MOV96]

Der in der Anwendung genutzte Seed setzt sich aus der Verkettung von *Salt*, *IP-Adresse*, dem String „*Aufgabe*“ und einem *Zähler* zusammen.

```
1 seed = "WS2019192.168.87.11Aufgabe1"
2 hash = md5(seed)
3 hash = 7072D70B3D47E8516056A8B777655174
```

Listing 2.1: Beispiel eines Seed und seines Hashs

Ein Salt wird benötigt, um den Flags eine Lebenszeit zu geben. In der derzeitigen Implementierung enthält der Salt das aktuelle Jahr sowie das jeweilige Semester. So sind nur Flags des aktuellen Semesters gültig und werden vom Auswertungs- und Überwachungssystem akzeptiert. Einer Verwendung von Flags aus vorherigen Semestern wird somit effektiv vorgebeugt.

Die IP-Adresse stellt hierbei den Bezug zum jeweiligen Team dar.

Der String „Aufgabe“ wird als Geheimnis verwendet, um das Fälschen von Flags zu erschweren und bestenfalls zu verhindern.

Damit pro Team mehrere eindeutige Flags generiert werden können, wird ein sogenannter Zähler genutzt. Dieser Zähler ist auf 0 initialisiert und wird pro generierter Flag um eins erhöht, bis die benötigte Anzahl an Flags generiert ist. [Sos10, S.48]

Webserver

Der Webserver stellt das GUI (Graphical User Interface) für die Studierenden und betreuenden Personen dar. Hier kann der aktuelle Punktestand eingesehen werden. Auch wird in dem GUI dargestellt, welches Team welchen Service abgesichert hat, inklusive der Negativpunkte für nicht abgesicherte Dienste, und wie viele Strafpunkte das jeweilige Team erhalten hat.

Neben diesen Darstellungen befinden sich auf dem Server ein sogenannter Flagshop und diverse Challenges, mit denen Studierende weitere Flags erhalten können.

Die betreuenden Personen haben die Möglichkeit, über das Web-GUI ein neues Spiel anzulegen und das Spiel zu starten oder zu stoppen. Auch kann von dem Spiel ein Backup erstellt werden. Neben diesen Funktionen zur Spielsteuerung können an die Teams Strafen für unfaires oder regelverletzendes Verhalten verteilt werden. Diese nehmen direkten Einfluss auf die Punkte des jeweiligen Teams. Außerdem besteht die Möglichkeit, weitere Benutzer für das Administrationsinterface zu registrieren.

Flagshop Der Flagshop ermöglicht es den Studierenden, weitere Flags mit ihren Punkten zu kaufen. Der Kauf von Flags lohnt sich, da die verkauften Flags mehr Punkte bringen als sie kosten. Um einen Einkauf im Flagshop durchzuführen, müssen die Teams vorher einen Account erstellen. Die Registrierung erfragt neben dem benötigten Benutzernamen und Passwort auch für den Flagshop irrelevante Daten ab. Diese ähneln persönlichen Informationen, die bei den meisten Onlineshops angegeben werden müssen. Das Format, hier die Repräsentation als Zahl oder String sowie die Länge, und die Erforderlichkeit der Daten werden nur im HTML-Formular festgelegt. Durch eine Manipulierung des Formulars kann dieses mit nicht konformen oder nicht vorhandenen Daten abgesendet werden. Für jede der nicht vorhandenen oder nicht konformen Informationen erhält der Studierende eine Flag. Daneben wird die Güte des angegebenen Passwortes anhand von Länge und Anzahl an Sonderzeichen, Groß- und Kleinbuchstaben sowie Ziffern bewertet und mit Flags belohnt.

Nach der Registrierung können die Studierenden sich für ihre Punkte Flags kaufen. Dazu stehen zwei Pakete mit 8 bzw. 6 Flags für den Preis von jeweils 4 Punkten pro Paket zur Verfügung. Dieser Preis kann auf zwei Arten reduziert werden. Einmal müssen sich die beiden Pakete gleichzeitig im Warenkorb befinden und deren Identifikationsnummern (ID) müssen auf nicht vorhandene Nummern gesetzt werden. Die Manipulation resultiert in einem reduzierten Preis von 4 Punkten für beide Pakete. Dies ist extra im Flagshop einprogrammiert und soll die Studierenden auf Manipulation von IDs aufmerksam machen. Durch die zweite Art ist es möglich, die Pakete kostenlos zu erhalten. Dazu muss im Warenkorb das sogenannte *Hidden-Input-Feld*, in dem der aktuelle Preis des Warenkorbs gespeichert wird, auf 0 gesetzt werden. Dann berechnet der Flagshop für den Kauf einen Preis von 0 Punkten. [Abt16, S. 63]

Ein *Hidden-Input-Feld* wird in der Repräsentation eines HTML-Dokumentes nicht angezeigt, kann jedoch durch die Entwicklertools eines Browser betrachtet und verändert werden. [w3s]

Auf diese Weise ist es auch möglich, einen negativen Preis festzulegen und dem eigenen Team Punkte zuzuschreiben, da die derzeitige Implementierung nicht prüft, ob der von dem Nutzenden eingegebene Preis kleiner als 0 ist, sondern ob dieser gleich 0 ist. Bei korrekter Implementierung würde ein Preis kleiner 0 geprüft und korrigiert.

Challenges Derzeitig sind fünf Challenges implementiert, die vom System in zufälliger Reihenfolge an interessierte Teams verteilt werden. Eine abgeschlossene oder abgebrochene Challenge, durch das Neuladen der Webseite oder durch das Betätigen der Zurück-Taste, kann nicht wiederholt werden. Eine Challenge kostet 10 Punkte. Nach erfolgreichem Abschluss

einer Challenge gibt es 10 Punkte plus eine gewisse Anzahl an Punkten für das Absolvieren der Aufgabe. Die folgenden Challenges sind implementiert: [Abt16, S.19]

Aufgabe 1: robots.txt Die Studierenden sollen in dieser Challenge lernen, dass die *robots.txt* Datei keinen Zugriffsschutz darstellt. Diese bittet nur Suchmaschinen, die angegebenen Verzeichnisse und Dateien nicht zu indexieren. Aus der *robots.txt* Datei können Informationen zu versteckten Dateien und Verzeichnissen erhalten werden. Die Studierenden sollen über die *robots.txt* Datei einen vorhandenen Ordner finden. In diesem befindet sich die Lösung zur Challenge. [Abt16, S.19-20]

Aufgabe 2: JavaScript-Login-Bypass Bei dieser Challenge ist das benötigte Passwort zur Lösung der Challenge als Klartext im Quelltext versteckt. Das Verstecken wird derzeitig mit einer Meldung wie „Seitenquelltext deaktiviert“ ([Abt16]) und vielen Leerzeilen realisiert. Im Inspector von Firefox Version 78.0.1 und Chromium Version 83.0.4103.116 ist dies nicht möglich, da die Leerzeilen entfernt werden und das Passwort daher oben im Quelltext zu sehen ist. [Abt16, S.20]

Aufgabe 3: Form-Modification In dieser Challenge sollen die Studierenden verstehen, dass auch die Werte von Drop-Down-Menüs, Checkboxen und Radio-Buttons durch Manipulation auf nicht vorgegebene Werte geändert werden können. Deshalb müssen Nutzereingaben stets auch serverseitig überprüft werden, da hier die Regeln der Überprüfung nicht durch den Nutzenden verändert werden können.

Die Aufgabe besteht darin, einen bestimmten Login-Namen aus einem Drop-Down-Menü auszuwählen. Da der geforderte Name nicht in dieser Liste ist, müssen die Studierenden das HTML-Formular so manipulieren, dass er auswählbar ist. [Abt16, S.20]

Aufgabe 4: JavaScript-Substrings Das Passwort, das die Studierenden eingeben müssen, wird clientseitig mithilfe einer JavaScript Funktion geprüft. Damit das Passwort nicht im Klartext im Quelltext steht, wird dieses verschleiert. So werden drei Strings Zeichen für Zeichen verglichen. Sollte ein Zeichen in mindestens zwei der drei Strings übereinstimmen, dann gehört dieses Zeichen zum Passwort. Im Anschluss wird geprüft, ob das generierte Passwort gleich dem ist, das die Studierenden als Passwort genutzt haben. [Abt16, S.20]

Aufgabe 5: URL-Hex-Injection Die Studierenden sollen an geheime Informationen in einem Ordner gelangen, der nach einem Wert aus dem Hexadezimalsystem benannt ist. Anders als im Dezimalsystem ist die Basis 16 statt 10. Die Aufgabe soll zeigen, dass Ordner, die nach einem Hexadezimalwert benannten sind, auf diese Art und Weise nicht vor Zugriffen geschützt werden können, da das Präfix Zeichen % für Hexadezimalzahlen selbst durch einen Hexadezimalwert (%25) dargestellt werden kann. [Abt16, S.20]

Abgabe von Flags

Um Flags abgeben zu können, müssen die Studierenden sich mit ihren Zugangsdaten, die sie durch das Lösen des Hackits erhalten haben, in der Web-GUI anmelden. Dort ist es möglich, in einem Input-Feld eine Flag synchron abzugeben. Das bedeutet, dass nach jeder Abgabe die Webseite neu geladen wird. Eine Abgabe mehrere Flags ist gleichzeitig nicht möglich.

2.4.2 Komponenten des Clients

Da sich die Bachelorarbeit mit der Modernisierung des Auswertungs- und Überwachungssystems beschäftigt, sind nur die für die Bachelorarbeit wichtigen Komponenten des Clients beschrieben.

Webserver des Clients

Auf den Clients wird ein Webserver mit einigen extra implementierten Schwachstellen betrieben. Diese sollen während des Versuches durch die Studierenden behoben werden.

Der Webserver stellt eine Art Kundenbewertungsformular mit implementierter XSS Schwachstelle bereit. In der verwendeten Implementierung wird die Eingabe des Nutzers in das HTML-Dokument geschrieben. Durch die Schwachstelle wird eine Nutzereingabe ungefiltert in das HTML-Formular übernommen und ein potenziell schädlicher Code wird vom Browser verarbeitet. Dieser Code kann dann beispielsweise vertrauliche Informationen wie Cookies, Session-Tokens oder persönliche Daten auslesen und den Angreifern übermitteln. [RG07]

Eine weitere Schwachstelle stellt der sogenannte „Login zum Membersbereich“ mit der implementierten SQL-Injection Schwachstelle dar. Bei einer SQL-Injection versucht ein Angreifer den verwendeten SQL-Befehl so zu erweitern, dass dieser neben der eigentlichen Abfrage an die Datenbank auch einen vom Angreifer vorbereiteten SQL-Befehl ausführt. Die Erweiterung des SQL-Befehls erfolgt, indem die Eingabe, die in den SQL-Befehl aufgenommen wird, das Zeichen für das Ende des SQL-Befehls inklusive des vom Angreifer intendierten SQL-Befehls enthält. Über diesen Angriff können dann unerlaubterweise Daten ausgelesen werden, aber auch eine Löschung der Daten ist denkbar. [Bac04]

In der verwendeten Implementierung werden der Benutzername und das Passwort ungefiltert in den SQL-Befehl übernommen. Diese Schwachstelle lässt sich erst beheben, wenn die Gruppe die SQL-Injection bei sich selbst erfolgreich durchgeführt hat. [Abt16, S.27-29]

Neben diesen Schwachstellen gibt es eine Registrierung für den Flagshop. Diese erfordert einige Eingaben wie Name, Alter, Postleitzahl und vieles mehr. Die Eingaben sind im HTML-Formular als Pflicht markiert und haben eine Vorgabe in der Form. Ein Absenden ist ohne Angabe dieser Daten nicht möglich. Die Studierenden erhalten jedoch für jede nicht getätigte und für jede nicht der Form entsprechenden Angabe Flags nach der Registrierung. [Abt16, S.26] Dies ist möglich, da das HTML-Formular durch die Studierenden geändert werden kann

und der Server nur die Angaben bezüglich Passwort und Nutzername prüft. Diese beiden Angaben werden genutzt, um sich am Flagshop des Servers anzumelden und Flags zu erwerben. *(Siehe: Abschnitt 2.4.1 Flagshop)*

Außerdem stellt der Webserver eine Bildergalerie zur Verfügung. In dieser befinden sich zum Spielstart zwei normal erscheinende Bilder. Entgegen dem Anschein sind in diesen Flags verschleiert. Die Steganografie wird durch die Kombination eines Bildes und eines ZIP-Archivs erreicht. [Abt16, S.27]

2.5 Schnittpunkte zwischen Server und Clients

Der Server und die Clients laufen auf getrennten Systemen. Da die Studierenden Schwachstellen auf ihren Clients beheben sollen, muss das Auswertungs- und Überwachungssystem auf diese Systeme zugreifen. Dadurch lassen sich die folgenden Schnittpunkte begründen.

Der Scanner prüft vom Auswertungs- und Überwachungssystem aus, ob

- das System online
- der Webserver erreichbar
- der Bubble-Server erreichbar
- der Login zum Membersbereich erreichbar und abgesichert
- die Kundenbewertung erreichbar und abgesichert
- das SQL-Passwort geändert wurde
- der FTP-Server gegen unautorisierten Zugriff abgesichert und
- der Telnet-Dienst auf Port 23 abgeschaltet ist.

Des Weiteren verbinden sich die Clients beim Starten mit dem Auswertungs- und Überwachungssystem, um Flags für die Flagshop-Registrierung und -Anmeldung zur Verfügung zu stellen.

2.6 Abgeleitete Anforderungen

Das Auswertungs- und Überwachungssystem muss anhand der vorhergehenden Analyse folgenden Anforderungen genügen:

- Überwachung von mindestens neun Studierendensystemen
- Ermittlung und Sicherung der Zustände von Diensten, die auf den Studierendensystemen angeboten werden müssen

- Entgegennahme und Prüfung von Flags, inkl. der Verrechnung von (Straf-)Punkten
- Ermittlung und Visualisierung der Teilergebnisse sowie des Gesamtergebnisses
- Informationsvermittlung aller Dienst- und Punkteänderungen durch unter anderem Dienststatusänderung, Flagabgabe und Strafen (fortlaufende Publikation für Studierende und betreuende Personen)
- Dokumentation aller Events durch Protokollierung der einzelnen Aktionen des Systems
- Bereitstellung von Challenges, damit Studierende sich weitere Punkte erarbeiten können
- Bereitstellung eines (Flag-) Shops, bei dem mehrere Sicherheitslücken ausgenutzt werden können, um Flags zu erhalten
- Einstellungen des Spiels sollen durch betreuende Personen geändert werden können
- Verwaltung von Benutzern
- Zugangskontrolle für teilnehmende Studierende durch Prüfung der Hackit-Zugänge
- Sicherung alter Spielstände.

3 Entwurf

Dieses Kapitel beinhaltet die Architektur des Systems sowie den Entwurf der einzelnen Komponenten.

Es wird die Struktur und Zusammensetzung des Systems sowie der einzelnen Komponenten skizziert. Auch werden die Anforderungen und Erwartungen an die verschiedenen Systemkomponenten dargestellt.

Bei einigen Komponenten werden auch Alternativen aufgezeigt, welche auf Grundlage der genannten Entscheidungen im Entwurf nicht verwendet worden sind.

3.1 Entwurfsziele

Bei dem Entwurf des neuen Systems sind neben den in der Analyse beschriebenen Anforderungen auch folgende Ziele beachtet worden:

Beibehaltung der Features

Die bereits implementierten Features Flagshop und Challenges sollen auch im neuen System verfügbar sein. Zusätzlich sollen die Studierenden angeregt werden, diese auch aktiv zu nutzen.

Lose Kopplung

Zwischen dem Scanner und dem Webserver soll eine lose Kopplung herrschen, damit die Entwicklung der beiden Komponenten unabhängig voneinander fortgesetzt werden kann.

Datenhaltung in Datenbank

Die Nutzung einer Datenbank sollte aufgrund zweier Überlegungen angestrebt werden: Erstens wären damit alle Daten an einem Ort gebündelt, zweitens könnte die Berechnung von Punkten an die Datenbank abgeben werden. Datenbanken sind unter anderem für solche Aufgaben geeignet.

Modernisierung der GUI

Das Graphical User Interface soll modernisiert werden, sodass es heutigen Standards entspricht. Auch sollen hierdurch die Verständlichkeit verbessert und die Challenges sowie der Flagshop besser platziert werden.

Einheitliche Programmiersprache

Eine einheitliche Programmiersprache sollte, sofern dies möglich ist, genutzt werden. Das Betreiben der Komponenten wird erleichtert, da nicht zwei verschiedene Programmiersprachen und/oder Umgebungen installiert werden müssen. Auch erleichtert es die Programmierung, wenn nur eine entwickelnde Person parallel an den Komponenten arbeitet. Die Gefahr von falschen Syntaxen und verschiedenen Konventionen kann dadurch reduziert werden. Sollte die Anwendung durch mehrere entwickelnde Personen implementiert, gewartet sowie auf verschiedenen Systemen betrieben werden, ist dieses Ziel nichtig.

Module sparsam nutzen

Bei der Implementierung der Software sollte, soweit dies notwendig und sinnvoll ist, auf bereits vorhandene Module und Frameworks zurückgegriffen werden. Durch die Minimierung von Abhängigkeiten ist die Wartung und Verwendung der Software durch Dritte leichter möglich. Auch wird die Gefahr von Fehler auslösenden Updates minimiert. Bei der Nutzung von Modulen und Frameworks sollte auf deren Verbreitung und Wartung geachtet werden, um eine Nutzung von potenziell inaktiven Modulen/Frameworks mit eventuell vorhandenen Schwächen oder Sicherheitslücken zu reduzieren.

Containerisierung

Die Anwendung soll mit möglichst kleinem Wartungsaufwand überall nutzbar sein. Um dies zu gewährleisten, sollte eine Containerisierung verwendet werden. Bei der Nutzung ist darauf zu achten, ob und mit welchen Einschränkungen diese einsetzbar ist.

Ressourcen schonen

Um die Ressourcen des Servers zu schonen, sollten die nicht benötigten Komponenten abgeschaltet werden. Hierbei ist der Scanner hervorzuheben, der nur während des Praktikums laufen muss.

3.2 Übersicht

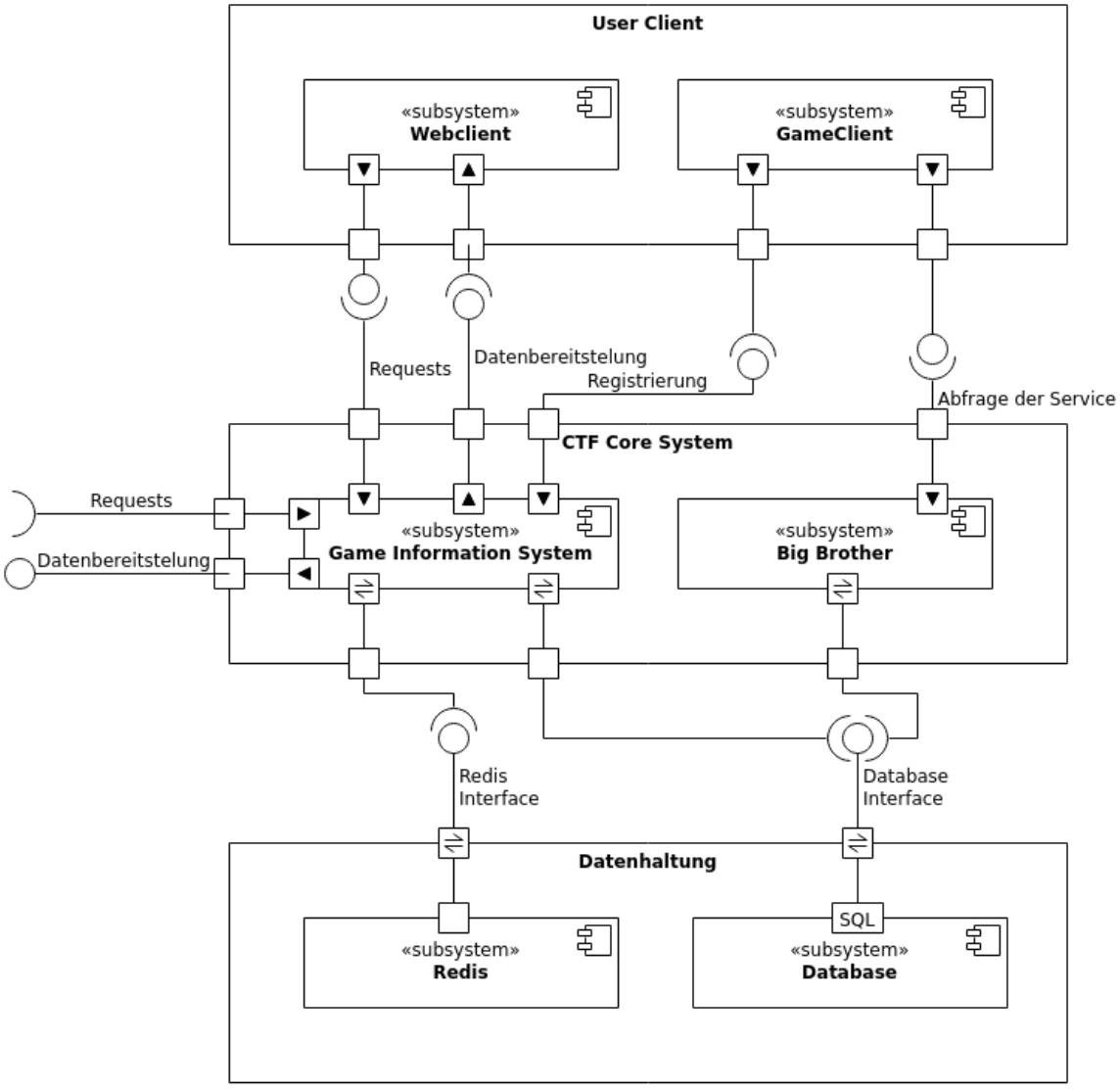

Abbildung 3.1: Übersicht über die Anwendung (Komponentendiagramm)

Die Gesamtheit des Systems (inklusive der User Clients), wie in Abbildung 3.1 zu sehen, lässt sich in drei Ebenen einteilen.

User Client Die erste Ebene *User Client* beinhaltet die auf einem User Client laufenden und für den Versuch relevanten Komponenten.

Bei der Komponente Webclient handelt es sich um einen geläufigen Webbrowser (häufig wird Firefox genutzt), der Informationen vom *CTF Core System* abruft und Daten an dieses übermittelt. Zu den Informationen gehören beispielsweise die Spielinformationen, der Spielstand

aber auch teilnehmende Gruppen und Einstellungen. Der Webclient übermittelt Daten wie gefundene Flags, Lösungen von Aufgaben und Änderungen an Einstellungen.

Die Komponente *GameClient* beinhaltet alle auf dem System für den Versuch installierte Anwendungen. Dazu zählen nicht nur die Dienste mit den Schwachstellen. Auch die clientseitige Spielverwaltungs-Software wird unter dieser Komponente zusammengefasst. Die Spielverwaltungs-Software sorgt für die richtige Konfiguration des User Clients und verteilt bzw. versteckt die Flags.

CTF Core System Die Ebene des *CTF Core System* besteht aus den Komponenten *Big Brother* und *Game Information System*.

Die Komponente *Big Brother* implementiert einen Scanner, der für die Überwachung des *GameClients* mit seinen Schwachstellen zuständig ist. Die Ergebnisse werden in der Komponente *Database* für die Bewertung und Dokumentation festgehalten.

Die zweite Komponente *Game Information System* stellt eine Schnittstelle zwischen der Datenbank und den Nutzern dar. Mithilfe dieser können Informationen unter Berücksichtigung von Berechtigungen auf einem einheitlichen Weg aus der Datenbank ausgelesen, verändert und hinzugefügt werden.

Datenhaltung Die Ebene *Datenhaltung* beinhaltet die Komponenten *Redis* und *Database*.

Die Komponente *Redis* ist nach der gleichnamigen Software Redis benannt. Bei Redis handelt es sich auch um eine Datenbank und könnte daher mit in die Komponente *Database* aufgenommen werden; die Trennung erfolgt aber auf der Grundlage der Verwendung innerhalb der Software-Architektur. Die Redis-Datenbank wird als Cache verwendet und persistiert die Daten nicht auf der Festplatte, sondern im Hauptspeicher.

Anders als bei Redis werden in der Komponente *Database* die Daten auf eine Festplatte geschrieben, um diese persistent nutzen zu können.

3.3 Containerisierung

Bei der Containerisierung wird jeder Container als eine eigene logische Maschine betrachtet. Es können mehrere Container gleichzeitig auf einer physischen Maschine betrieben werden. Die verschiedenen Container laufen unabhängig voneinander und wissen nichts von der Existenz weiterer Container. Sollte eine Kommunikation zwischen zwei Containern benötigt werden, erfolgt diese über die Netzwerkschnittstelle ähnlich der Kommunikation zwischen Anwendungen, die auf verschiedenen physischen Geräten verwendet werden. [Boe19] Damit die Container kommunizieren können, muss vorher ein Netzwerk konfiguriert werden, in dem sich die Container gemeinsam befinden.

Im Gegensatz zur Virtualisierung teilen sich die Container das zugrunde liegende Betriebssystem nicht. Dieses reduziert den Verbrauch von Ressourcen wie CPU und Arbeitsspeicher, da diese nur für die Anwendung und nicht das Betriebssystem benötigt werden.

Containerisierung wird verwendet, um Anwendungen getrennt betreiben zu können. So können verschiedene Versionen einer Software gleichzeitig auf der gleichen physischen Maschine betrieben werden, ohne dass es zu Interferenzen kommt oder das veränderte Konfigurationen notwendig sind. Darüber hinaus läuft ein Container auf die gleiche Art und Weise unabhängig von der darunter liegenden Hardwarearchitektur. So arbeitet ein Container auf dem Rechner des Entwickelnden genauso wie ein Container in einem Rechenzentrum. Durch diese Eigenschaft sind die Anwendungen in Containeren portabel. [Boe19]

Durch die Isolierungen können Anwendungen keinen negativen Einfluss auf weitere Container nehmen.

Docker wurde erstmalig im Jahr 2013 als Open-Source-Software veröffentlicht und zählt derzeitig zu der am weitesten verbreiteten Anwendung für Containierisierung. Es ist de facto der Standard für Containierisierung. [Boe19]

Heutzutage nutzen immer mehr Unternehmen Orchestrierungstechnologien wie Kubernetes oder OpenStack, um Container zentral über mehrere physische Maschinen zu verwalten und zu skalieren. [Gav18]

Eine Kubernetes-Installation ist im EZS-Labor bisher noch nicht vorhanden. Auch ist eine Installation für diese Bachelorarbeit aufgrund der Kosten-Nutzen-Differenz nicht empfehlenswert.

Anstatt einer Orchestrierungssoftware wird *Docker Compose* genutzt. Mithilfe dieses Tools können mehrere Container zu einem Service zusammengefasst und verwaltet werden. *Docker Compose* wird verwendet, um die beiden serverseitigen Anwendungen sowie die beiden Datenbanken zu einem Service zusammenzufassen. Auch sorgt es dafür, dass die Container in einem eigenen virtuellen Netzwerk laufen und nur untereinander kommunizieren können. [Doc20]

3.4 Scanner

Im Rahmen des Versuches sollen die Studierenden ihre Systeme auf Schwachstellen untersuchen. Diese ausfindig gemachten Schwachstellen sollen im Anschluss beseitigt werden. Eine Überwachung wird benötigt, um zu prüfen, ob die Studierenden diese Schwachstellen behoben haben. Das Abschalten, beziehungsweise die Verhinderung der Verwendung eines Dienstes, stellt in den meisten Fällen keine Behebung der Schwachstelle dar und wird deshalb ebenfalls geprüft. Um diese Überprüfung zu ermöglichen, wird ein Scanner benötigt, der die Dienste auf allen teilnehmenden *GameClients* abfragt und auswertet. Die bei der Überprüfung gesammelten Daten werden benötigt, um die Servicepunkte der Gruppen zu berechnen.

3.4.1 Verteilte Scanner

Eine Idee für die Lastverteilung des Scanners ist ein verteilter Scanner. Hierbei wird ein Worker-Scanner auf jedem GameClient implementiert, der das eigene System überwacht. Ein Main-Scanner sammelt die von den Worker Scannern erstellten Ergebnisse ein und speichert diese in einer Datenbank ab.

Der Vorteil des verteilten Scanners ist die Skalierbarkeit, da der Scanner auf dem Server nur von weiteren Scannern die Ergebnisse abholen, jedoch keine Überwachung durchführen muss.

Der verteilte Scanner bringt jedoch auch einige Nachteile mit sich. So muss sichergestellt werden, dass der Scanner auf dem User Client nicht manipuliert worden ist und die Ergebnisse valide sind. Eine solche Überprüfung könnte mithilfe eines „Fingerabdrucks“ geschehen. Jedoch muss dann auch geprüft werden, ob das Programm zur Erstellung des Fingerabdrucks manipuliert worden ist. Außerdem muss gewährleistet werden, dass der auf dem Client laufende Scanner dieselben Antworten und Ergebnisse erhält, wie ein fremder Nutzer. Dies ist notwendig, da die überwachten Services weiterhin von anderen Mitspielenden verwendet werden sollen.

Auf Grundlage der Nachteile, des Aufwandes der Implementierung und der ausreichenden Leistung des zentralen Scanners für den aktuellen Anwendungsfall wird ein verteilter Scanner nicht in Betracht gezogen. Die Idee des zentralen Scanners wird weiterverfolgt.

3.4.2 Zentraler Scanner

Die Herangehensweise des zentralen Scanners vermeidet die vorher beschriebenen Nachteile, da dem Ergebnis des Scanners vertraut werden kann und der Scanner zwangsläufig eine externe Sicht auf das System einnimmt. Das Ergebnis wird auf einem System ohne Einfluss der Mitspielenden berechnet.

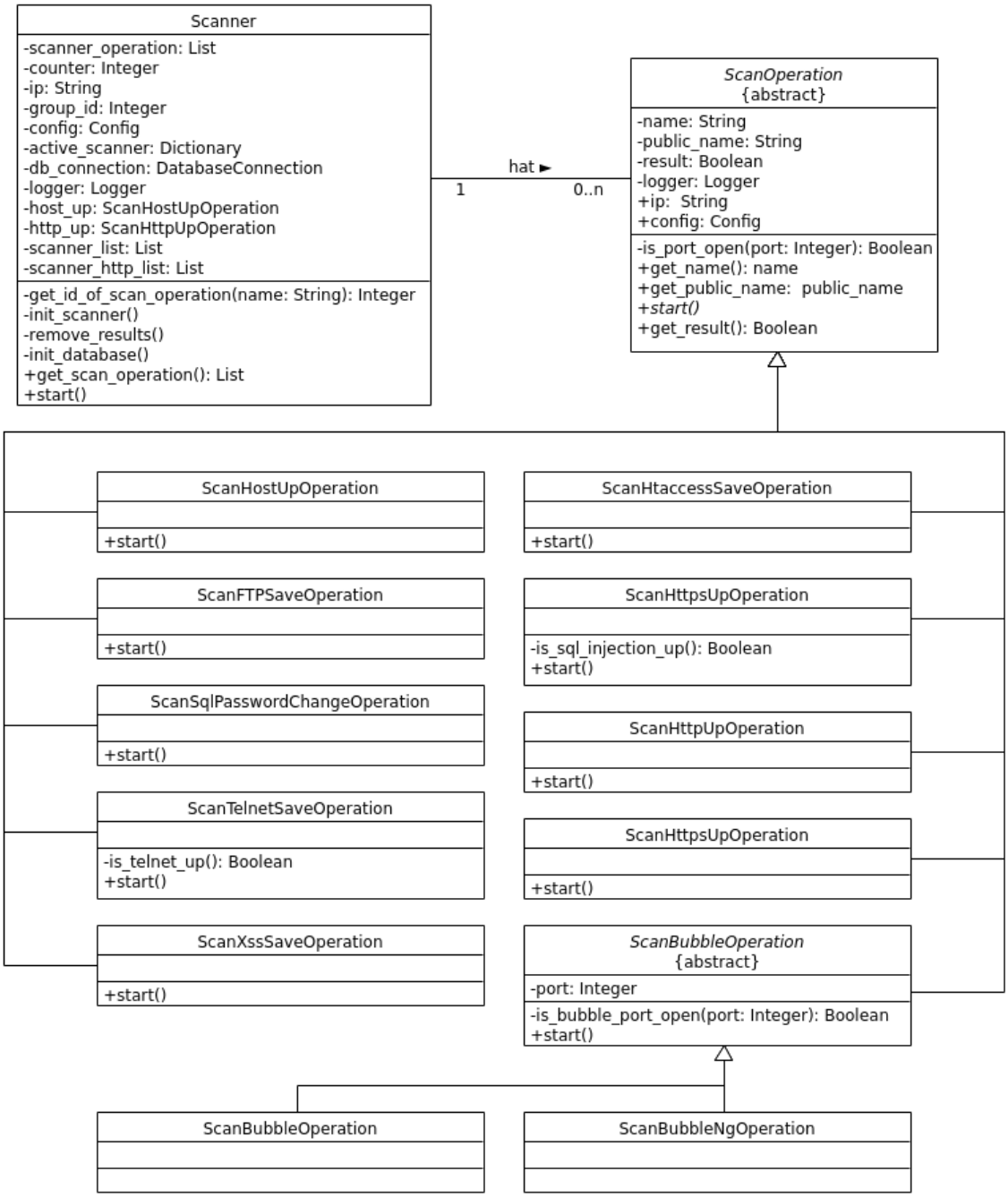

Abbildung 3.2: Klassen der Big Brother Komponente (Klassendiagramm)

Wie in Bild 3.2 erkennbar besteht die Komponente Big Brother aus der Klasse Scanner, der abstrakten Klasse ScanOperation sowie den abgeleiteten Scan-Operationen.

Die Klasse *ScanOperation* definiert die abstrakte Methode *start()*. Diese wird von den abgeleiteten Klassen implementiert und ermöglicht das Starten der einzelnen Scan-Operationen.

Ebenfalls speichern alle Scan-Operationen ihr Ergebnis im privaten Attribute *result* ab. Mithilfe der von der abstrakten Klasse implementierten Methode *get_results()* kann der Scanner das Ergebnis der Scan-Operation auslesen. Jedes Objekt der Klasse Scanner startet 0 bis n Scan-Operationen, abhängig von der Konfiguration beziehungsweise den aktiven Diensten. Außerdem stellt die abstrakte Klasse die Methode *is_port_open()* bereit, die von den Scan-Operationen genutzt werden kann, um den Status eines Ports auf dem fremden System zu prüfen. Ein Objekt der Klasse Scanner kann pro Typ maximal eine Scan-Operation starten und beinhaltet bzw. verwaltet alle Scan-Operationen für einen GameClient.

Pro GameClient wird ein Objekt der Klasse Scanner für die Überwachung benötigt.

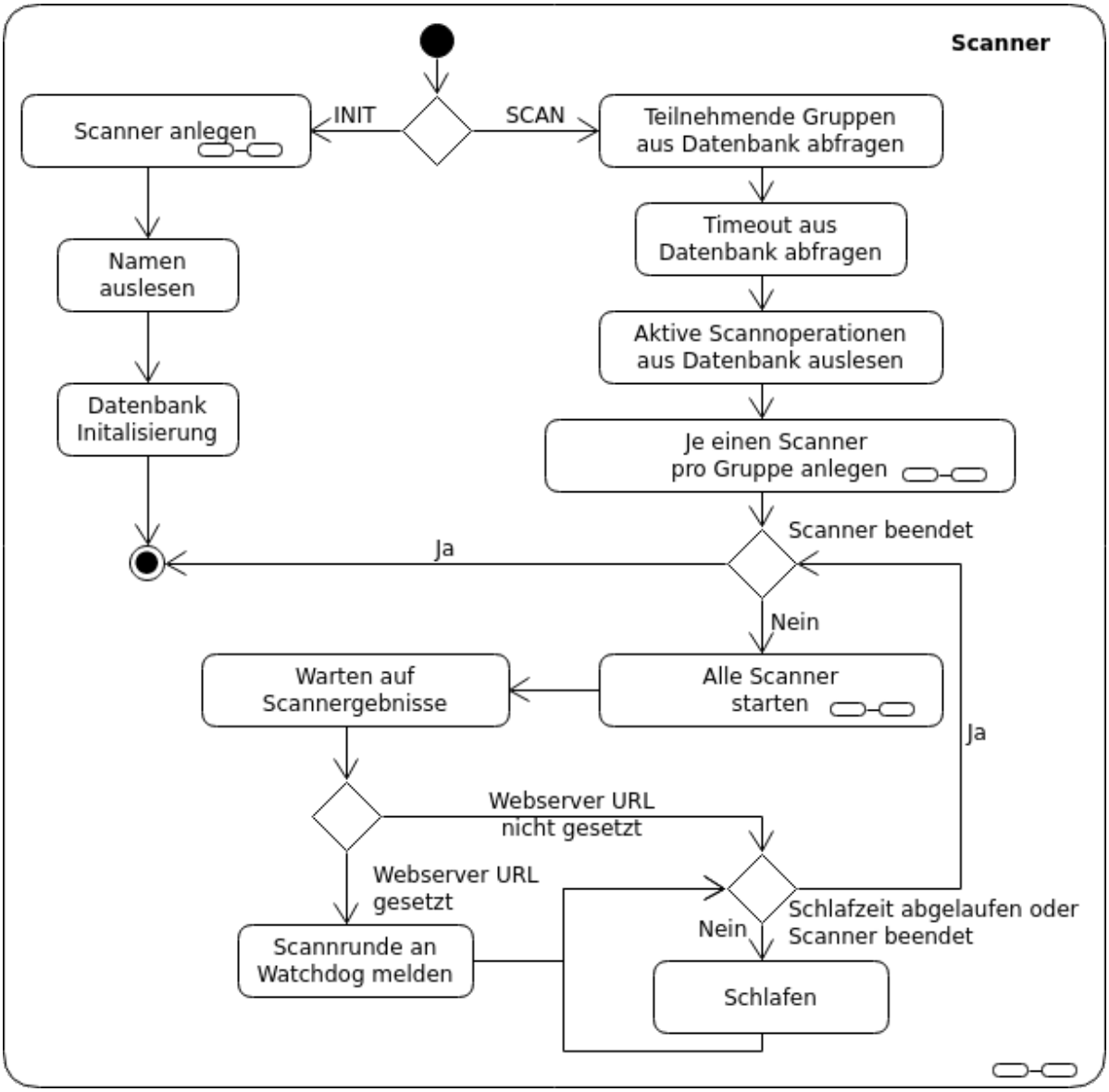

Abbildung 3.3: Ansicht des Scanners (Zustandsdiagramm)

Beim Starten des Scanners wird die zu bearbeitende Aufgabe spezifiziert.

Bekommt der Scanner die Aufgabe „INIT“, soll dieser die Datenbank Service mit den implementierten Scan-Operationen füllen, sodass Administrierende diese an- oder ausschalten können. Um die Datenbank Service zu füllen, wird zunächst ein Dummy der Klasse Scanner angelegt. Aus diesem Objekt werden von allen Scan-Operationen der Anzeigename und der interne Name ausgelesen. Nach dem Auslesen aller Operationen werden die erhaltenen Daten gebündelt in die Datenbank geschrieben. Danach beendet der Scanner die Verbindung zur Datenbank und endet erfolgreich mit dem Statuscode 0.

Falls die Aufgabe des Scanners „SCAN“ ist, wird der Scan der GameClients gestartet. Hierzu werden die teilnehmenden Gruppen und das Intervall der Scandurchgänge aus der Datenbank ausgelesen. Neben diesen Informationen werden die aktiven Scanner aus der Datenbank abgefragt. Sind alle Informationen vorhanden, wird für jede Gruppe ein Objekt der Klasse Scanner erstellt. Bei der Erstellung werden die aktiven Scan-Operationen sowie die zu überwachende Gruppe übergeben. Das Objekt legt dann für die benötigten Scan-Operationen die jeweiligen Objekte an.

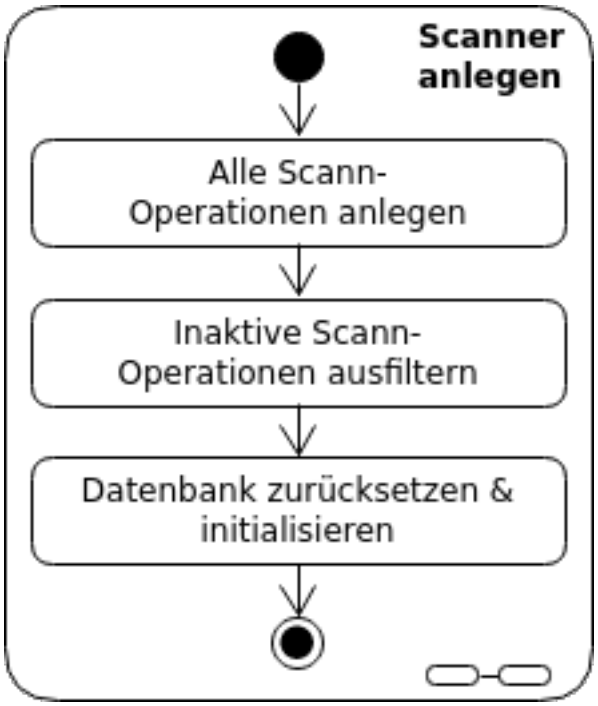

Abbildung 3.4: Erstellung eines Scanners (Zustandsdiagramm)

Nach dem Anlegen aller Scanner wird die Methode *start()* parallel gestartet. Im Anschluss daran wird auf das Beenden der gestarteten Methoden gewartet. Sollten alle abgeschlossen sein, wird geprüft, ob das Durchführen einer Scan-Runde an den Webserver gemeldet werden soll. Ist dies der Fall, wird der Server darüber in Kenntnis gesetzt, dass neue Daten in der Datenbank vorhanden sind. Danach schläft der Scanner bis zum nächsten Durchlauf.

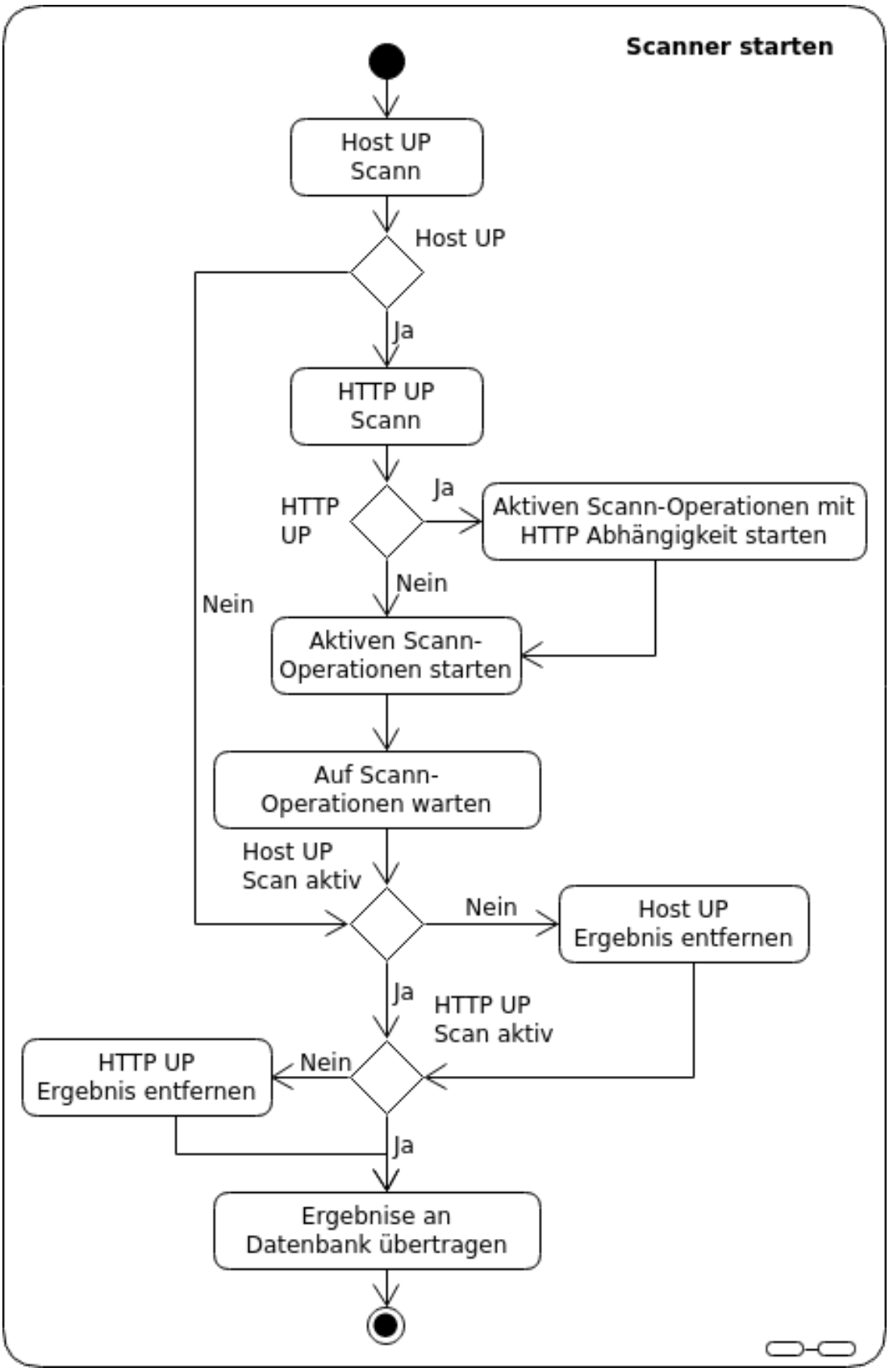

Abbildung 3.5: Starten eines Scanners (Zustandsdiagramm)

Beim Ausführen der Methode *start()* des Scanners wird zunächst geprüft, ob das entfernte System erreichbar ist. Sollte dies nicht der Fall sein, werden alle nachfolgenden Scan-Operationen nicht durchgeführt, da diese fehlschlagen würden. Anschließend wird getestet, ob der HTTP-Dienst des entfernten Systems erreichbar ist, da dieser für einige weitere Tests benötigt wird. Ist der HTTP-Dienst erreichbar, werden die Scan-Operationen, die auf dem HTTP-Dienst basieren, mit in die Liste der abzuarbeitenden Scan-Operationen aufgenommen. Danach werden alle verbleibenden Scan-Operationen parallel gestartet. Nachdem die Scan-Operationen ihre Aufgabe abgeschlossen haben, sammelt der Scanner alle Ergebnisse ein. Falls die *Host UP Scan-Operation* oder die *HTTP UP Scan-Operation* deaktiviert sind, werden diese aus dem Ergebnis entfernt. Danach übermittelt der Scanner die gesammelten Ergebnisse an die Datenbank und beendet seine Scan-Runde.

3.4.3 Scan-Operationen

Die Scan-Operationen werden parallel abgearbeitet, um so die Dauer eines kompletten Scans zu minimieren. Eine Scan-Operation prüft genau einen Dienst beziehungsweise eine Schwachstelle auf dem entfernten Rechner. Die im alten System implementierten Scans werden in die Scan-Operationen überführt. Deshalb sollen die folgenden Scan-Operationen implementiert werden.

- Host-Up
  prüft, ob der entfernte Rechner mithilfe von ICMP Paketen erreichbar ist
- Bubble-Up
  prüft, ob der Bubble-Server erreichbar ist und ob die Telnet Steuerung funktioniert
- BubbleNg-Up
  prüft, ob der Bubble-NG-Server erreichbar ist und ob die Telnet Steuerung funktioniert
- FTP-Save
  prüft, ob der FTP-Server erreichbar und ob die Nutzung des anonymen Logins unterbunden worden ist
- Htaccess-Save
  prüft, ob die Kombination aus Nutzername und Passwort des Htaccess-Schutzes geändert wurde
- SQL-Injection-Save
  prüft, ob die SQL-Injection im Login zum Membersbereich verhindert wurde
- SQL-Password-Save
  prüft, ob das lokale Passwort des SQL-Nutzers root geändert wurde
- Telnet-Save
  prüft, ob der Telnet Server deaktiviert bzw. deinstalliert wurde

- HTTP-UP
  prüft, ob der HTTP Dienst des entfernten Rechners nutzbar ist
- HTTPS-UP
  prüft, ob der HTTPS Dienst des entfernten Rechners nutzbar ist
- XSS-Save
  prüft, ob der Cross-Site-Scripting Angriff im Bewertungsformular behoben wurde.

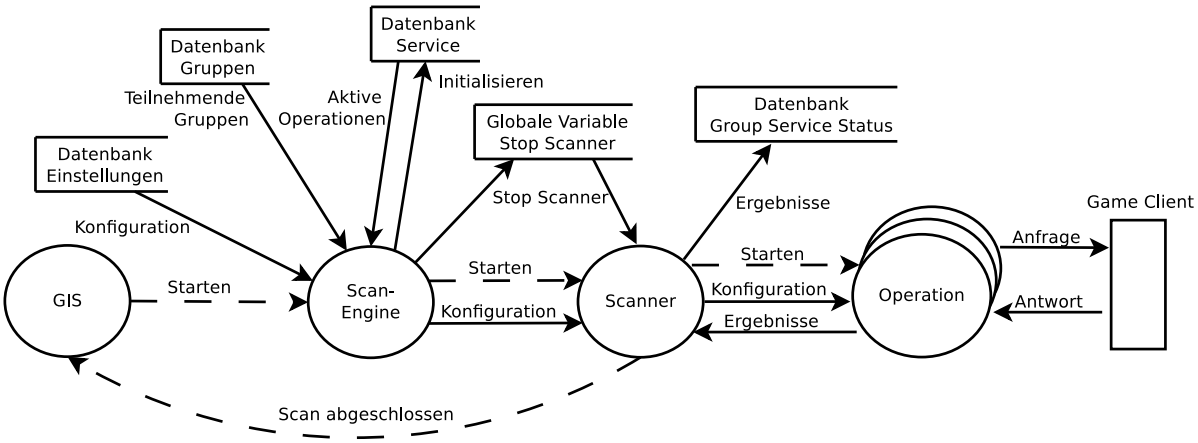

Abbildung 3.6: Datenfluss in der Scanner Komponente (Datenflussdiagramm)

Die im Datenflussdiagramm (Abbildung 3.6) sichtbaren, aber bisher nicht beschriebenen Datenflüsse finden zwischen dem Webserver und dem Scanner oder einer Scan-Operation und dem GameClient statt. Administrierende und betreuende Personen können über den Webserver den Scanner an- und abschalten. Die Scan-Operationen fragen bei dem GameClient ihren überwachten Dienst beziehungsweise ihre überwachte Schwachstelle an und erhalten eine Antwort zurück. Anhand dieser wird das Ergebnis der Scan-Operationen bestimmt.

3.5 Webserver

Der Webserver bietet die Möglichkeit der Verwaltung des Spiels, der Abgabe von Flags sowie der Durchführung von Käufen, im Flagshop, und Challenges. Auch können Informationen zum Spiel wie Einstellungen, Spielstand, Strafen und Teilnehmer abgerufen werden.

3.5.1 Verwendung mehrerer Microservices

Der Webserver kann nach dem Architekturmuster Microservices implementiert werden. Hierbei besteht die Anwendung aus mehreren unabhängigen Komponenten, die, sofern dies notwendig ist, untereinander kommunizieren. Die Komponenten sind so klein wie möglich und können jederzeit durch eine andere Implementierung ausgetauscht werden. [Wol15]

So wären die verschiedenen Komponenten des Webservers (Flagabgabe, Nutzerverwaltung, Spielsteuerung, etc.) unabhängig voneinander und können leichter weiterentwickelt oder ersetzt werden.

Die Verwendung dieses Architekturmusters wird bei der aktuellen Aufgabenstellung nicht angewendet, da durch die Entwicklung der vielen Einzelkomponenten ein zu hoher Aufwand zu mit geringem Nutzen entstehen würde.

3.5.2 Fat Webserver

Der Webserver beinhaltet sowohl die Logik als auch die Darstellung. Bei einer Anfrage an den Webserver wird eine Antwort bestimmt und diese dann in eine Vorlage beziehungsweise ein HTML Dokument eingebettet. Danach wird die Antwort zurück an den Anfragenden gesendet.

Während der Implementierung des Prototyps und des weiteren Entwurfs hat sich ein Problem mit dem Flagshop Login herausgestellt.

Für die Nutzung des Flagshops ist ein Multi-Login notwendig, da nur am Webserver eingeloggte Nutzende sich mit einem extra angelegten Account am Flagshop anmelden und diesen verwenden dürfen. Dieser Multi-Login ließ sich mit dem im Prototypen implementierten Session basierten Login nicht einfach umsetzen.

So ist für diesen Anwendungsfall ein Stateless-Login besser geeignet, da hier die benötigten Informationen vom Client, je nach Anliegen, gesendet werden können.

Die Nutzung des Stateless-Logins bietet die Perspektive der Verwendung eines Stateless-Webservers sowie eines Thin-Webservers. Bei einem Thin-Webserver werden nur Daten und keine Repräsentation an den Client zurückgesendet. So wird die Aufgabe der Darstellung der Daten an den Client übertragen.

Durch die Nutzung eines Thin-Servers werden zwei Dinge ermöglicht:

Zum einen ist es möglich, den Client und Server unabhängig voneinander zu entwickeln, zu verändern und zu verbessern. Damit kann in Zukunft eine Iteration der Software einfacher geschehen. Zum anderen können die Studierenden eigene Clients programmieren, um mit der Anwendung zu interagieren.

3.5.3 Thin Webserver

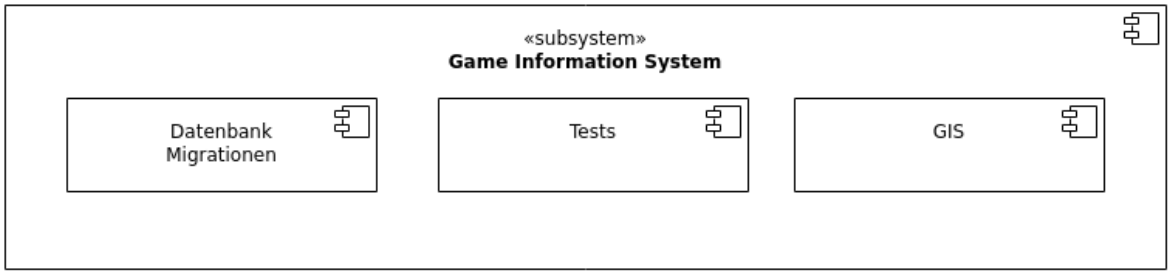

Abbildung 3.7: REST Interface im Überblick (Komponentendiagramm)

Der Server besteht aus der Komponente *Game Information System*, die wiederum aus drei weiteren Komponenten besteht.

Die Komponente *GIS* implementiert die gesamte Server Logik. In diesem Modul wird der eigentliche Thin Server, über den die teilnehmenden und betreuenden Personen mit der Anwendung interagieren können, implementiert.

In der Komponente Datenbank Migrationen sind Migrationsskripts hinterlegt, die die Datenbank Iterationen festhalten. Diese Skripts sollen genutzt werden, um die verwendete Datenbank einfach mit dem benötigten Schema zu initialisieren.

Die letzte Komponente beinhaltet Unit-Tests. Mithilfe der implementierten Unit-Tests kann die Funktionalität der Anwendung bei späteren Änderungen überprüft werden.

Da der Thin Server nur eine Brücke zwischen Nutzenden und Scanner oder Datenbank darstellt, kann von einer API (Application Programming Interface) gesprochen werden. Mithilfe dieser API können beispielsweise der aktuelle Spielstand aus der Datenbank ausgelesen und Strafen eingetragen werden.

API

Um bei der Implementierung der API einem Standard zu folgen, wird der de facto Standard für HTTP-APIs Representational State Transfer (REST) verwendet.

Ein RESTful-Interface ermöglicht, dass Ressourcen auf dem Webserver eindeutig identifizierbar sind, damit diese als Ziel von Operationen ausgewählt werden können. Darüber hinaus werden einheitliche Schnittstellen genutzt. Dazu müssen Standardmethoden und -repräsentationen verwendet werden. [Bei14]

Durch diese Anforderungen bietet das REST-Interface die Möglichkeiten, alle zur Verfügung stehenden Ressourcen auf die gleiche Art und Weise zu verwalten. Außerdem wird mit dieser Herangehensweise die HTTP-Methode *GET* nicht länger zur Veränderung oder Erschaffung

von Ressourcen verwendet, sondern die dafür ausgelegten HTTP-Methoden. Die für die Verwendung benötigten Daten werden auch nicht länger als Parameter der Anfrage beigefügt, sondern im Body der Anfrage übertragen. [Bei14]

Alle Routen werden mit dem Versionspräfix */v1* versehen. Diese Versionierung soll bei späteren Iterationen der API Kollisionen verhindern und die Nutzung der alten Versionen weiterhin ermöglichen.

Bei der Implementierung sollen die zur Verfügung stehenden HTTP-Methoden benutzt werden. Das REST-Interface soll auch mit entsprechenden HTTP-Codes antworten, um die Antwort und den Erfolg einer Anfrage auch ohne Antworttext interpretieren zu können. Bei der Datenübertragung zwischen Client und Server soll das JavaScript Object Notation (JSON) Format für die Formatierung der gesendeten Daten verwendet werden.

GET	Auf Ressourcen zugreifen
POST	Neue Ressourcen erzeugen
PUT	Bestehende Ressourcen verändern
DELETE	Vorhandene Ressourcen löschen

Tabelle 3.1: Übersicht über die verwendeten HTTP-Methoden

Neben den in Tabelle 3.1 aufgezeigten Methoden gibt es noch weitere HTTP-Methoden, die keine Anwendung in dem zu implementierenden REST-Interface erhalten.

3.5.4 Funktionale Eigenschaften

Login

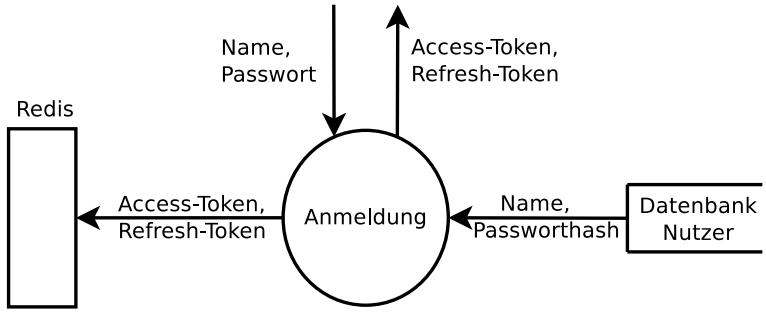

Abbildung 3.8: Datenfluss während eines Logins (Datenflussdiagramm)

Bei einem Login sendet der Nutzende die Kombination aus Nutzernamen und Passwort in seiner Anfrage an den Webserver. Sollten der Nutzername und das Passwort mit den in der

Datenbank gespeicherten Informationen übereinstimmen, wird ein Access- und ein Refresh-Token ausgestellt. Wichtig hierbei ist, dass das Passwort in der Datenbank **nur** im gehashten Format vorliegt.

Sollte kein Hackitzugang benötigt werden, muss für die Ausstellung der Tokens kein Nutzername und Passwort angegeben werden.

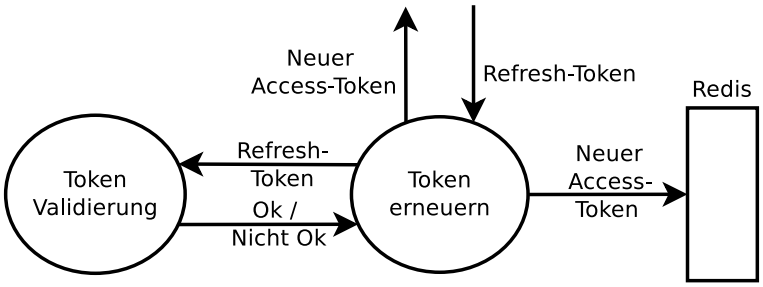

Abbildung 3.9: Datenfluss während eines Logins mit Refresh Token (Datenflussdiagramm)

Ein Refresh-Token wird ausgestellt, da beide Tokens eine begrenzte Lebenszeit haben. Der Refresh-Token hat eine längere Lebenszeit und ermöglicht die erneute Ausstellung eines Access-Tokens ohne Angabe eines Nutzernamens und Passwortes.

Des Weiteren werden alle ausgestellten Token für die Dauer der Gültigkeit in der Redis-Datenbank gespeichert, um diese bei späteren Anfragen zu validieren.

Authentifizierung

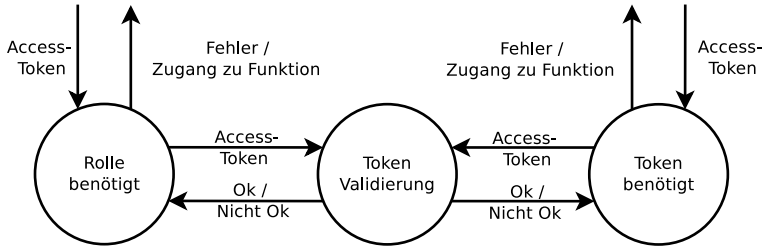

Abbildung 3.10: Datenfluss der Authentifizierung und Autorisierung (Datenflussdiagramm)

Die Authentifizierung des Benutzers wird über die in Abschnitt 3.5.4 erwähnten Tokens realisiert. Die Tokens müssen vom Client bei jeder Anfrage mitgesendet werden und enthalten neben der Nutzerkennung weitere Informationen zu Berechtigungen.

Dies reduziert die Anfragen an die Datenbank. Nachteil hierbei sind langsame Änderungen der Berechtigungen, da die neuen Berechtigungen erst mit dem Erstellen eines neuen Tokens übernommen werden. Da Änderungen an der Berechtigung selten vorkommen, kann dieser Nachteil relativiert werden.

Mit den im Token codierten Berechtigungen kann der Server die Autorisierung der angefragten Aktion durchführen.

Tokenvalidierung

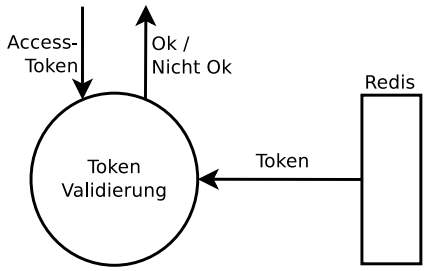

Abbildung 3.11: Datenfluss während der Tokenvalidierung (Datenflussdiagramm)

Für die Authentifizierung und Autorisierung müssen die mitgesendeten Tokens validiert werden. Dazu erhalten diese einen Zeitstempel der Ausstellung und der Verjährung sowie eine Signatur, die eine Manipulation verhindern soll. Führt der Nutzende einen Logout durch, wird der Token in der Datenbank als widerrufen markiert.

Registierung von GameClients

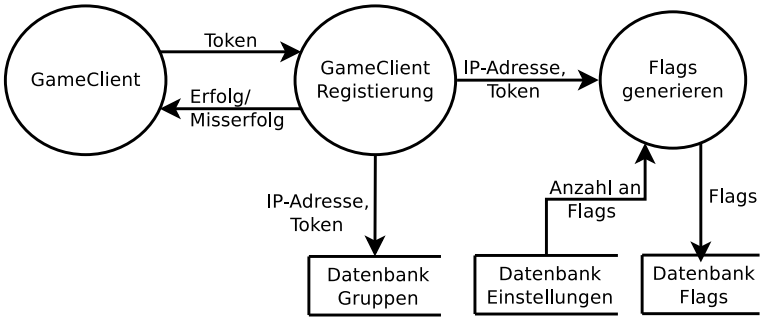

Abbildung 3.12: Datenfluss der Registrierung (Datenflussdiagramm)

Die teilnehmenden GameClients sollen aufgrund der Anforderung nicht länger fest in der Anwendung vorliegen, sondern dynamisch hinterlegt werden. Deshalb müssen die GameClients sich vor Spielstart registrieren.

Flaggenerierung Durch die benötigte Registrierung ist es möglich, dass der Algorithmus zur Flaggenerierung verändert wird. Dies soll geschehen, damit die Flags nicht länger pro Semester, sondern im besten Fall für immer einzigartig sind.

Damit die Flags einzigartig sind, überträgt der GameClient einen Token bei der Registrierung. Dieser wird sowohl vom GameClient als auch vom Server genutzt, um die benötigten Flags zu generieren.

Die Idee, dass die Flaggenerierung nur auf dem Server stattfindet, wurde verworfen, da die nutzende Person auf dem Client die Möglichkeit hat, die Flags bei der Übertragung abzufangen und abzugeben. Eine Verschlüsselung oder andere Verfahren würden keinen Mehrwert in puncto Sicherheit liefern, da die nutzende Person Root-Rechte auf ihrem System hat. Eine Verschlüsselung der Flags mit dem PGP-Verfahren bringt keinen wirklichen Mehrwert, da auf den vorhandenen PGP-Key zugegriffen werden kann.

Flagabgabe

Da die Abgabe der Flags einen Hauptteil des Versuchs darstellt, müssen authentifizierte Spielende die auf dem eigenen oder gegnerischen System gefundene Flags abgeben können.

Das GIS muss die abgegebene Flag auf Gültigkeit prüfen und entweder Offensiv- und Defensiv- oder Discoverpunkte verrechnen. Auch muss sichergestellt werden, dass eine Gruppe eine Flag nur einmal abgeben kann.

Gruppen können alle eigenen Flags sowie gegnerische Flags, die auf den GameClients verteilt sind, abgeben. Gegnerische Flagshop-Flags können nicht abgegeben werden, da diese als nicht schützenswert betrachtet werden.

Ob die Abgabe im Angriffszeitraum geschieht, muss geprüft werden. Andernfalls soll die Abgabe nicht weiter verarbeitet werden und bei einer Abgabe während der Discoverzeit soll eine Strafe ausgesprochen werden.

Spielsteuerung

Die Betreuenden sollen über das GIS das Spiel steuern können. Dabei müssen sie in der Lage sein, ein Spiel zu starten, zu pausieren und zu beenden. Die Steuerung des Scanners soll an das Spiel gekoppelt werden, aber auch eigenständig funktionieren.

Des Weiteren sollen die betreuenden Personen die Möglichkeit besitzen, Strafen an Gruppen, die die Regeln gebrochen haben, verteilen zu können. Hier müssen die Gruppe, der Grund der Bestrafung sowie die Anzahl der Strafpunkte übermittelt werden.

Accountverwaltung

Administrierende und betreuende Personen sollen die Möglichkeit erhalten, neue Accounts anzulegen, zu bearbeiten und zu löschen. Wichtig hierbei ist, dass die betreuenden Personen nur Accounts der Rolle Player verwalten kann.

Außerdem ist ein Import der Studierendenzugänge zu realisieren, damit die generierten Hackit-Zugänge einfach übernommen werden können.

Spielstände

Die abgespeicherten Spielstände sollen über das GIS abrufbar sein. Hierzu soll zuerst eine Liste der verfügbaren Spielstände, inklusive des Datums der Speicherung, an den Anfragenden übermittelt werden. Mit dieser Liste kann der benötigte Spielstand ermittelt und im nächsten Schritt angefragt werden.

Dieser detaillierte Spielstand beinhaltet die Punkte aller teilnehmenden Gruppen, die überwachten Schwachstellen und Dienste sowie den Spielstatus.

Steuerung der Services

Die betreuenden Personen des Versuchs sollen die Möglichkeit erhalten, einzelne Scan-Operationen zu (de-)aktivieren, ohne dass eine Änderung an der Software vorgenommen werden muss. Die Gewichtung der einzelnen Services soll jederzeit änderbar sein.

Einstellungen

Die betreuenden Personen sollen die Einstellungen des Spiels verändern können. Die Spieleinstellungen sollen als Key-Value-Paar vorliegen. Zu den Einstellungen zählen beispielsweise die Anzahl der generierten Flags pro Gruppe, aber auch, ob ein anonymer Login für die Rolle *player* vorgesehen ist.

Notizen

Den betreuenden Personen soll es ermöglicht werden, permanente Hinweise zu geben. Außerdem sollen sie in der Lage sein, bestehende Hinweise zu bearbeiten oder zu löschen.

Studierende können die verfassten Notizen einsehen, um Hinweise oder Anweisungen zum Versuch zu erhalten.

Flagshop

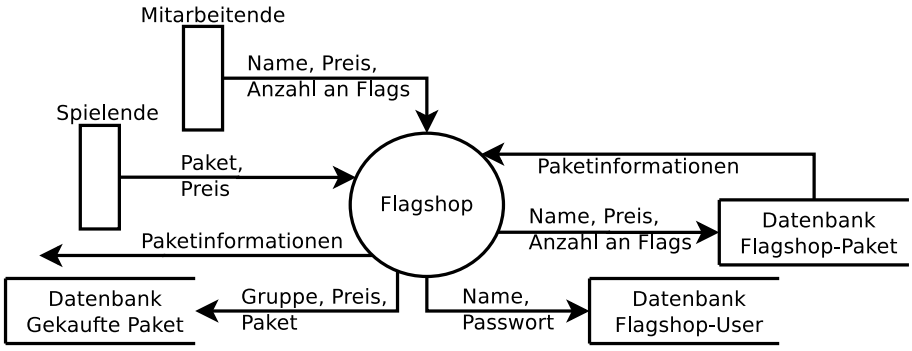

Abbildung 3.13: Datenfluss im Flagshop (Datenflussdiagramm)

Der Flagshop muss die Verwaltung von Flagshop-Usern und Flagshop-Paketen sowie einen Kauf der Flagshop-Pakete ermöglichen.

Flagshop-User können durch die teilnehmenden Teams erstellt werden. Für die Erstellung werden Flags an die Gruppe verteilt. Betreuende Personen können ebenfalls für Gruppen Flagshop-User erstellen. Es ist ihnen auch möglich, bestehende Flagshop-User zu ändern und zu löschen.

Außerdem müssen Flagshop-Pakete durch die Betreuenden verwaltet werden. Die Erstellung von Flagshop-Paketen soll nur vor einem Spielstart möglich sein, damit die richtige Anzahl an Flags bei der Registrierung eines GameClients erzeugt werden kann. Die Flagshop-Pakete enthalten neben den Kosten und dem Namen auch die Anzahl der erwerbbaren Flags.

Studierende können Flagshop-Pakete erwerben. Die Besonderheit hierbei soll sein, dass diese den Kaufpreis mitsenden und so die Flagshop-Pakete für einen anderen Preis als angegeben erwerben können.

Challenges

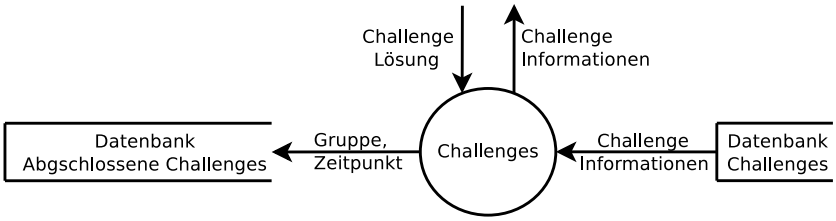

Abbildung 3.14: Datenfluss der Challenges (Datenflussdiagramm)

Die Studierenden müssen Informationen zu Challenges sowie deren Ablageort abrufen können. Darüber hinaus sollen sie in der Lage sein, die Challenges abzuschließen. Hierfür teilen

die Challenges am Ende ein Passwort mit, welches ebenfalls in der Datenbank vorhanden ist. Die Studierenden können dieses Passwort dann als Beweis der Lösung der Challenge an das GIS übermitteln.

Die Challenges werden von einem Webserver zur Verfügung gestellt. Sie werden nicht über das GIS verteilt, da dies als API nur Daten und keine Ressourcen an den Client übermittelt.

3.5.5 Reverse Proxy

Für das Betreiben wird auf den bereits im EZS-Labor verwendeten Apache HTTP-Server zurückgegriffen. Durch die Entscheidung bleibt dieser als einziger Einstiegspunkt für Anfragen zuständig. Die Protokollierung der Anfragen sowie die Verschlüsselung der Verbindung kann an die Webserver Software abgegeben werden.

Der Reverse Proxy muss bei der Weiterleitung der Anfrage an das GIS sicherstellen, dass die eigentliche IP-Adresse der Anfrage dem GIS zugänglich ist. Diese wird benötigt, um die zugehörige Gruppe bei der Registrierung und dem Login zu bestimmen. Die originale IP-Adresse der Anfrage soll in einem HTTP-Header gespeichert werden.

Dem HTTP-Header können Server und Client zusätzliche Informationen zur Anfrage oder Antwort beifügen. [cM20a]

Um einen Spoofing-Angriff zu erschweren und das Vertrauen in den Header mit der IP-Adresse zu erhöhen, soll ein weiterer Header mit einem Geheimnis gesetzt werden. Anhand dieses Geheimnisses kann das GIS entscheiden, ob die Anfrage über einen vertrauenswürdigen Proxy weitergeleitet oder ob ein Spoofing-Angriff durchgeführt worden ist. Die Erweiterung wird benötigt, da die Anwendung auch ohne einen Reverse Proxy betrieben werden kann. In diesem Anwendungsfall kann die angreifende Person sich als Proxy für eine andere Gruppe ausgeben.

Bei einem Spoofing-Angriff versendet die angreifende Person Anfragen für andere IP-Adressen und erhält die Antwort. So könnte diese im Namen anderer agieren und den Angegriffenen beispielsweise Minuspunkte verschaffen.

3.6 Datenbank

Im Gegensatz zur bestehenden Anwendung wird auf eine relationale Datenbank anstatt auf Dateien zurückgegriffen. Das Vorgehen resultiert unter anderem aus der gestiegenen Datenmenge. Eine NoSQL-Datenbank wird nicht benötigt, da sich alle Daten in Schemen untergliedern lassen und häufige Änderungen vorgesehen sind. Die Nutzung einer Datenbank stellt sicher, dass sich die Daten an einer zentralen Stelle befinden und nicht über die Anwendung verstreut sind. Auch wird eine mögliche Redundanz von Daten verhindert, da in Datenbanken

mit Relationen gearbeitet werden kann. Durch die AKID-Eigenschaft (Atomarität, Konsistenz, Isolation und Dauerhaftigkeit) der Datenbanken wird sichergestellt, dass Daten in der Datenbank vollständig und konsistent sind. Ferner sind die Abfragen an eine Datenbank besser optimiert, als Abfragen auf eine Datei. [DO17]

Es ist möglich, die Punkteberechnung in der Datenbank unter Zuhilfenahme von *Views* durchzuführen.

3.6.1 Basis-Tabellen

Abbildung 3.15: Ansicht der Basis-Tabellen (ER-Diagramm)

In der Abbildung 3.15 ist neben der alleinstehenden Tabelle ***settings*** die Tabelle ***users*** sowie die n:m-Beziehung zwischen ***users*** und ***groups*** dargestellt. Die n:m-Beziehung wird über eine Referenztabelle (***penalties***) mit einer 1:n-Beziehung zwischen ***groups*** und ***penalties*** und einer 0,1:n-Beziehung zwischen ***users*** und ***penalties*** modelliert. In der n:m-Beziehung werden die ausgesprochenen Strafen gespeichert. Jede Gruppe kann von mehreren betreuenden Personen oder dem System mehrmals verwarnt werden.

Einstellungen

Die ***settings***-Tabelle beinhaltet Einstellungen des Spiels als key-value Paar. Hierbei sind beide Spalten Strings und Key einzigartig. Zu den Einstellungen gehören beispielsweise das Scan-Intervall, die Anzahl der Flags pro Team sowie das Ende der Discover- und Attackzeit.

Account

In der Tabelle ***users*** sind die Login-Informationen der administrierenden, der betreuenden und der spielenden Personen hinterlegt. Da der Login aus Nutzername und Passwort besteht, werden die Spalten *name* und *password* benötigt. Auch muss anhand des Nutzernamens eindeutig auf einen Account geschlossen werden, deshalb ist die Spalte *name* als unique gekennzeichnet. Die Rolle des Accounts wird mithilfe eines Enums (*role*) definiert. Ein Enum ist eine Liste von benannten Werten.

Strafen

Die durch das System oder die betreuenden Personen verteilten Strafen werden in der Tabelle ***penalties*** gespeichert. Dazu wird neben dem Grund der Strafe (*reason*), dem Zeitpunkt der Bestrafung (*timestamp*), auch die Anzahl an Strafpunkten (*points*) festgehalten. Damit Gruppen von einer betreuenden Person öfters bestraft werden können, ist der Primärschlüssel des Eintrages die ID der Strafe.

3.6.2 Gruppen-Tabellen

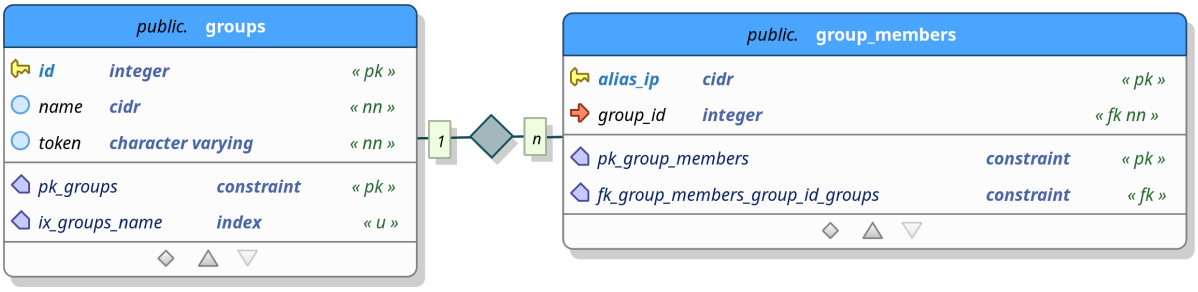

Abbildung 3.16: Ansicht der Gruppen-Tabellen (ER-Diagramm)

Wie in Abbildung 3.16 zu sehen ist, existiert zwischen der Tabelle ***groups*** und ***group_members*** eine 1:n-Beziehung. Dies bedeutet, dass eine Gruppe mehrere Mitglieder haben kann, aber jedes Mitglied nur genau in einer Gruppe vertreten sein kann.

Gruppe

In der Tabelle ***groups*** werden alle teilnehmenden GameClients vermerkt. Jede Gruppe besitzt eine eindeutige *ID* und einen eindeutigen *Namen*. Der *Name* der Gruppe repräsentiert die IP-Adresse des GameClients und ist nach der *Classless Internet Domain Routing (CIDR)* Konvention abgelegt. Des Weiteren besitzt die ***groups*** Tabelle die Spalte *token*. In dieser wird der individuelle Token, der zur Flaggenerierung benötigt wird, abgespeichert. Dies ist nötig, damit der Server dieselben Flags wie der GameClient generieren kann und die betreuenden Personen die Korrektheit verifizieren können.

Gruppenmitglieder

Die Tabelle ***group_members*** beinhaltet alle IP-Adressen, die im Namen einer Gruppe agieren dürfen. Dafür wird die IP-Adresse im Feld *alias_ip* und die vertretende Gruppe als Referenz im Feld *group_id* gespeichert. Da das Feld *alias_ip* als Primärschlüssel verwendet wird, kann jede IP-Adresse nur genau eine Gruppe vertreten.

3.6.3 Flags-Tabellen

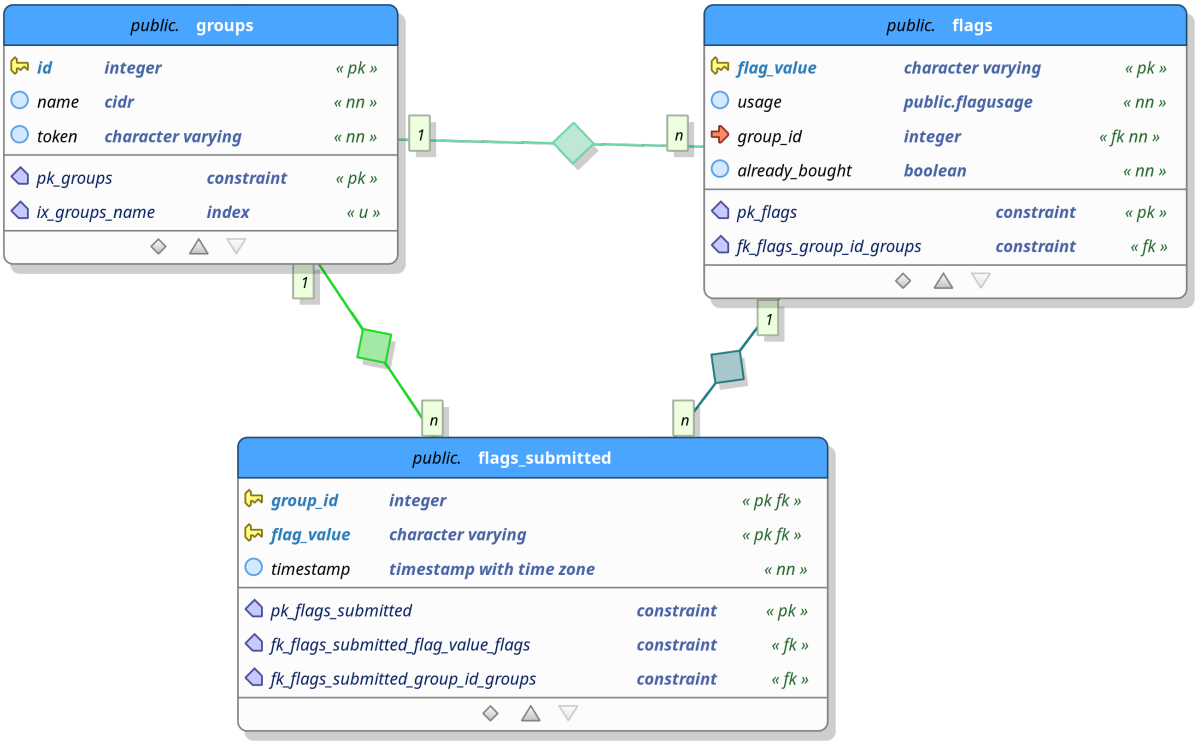

Abbildung 3.17: Ansicht der Flags-Tabellen (ER-Diagramm)

In der Abbildung 3.17 ist die 1:n-Beziehung zwischen den Tabellen ***groups*** und ***flags*** erkennbar. Dieses hat zur Folge, dass eine Gruppe mehrere Flags besitzen kann, jedoch eine Flag immer genau zu einer Gruppe gehört. In der Abbildung ist weiter die n:m-Beziehung zwischen ***groups*** und ***flags*** abgebildet. Diese wird in eine Verknüpfungstabelle namens ***flags_submitted*** mit zwei 1:n-Beziehung zerlegt. Diese Tabelle enthält die abgegebenen Flags.

Flags

Die während eines Spiels generierten und benötigten Flags werden in der Tabelle ***flags*** abgelegt. Hierzu wird der Wert einer Flag als String in der Spalte *flag_value* festgehalten. Auch wird die Nutzungsart unter Zuhilfenahme eines Enums und die Gruppenzugehörigkeit als Referenz gespeichert. Außerdem wird in der Spalte *already_bought* als Bool vermerkt, ob diese Flag bereits im Flagshop erworben wurde. Diese Spalte besitzt nur Relevanz für die dementsprechenden Flagshop Flags und wird bei den übrigen Flags ignoriert.

Abgegebene Flags

Die durch die Studierenden abgegebenen Flags werden in der Tabelle ***flags_submitted*** eingetragen. Auch wird der Zeitpunkt der Abgabe mitgespeichert, um eventuelle Betrugsversuche aufzudecken. Die Spalten *group_id* und *flag_value* bilden zusammen den Primärschlüssel. Dieses ermöglicht, dass jede Gruppe jede Flag abgeben kann. Die Mehrfachabgabe einer Flag durch eine Gruppe wird so auch unterbunden.

3.6.4 Service-Tabellen

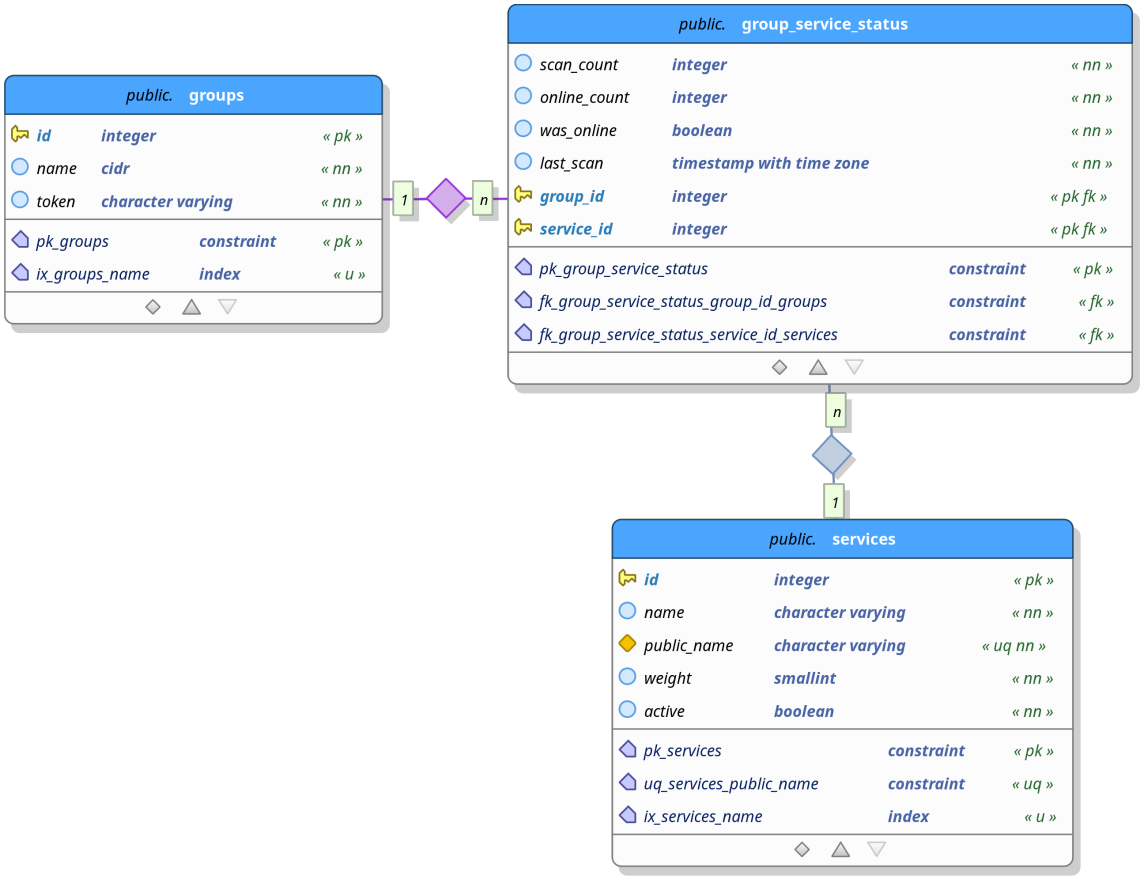

Abbildung 3.18: Ansicht der Service-Tabellen (ER-Diagramm)

In der Abbildung 3.18 ist die ***services***-Tabelle sowie die n:m-Beziehung zwischen ***groups*** und ***services*** zu sehen. Diese wird als Referenztabelle (***group_service_status***) mit zwei 1:n-Beziehungen dargestellt.

Services

In der ***services***-Tabelle sind alle vom Big Brother überwachten Dienste und Schwachstellen eingetragen. Jeder dieser Einträge besteht aus einer eindeutigen ID, dem internen Namen der Operation (*name*), dem angezeigten Namen (*public_name*) und der Gewichtung im Gesamtergebnis (*weight*). Die Überprüfung der Schwachstelle kann mithilfe des Bool-Wertes *active* an- oder abgeschaltet werden.

Servicestatus der Gruppen

In der Referenztabelle ***group_service_status*** wird das Ergebnis der Überprüfung der Schwachstellen und Dienste jeder Gruppe abgespeichert. Dazu wird für die Kombination aus Gruppe und Service ein Eintrag angelegt und die Anzahl der Scans in *scan_count* und die Anzahl der erfolgreichen Scans in *online_count* gespeichert. Des Weiteren wird der Zeitpunkt des letzten Scans in *last_scan* und der Erfolg des letzten Scans in *was_online* festgehalten. Erfolgreich bedeutet in diesem Kontext, dass die Studierenden die überwachte Schwachstelle ausgebessert und den überwachten Dienst erreichbar gehalten haben.

3.6.5 Flagshop-Tabellen

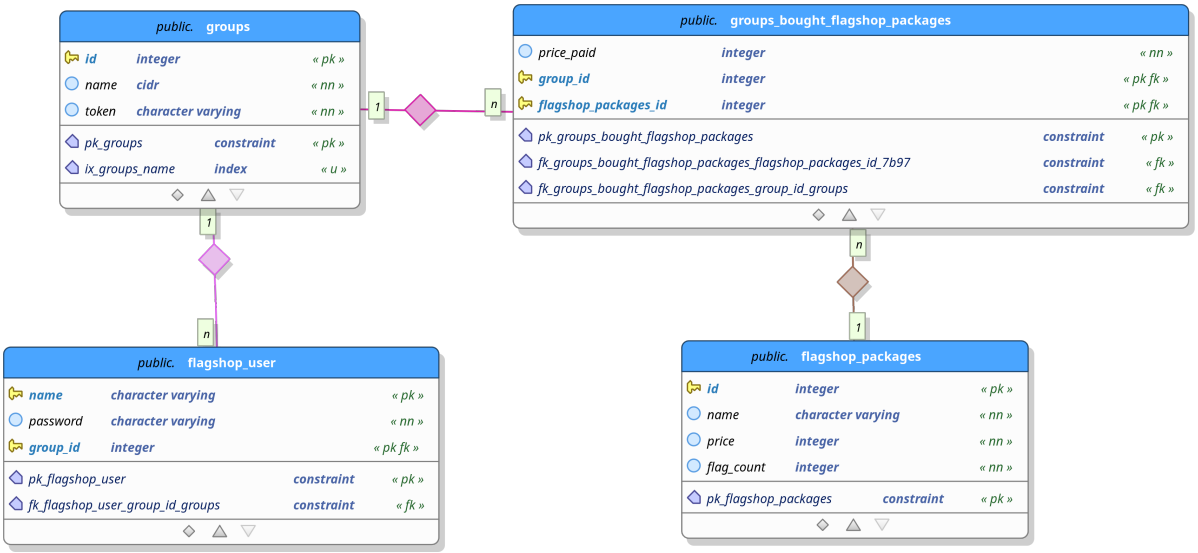

Abbildung 3.19: Ansicht der Flagshop-Tabellen (ER-Diagramm)

Die Abbildung 3.19 stellt die 1:n-Beziehung zwischen den Tabellen ***groups*** und ***flagshop_user*** dar. Das heißt, dass jeder Gruppe mehrere Flagshop-User zugehörig sein können, jeder User aber genau einer Gruppe angehört. Die n:m-Beziehung zwischen ***groups*** und ***flagshop_packages*** wird als Referenztabelle (***groups_bought_flagshop_packages***) mit zwei 1:n-Beziehung dargestellt. Die Referenztabelle beinhaltet die gekauften Flagshop-Pakete pro Team.

Flagshop-User

Für die Nutzung des Flagshops müssen die Gruppen einen oder mehrere Flagshop-User anlegen. Diese werden in der Tabelle ***flagshop_user*** abgelegt. Hierfür werden neben der erstellenden Gruppe (*group_id*) auch das Passwort (*password*) und der Name (*name*) abgespeichert.

Der Primärschlüssel wird aus der Gruppe und dem Namen gebildet. So ist der Nutzername für die Gruppe eindeutig. Mehrere Gruppen können aber Flagshop-Useraccounts mit demselben Namen anlegen.

Flagshop-Pakete

Die betreuenden und administrierenden Personen können Flagshop-Pakete anlegen, die von den Studierenden gekauft werden können. Diese werden in der Tabelle ***flagshop_packages*** abgelegt und enthalten Informationen zum Preis (*price*) und der Anzahl der erhaltenen Flags (*flag_count*). Auch muss ein Name für das Paket über das Feld *name* angegeben werden.

Gekaufte Flagshop-Pakete

Die von den Gruppen gekauften Flagshop Pakete werden in der Tabelle ***groups_bought_flagshop_packages*** abgespeichert. Auch wird im Feld *price_paid* der beim Kauf gezahlte Preis festgehalten, da die Studierenden den Preis eines Flagshop-Paketes beim Kauf reduzieren können. Jedes Paket kann genau einmal von jeder Gruppe gekauft werden, da sich der Primärschlüssel aus Gruppe (*group_id*) und Paket (*flagshop_packages_id*) zusammensetzt.

3.6.6 Challenge-Tabellen

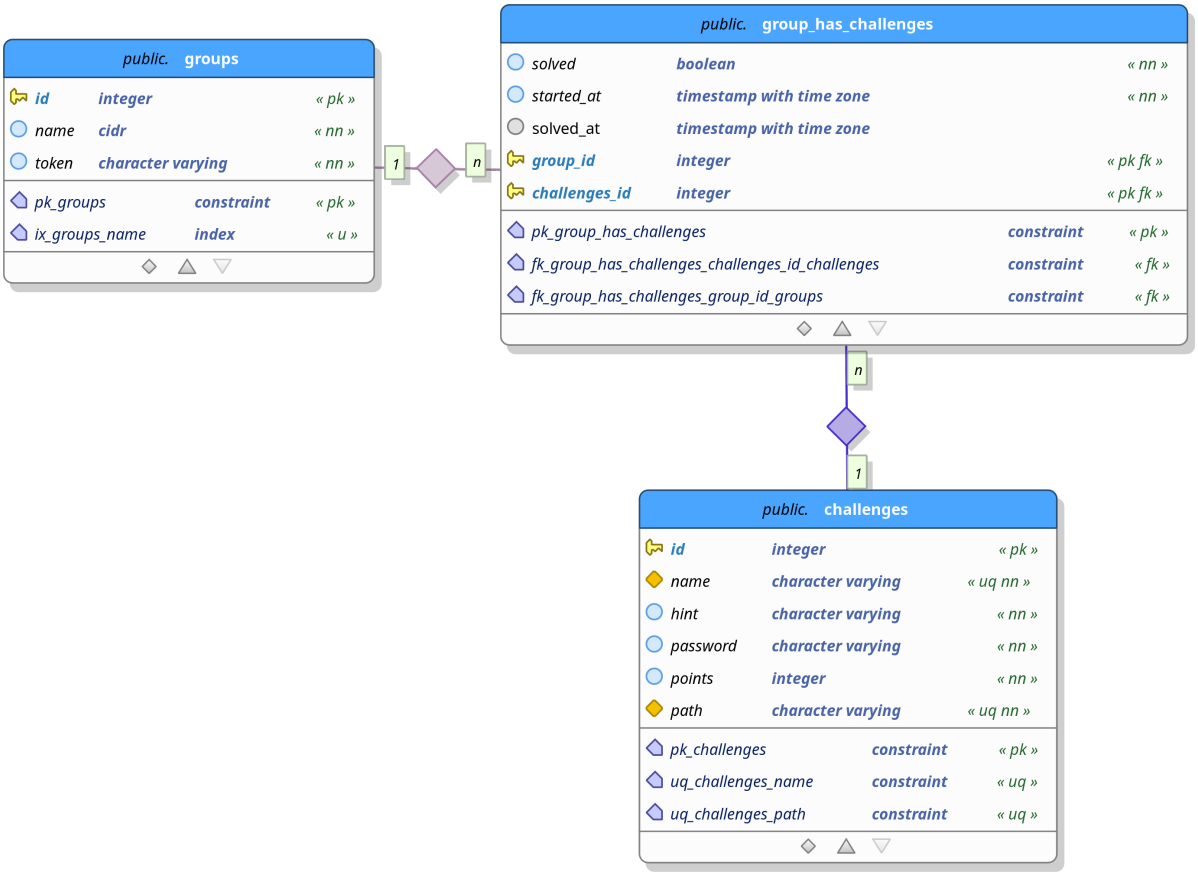

Abbildung 3.20: Ansicht der Service-Tabellen (ER-Diagramm)

In der Abbildung 3.20 ist die ***Challenge*** Tabelle und die n:m-Beziehung zwischen den Tabellen ***groups*** und ***challenges*** zu sehen. Diese Beziehung ist als Verknüpfungstabelle (***group_has_challenges***) mit zwei 1:n-Beziehung realisiert.

Challenges

Die ***challenges*** Tabelle beinhaltet alle Challenges, die von den Studierenden gelöst werden können. Jede Challenge besitzt eine eindeutige ID. Neben dieser werden für jede Challenge ein Titel in *name*, ein Hinweis in *hint*, das zum Lösen benötigte Passwort in *password*, die Anzahl der Punkte in *points* und der URL-Pfad, über den die Challenge abgerufen werden kann, in *path* gespeichert.

Gestartete / Abgeschlossene Challenges

Alle angefangenen und abgeschlossenen Challenges werden in der Tabelle ***group_has_challenges*** abgespeichert. Es werden das Startdatum (*started_at*) der Challenge, der Status des Abschlusses (*solved*) und bei erfolgreicher Absolvierung der Challenge das Enddatum (*solved_at*) gespeichert. Durch die Zusammensetzung des Primärschlüssels aus Challenge und Gruppe kann jede Gruppe jede Challenge genau einmal starten und absolvieren.

3.6.7 Weitere Tabellen

Abbildung 3.21: Ansicht der weiteren Tabellen (ER-Diagramm)

Wie in Abbildung 3.21 erkennbar, besitzen die Tabellen keine Abhängigkeiten und Verbindungen zu anderen Tabellen. Im Folgenden werden die Tabellen für die Logs, die Notizen und die alten Spielstände vorgestellt.

Logs

Um die Aktionen innerhalb der Anwendung nachverfolgen zu können, werden diese in die ***logs*** Tabelle geschrieben. Jeder Eintrag besteht aus einer ID, einem Enum, in dem der Typ der Aktion festgehalten wird (*action*), einem Zeitpunkt (*timestamp*) und dem Inhalt der Aktion (*data*).

Innerhalb der ***logs*** Tabelle werden nur die Aktionen des derzeitig aktiven Spiels gespeichert. Neben diesen werden auch die Aktionen des letzten Spiels in der Tabelle ***logs_old*** gespeichert.

Sollte ein Spiel beendet werden, werden die Aktionen des alten Spiels mit den neuen Aktionen überschrieben.

Notizen

In der Tabelle ***notes*** soll den betreuenden und administrierenden Personen die Möglichkeit geboten werden, Notizen und Hinweise zum Rahmen und zur Durchführung des Versuches zu platzieren. Diese Notizen können durch die Studierenden eingesehen werden. Dafür werden Titel und Inhalt jeweils in einem String gespeichert. Um einzelne Notizen abrufen zu können, erhalten diese eine eindeutige ID.

Alte Spiele

Die Tabelle ***backups*** soll alle alten Spielstände speichern. Dazu wird jedem Backup eine eindeutige ID zugewiesen und der Zeitpunkt des Erstellens wird in der Spalte *created_at* abgespeichert. Die Informationen zu einem Spiel sind im JSON-Format in der Spalte *data* abgelegt.

3.7 Webclient

In diesem Abschnitt wird zuerst auf den Unterschied zwischen Multi Page Applications und Single Page Applications eingegangen, um eine Architektur für die neue Weboberfläche zu wählen. Im Anschluss wird das Design unter Zuhilfenahme von Mockups erläutert.

3.7.1 SPA vs MPA

Multi Page Applications Multi Page Application, kurz MPA, ist die klassische Architektur für Webanwendungen. Bei dieser Architektur wird für jeden Request (Anfrage) an den Webserver eine neue Seite inklusive Ressourcen wie Cascading Style Sheets (CSS)[1], JavaScript und Bildern geladen. Dieses kann an einem Beispiel verdeutlicht werden.

Auf einer Shop-Seite befinden sich 10 Produkte inkl. Bild und Kurzbeschreibung. Wird ein Produkt ausgewählt, sendet der Client eine Anfrage an den Webserver. Der Webserver antwortet mit allen Ressourcen (siehe oben), die für das Produkt benötigt werden. Der Client stellt dann aus den Ressourcen die Ansicht dar und das Produkt inklusive der Details ist für die Nutzenden zu sehen.

Der Vorteil von MPAs ist die einfachere Optimierbarkeit für Suchmaschinen, das sogenannte SEO (Search Engine Optimization). Ein gutes SEO-Rating sorgt dafür, dass die Webseite bei

[1] Beinhalten Regeln für die Darstellung von unter anderem HTML-Dokumenten

Suchmaschinen weit oben zu finden ist. Dies ist besonders für Webseiten wichtig, die um Kunden konkurrieren. Anzuführen sind hier diverse Online-Shops und Zeitungen.

Single Page Applications Die Single Page Application, kurz SPA, stellt das genaue Gegenteil von MPAs dar. Bei SPAs besteht die Anwendung aus genau einem HTML-Dokument, dessen Inhalt bei Bedarf dynamisch nachgeladen wird. Dafür findet ein asynchroner Datenaustausch zwischen Client und Server statt, bei dem benötigte Ressourcen wie Bilder, JavaScript und CSS ausgetauscht werden. Durch dieses Verfahren wird sichergestellt, dass gleiche Elemente oder Ressourcen nicht erneut heruntergeladen werden müssen. Bei Änderungen werden nur Teile des DOMs[2] ersetzt und neu gerendert.

Die Interaktion mit dem DOM oder auch Virtual DOM kann selbst entwickelt werden. Jedoch ist hierbei zu raten, auf bereits bestehende Frameworks wie Angular (Entwickelt unter der Leitung vom Angular Team bei Google), React (Entwickelt unter der Leitung von Facebook) oder Vue (Evan You und Core Team) zurückzugreifen.

Der große Vorteil von SPAs ist die Geschwindigkeit der Anwendung, da hier nur einzelne Teile ausgetauscht werden müssen. SPAs bieten außerdem den Vorteil, dass die Entwicklung von Front- und Backend entkoppelt wird. Das heißt, dass die Programmierenden des Front- und Backends weitestgehend unabhängig voneinander arbeiten können.

Die SEO-Optimierung gestaltet sich schwieriger, da es sich um eine dynamische Anwendung handelt. Zur Nutzung von SPAs muss im Browser JavaScript in der benötigten Version verfügbar und aktiviert sein.

[2]Das Document Object Model repräsentiert die Webseite als Baumstruktur

Zusammenfassung Vor- und Nachteile

	SPA	MPA
Vorteile	• Sehr schnell, dank dynamischem Nachladen • Entkoppelung zwischen Front- und Backend • Effizientes cachen von Daten	• MPA Architektur ist ausgereift • MPAs sind entwicklerfreundlich, da ein kleiner Technologiestack benötigt wird • Ältere Browser werden in der Regel unterstützt • SEO ist einfacher zu implementieren
Nachteile	• JavaScript muss in der benötigten Version im Browser verfügbar sein • Alte Browser werden nur teilweise unterstützt • Herausfordernde SEO Implementierung	• Anwendungen sind weniger performant als SPAs • Front- und Backend haben eine starke Kopplung

Bei der Entwicklung der Anwendung entscheide ich mich für die Verwendung einer SPA. Dies geschieht unter den Gesichtspunkten der Entkopplung zwischen Front- und Backend, der Performance der Anwendung und der Zukunftssicherheit, die meiner Meinung nach für SPAs besteht. Die Nachteile vom SPAs betreffen meine Anwendung nur wenig. So ist auf den Rechnern im Labor ein moderner Webbrowser installiert und in diesem JavaScript aktiviert. Auch handelt es sich um eine interne Anwendung, bei der die SEO Optimierung keine Rolle spielt. [Mel20]

3.7.2 Mockups

Für das Layout der Oberfläche wird die Designsprache Material Design von Google verwendet, da dieses konkrete Gestaltungsregeln vorgibt und einen Styleguide zur Verfügung stellt. Material Design wurde im Jahr 2014 vorgestellt und in Apps von Google verwendet. Besonders die Optimierung der Benutzerfreundlichkeit steht im Vordergrund. Eine Besonderheit des Material Designs ist, dass physikalische Gesetze beachtet werden und die Inhalte für eine Priorisierung auf einer virtuellen Z-Ebene verschoben werden. [kul17]

Den Prinzipien des Dark Theme von Material Design wird gefolgt, da die bestehende Anwendung bereits nur im Dark Mode vorhanden ist. Derzeit werden Anwendungen häufig mit

Dark Mode als Default Theme veröffentlicht und bestehende Anwendungen ohne Dark Mode erhalten diesen als zusätzliches Farbschema. [Col20]

Das Dark Themes trägt zur Verbesserung der visuellen Ergonomie bei, was zu einer Reduktion der Belastung der Augen führt. Bei OLED-Bildschirmen wird die Batterieleistung geschont. Die im Light Theme verwendete Verschiebung auf der Z-Ebene wird durch Schatten erreicht und ist deshalb im Dark Theme nicht nutzbar. Anstatt eine Verschiebung vorzunehmen, werden die Elemente unterschiedlich beleuchtet.[Goo]

Damit die Icons zum Design der Seite passen und einheitlich sind, werden Material Design Icons[3] verwendet. Hier sind die offiziellen Icons sowie Icons von der Community, die dem Material Design Ansatz folgen, vorhanden.

Abbildung 3.22: Grundlayout der Weboberfläche (Mockup)

Das Grundlayout ist wie in Abbildung 3.22 dargestellt in drei Bereiche, Header (1), Navigation (2) und Content (3), eingeteilt.

Der Bereich des Headers ist im oberen Teil erkennbar. Dort sind ein Logo, derzeitig noch das Logo aus der alten Anwendung, eine Überschrift sowie ein Untertitel platziert. Über die Überschrift und den Untertitel können die verschieden Seiten (Admin-, Spieler-, Flagshop- und Challengeseite) unterscheidbar gemacht werden.

Die Navigation ist auf der linken Seite erkennbar und als Sidebar mit Icons realisiert. Diese beinhaltet Verweise zu den Unterseiten. Sollte ein Eintrag der Navigation aktiv sein, wird dieser mithilfe der im Material Design vorgesehenen Farbe hervorgehoben.

Der Rest der Seite ist der Content Bereich. Dieser wird dynamisch mit dem anzuzeigenden Inhalt gefüllt.

[3] `https://materialdesignicons.com/`

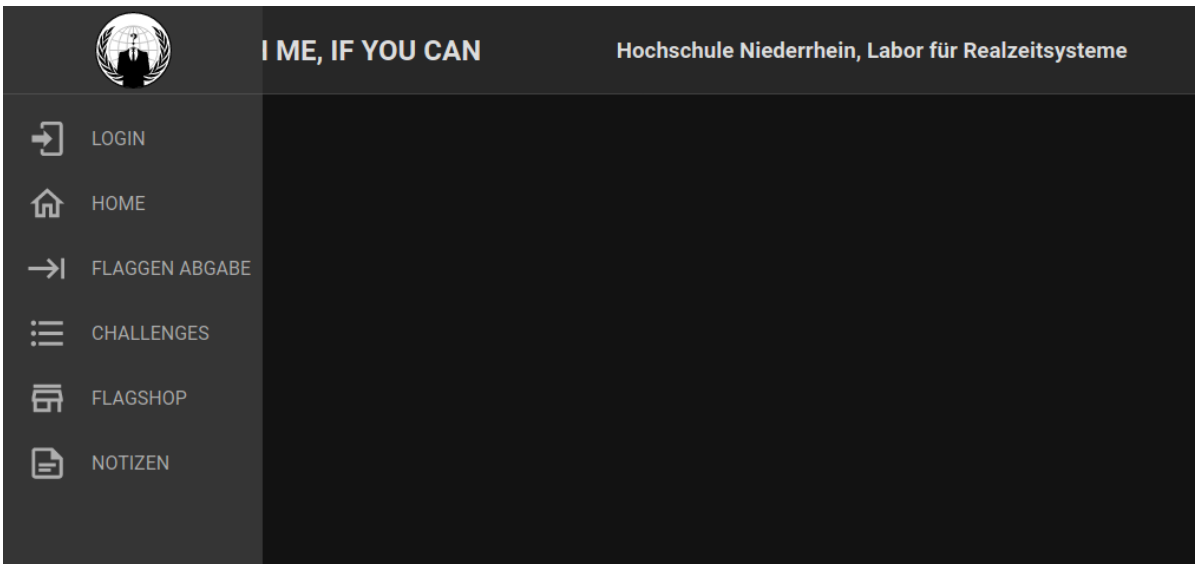

Abbildung 3.23: Ausgeklapptes Menü beim Drüberfahren (Mockup)

Damit die Navigation verständlich wird, gibt es – wie in Abbildung 3.23 zu erkennen ist – neben den Icons auch beschreibenden Text. Dieser Text wird angezeigt, wenn die Nutzenden mit der Maus über die Sidebar fahren. Die Sidebar überdeckt den Content.

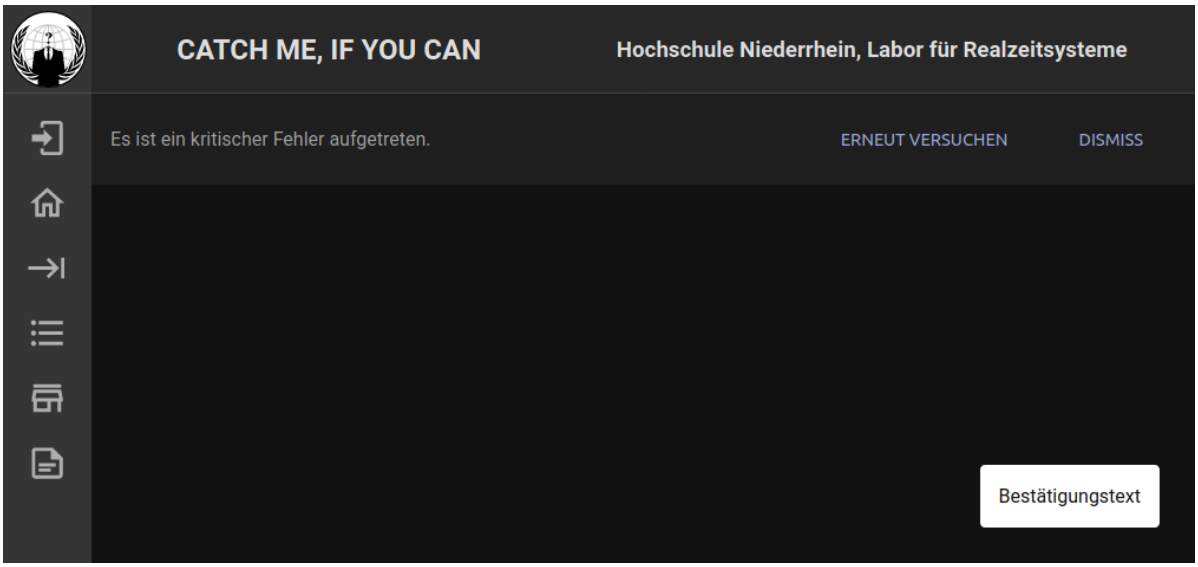

Abbildung 3.24: Fehlernachrichten (Mockup)

Sollten die Nutzenden über Ereignisse informiert werden, stehen zwei Möglichkeiten zur Verfügung.

Wenn das Ereignis wichtig ist, wird ein Banner verwendet. Dieses Banner besteht aus drei Elementen, einem Element für die Nachricht und zwei Buttons. Der Linke der beiden ermöglicht eine Aktion direkt auszuführen. Dies kann unter anderem eine Weiterleitung zu einer

bestimmten Seite sein, über die die Nutzenden weitere Informationen erhält. Über den anderen Button kann das Banner ausgeblendet werden. Es wird maximal ein Banner zeitgleich angezeigt.

Die andere Möglichkeit, die Nutzenden zu informieren, ist der Weg über die sogenannte Snackbar. Diese wird verwendet, wenn die Benutzererfahrung nicht unterbrochen werden soll und keine Benutzereingabe benötigt wird. Die Snackbar wird am unteren rechten Rand platziert und verschwindet im Gegensatz zum Banner nach einer definierten Zeit automatisch. Wenn möglich wird auch hier maximal eine Snackbar gleichzeitig verwendet.

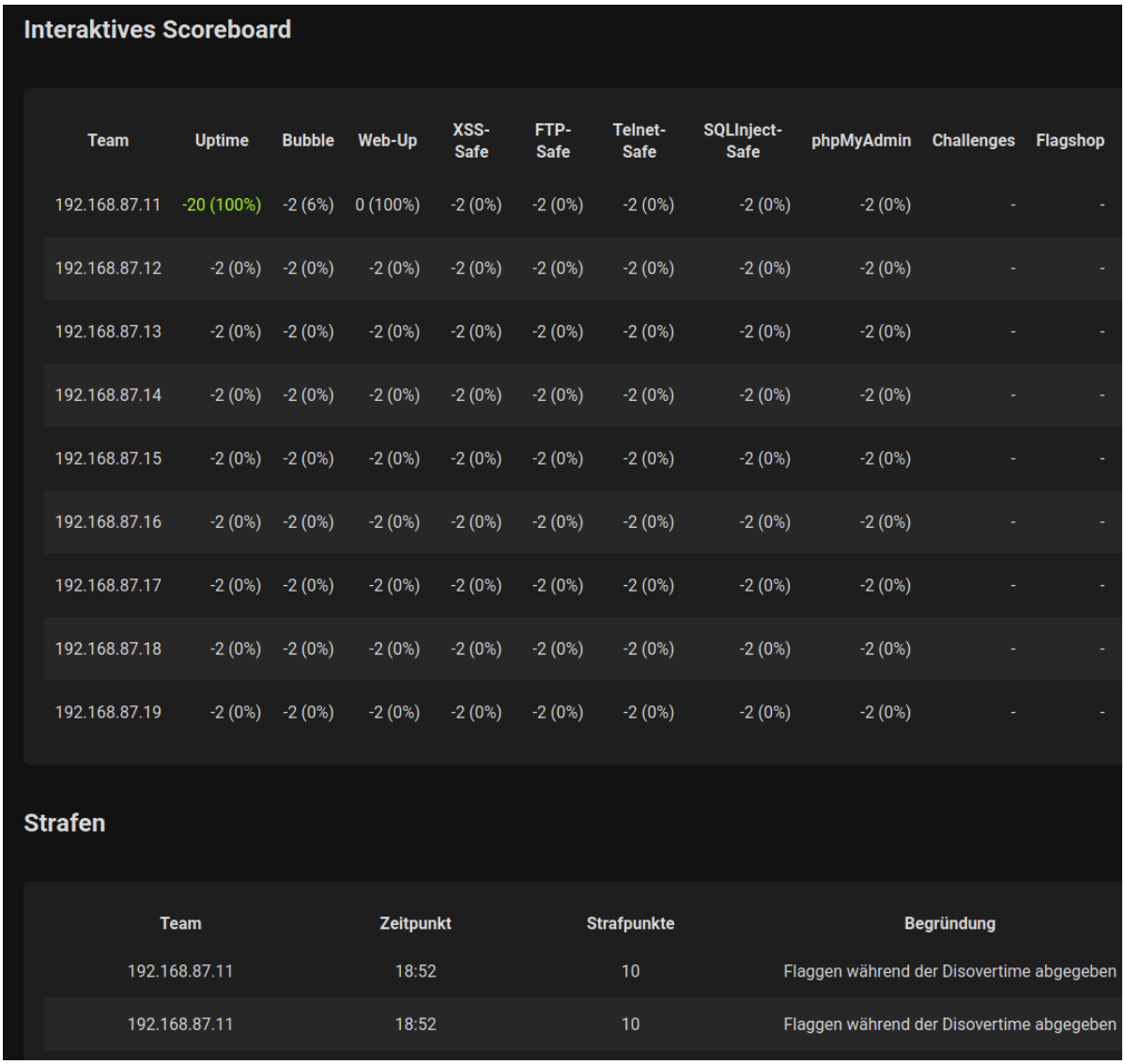

Abbildung 3.25: Ansicht des Spielstandes (Mockup)

In Abbildung 3.25 ist ein Teil der Ansicht des Interaktiven Scoreboards inklusive der Strafen abgebildet. Die Ansicht ist in das Grundlayout der Anwendung (Abbildung 3.22) eingebettet.

Wie bereits erwähnt, wird die Tabelle, um diese vom Hintergrund abzusetzen, unter Zuhilfenahme einer helleren Farbe dargestellt. Damit die Zeilen in der Tabelle besser erkennbar sind, ist jede zweite Zeile hervorgehoben.

Dienste, die beim letzten Scan den gewünschten Zustand hatten, werden mit einem kontrastarmen Grün dargestellt. Im Gegensatz zu der vorhanden Anwendung wird auf eine rote Darstellung bei einem ungenügendem Zustand der Dienste und Schwachstellen verzichtet. Einzig die Gesamtpunkte werden im negativen Bereich rot dargestellt.

3.8 GameClient

Der GameClient bleibt durch diese Bachelorarbeit unberührt. Es ist jedoch eine Änderung an ihm vorzunehmen, damit er mit dem *Game Information System* kompatibel ist. Am Spiel können nur noch Gruppen teilnehmen, die ihre Teilnahme gegenüber dem GIS bekunden. Bei dieser Anmeldung erwartet das GIS einen Token, der bei der Generierung der Flags verwendet wird.

In der derzeitig implementierten Startgame-Routine findet durch den GameClient keine Bekundung der Teilnahme statt, somit wird auch kein Token übermittelt.

Auch muss die Hashfunktion zur Generierung an die im GIS verwendete Funktion angepasst werden, wenn nicht der auf dem GameClient verwendete *MD5* Hash-Algorithmus implementiert ist. Dies ist notwendig, damit die generierten Flags vergleichbar bleiben.

4 Technologien

4.1 Backend

Bei der Entwicklung des Webservers wird auf ein Backend-Webframework zurückgegriffen. Das erleichtert die Entwicklung und Wartung der Webanwendung, indem Werkzeuge und Bibliotheken zur Verfügung gestellt werden, die allgemeine Aufgaben übernehmen. Zu diesen allgemeinen Aufgaben gehören das Verarbeiten von HTTP-Anfragen, das Erstellen von HTTP-Antworten und das Weiterleiten von Anfragen an die entsprechenden Funktionen. Bei der Implementierung der Logik müssen alle diese allgemeinen Aufgaben nicht mehr umgesetzt werden. [cM20b]

Um die Weiterentwicklung durch Studierende des Bachelor Informatik-Studiengangs der Hochschule Niederrhein zu gewährleisten, werden nur Webframeworks, die eine Programmierung in C, C++, Python oder JavaScript vorsehen, in Betracht gezogen, da eben diese an der Hochschule Niederrhein gelehrt werden. Derzeitig sind neben Frameworks wie Spring (Java), Ruby on Rails (Ruby) und Laravel (PHP), die nicht in den Vergleich aufgenommen werden, Django (Python), Flask(Python) und Express (Node.js/JavaScript) verbreitet. [cM20b]

In der Lehrveranstaltung *Web-Engineering* wird die Entwicklung mehrerer Backend-Server unter Zuhilfenahme von Python durchgeführt. Deshalb werden im folgenden nur Django und Flask betrachtet.

Anschließend wird ein Framework gewählt, mit dem der Backend-Server realisiert werden soll.

4.1.1 Django

Django folgt der sogenannten *Batteries Included* Philosophie.

Bei dieser werden vielseitige Standardbibliotheken mit ausgeliefert, sodass eine nutzende Person keine oder nur wenige separate Pakete herunterladen muss. [Kuc]

Django liefert unter anderem Module für Caching, Logging, Versenden von E-Mails, RSS-Feeds, Pagination, Security und der automatischen Generierung von Administrationsseiten. [Dja] Durch den *Batteries Included* Ansatz funktionieren die Komponenten reibungslos untereinander und können die Kernanwendung problemlos erweitern. Die Dokumentationen der verschiedenen Module und Erweiterungen sind zentral an einem Ort verfügbar.

Django wurde von einem Team entwickelt, das ursprünglich Webseiten von Zeitungen erstellt und verwaltet hat. Dieses Team hat die gemeinsame Codebasis abstrahiert und in ein Framework überführt. Daher ist Django besonders gut, aber nicht nur für Nachrichtenseiten und Content Managment Systeme (CMS) geeignet. [cM19]

4.1.2 Flask

Im Gegensatz zu Django ist Flask ein Mikroframework und verfolgt die *Batteries Included* Philosophie nicht. Bei einem Mikroframework ist der Kern einfach, aber erweiterbar gestaltet und es gibt wenige bis keine Abhängigkeiten von externen Bibliotheken. Dieser Ansatz überträgt der programmierenden Person mehr Verantwortung aber auch mehr Freiheiten. So trifft Flask beispielsweise nicht die Entscheidung, welche Datenbank genutzt werden soll. [Pal10b]

Wie bei Django (Unterabschnitt 4.1.1) werden zwei Module bei der Installation von Flask mit geladen. Zum einen wird auf das Modul *Werkzeug* zurückgegriffen, um eine ordnungsgemäße WSGI-Anwendung zu ermöglichen. Zum anderen wird Jinja2 mitgeliefert, da die meisten Anwendungen eine Template-Engine benötigen. Flask will eine solide Grundlage für alle Anwendungen sein, bei der die entwickelnde Person viele Freiheiten genießt. [Pal10a]

4.1.3 Wahl des Frameworks

Das Mikroframework Flask wird zur Implementierung des Webservers genutzt. Die Anwendung wird in Python geschrieben und kann daher durch Studierende weiterentwickelt werden, ohne dass diese sich in eine neue Programmiersprache einarbeiten müssen. Außerdem werden viele Module von Django nicht benötigt. Sie stellen einen Overhead dar. Auch handelt es sich bei der zu entwickelnden Software um eine vergleichbar kleine Anwendung. Es würden nur wenige Vorteile von Django genutzt. Flask kann mit Hilfe von Community-Modulen bei Bedarf erweitert werden.

4.2 Datenhaltung

Es gibt eine Vielzahl von verschiedenen relationalen Datenbankmanagementsystemen, die alle einen ähnlichen Funktionsumfang mitbringen.

Im Folgenden werden nur die kostenlos nutzbaren und am weitesten verbreiteten Datenbanksysteme dargestellt. Zu dieser Liste zählen MySQL (Oracle), PostgreSQL (PostgreSQL Global Development Group), MariaDB (MariaDB Foundation) und SQLite (Dwayne Richard Hipp). [DB-]

Im Anschluss an die Darstellung wird die Wahl der Datenbank für die Anwendung begründet.

4.2.1 MySQL und MariaDB

MySQL ist das beliebteste Open-Source SQL Datenbankmanagementsystem, das im Jahr 1995 veröffentlicht worden ist. Dieses ist als Client/Server Architektur implementiert, kann aber auch in eingebetteten Systemen verwendet werden. [Ora20a] MySQL ist in C / C++ geschrieben und kompatibel mit vielen Programmiersprachen, darunter auch Python. Es ist auf diversen Plattformen einsetzbar und für große Datenmengen optimiert. [Ora20b]

MariaDB ist eine Abspaltung von MySQL. Sie resultierte aus der bevorstehenden Übernahme von MySQL durch Sun Microsystems unter der Führung von Oracle. [ION20]

4.2.2 PostgreSQL

PostgreSQL ist frei verfügbar und mit einer kostenlosen Lizenz nutzbar. Es wurde als universitäres Projekt an der University of California at Berkeley Computer Science Department gestartet. Es sind neben den SQL92 und SQL99 Standards auch einige eigene Erweiterungen implementiert. [BB]

Nach Angaben der Entwickler ist „PostgreSQL [...] heute die fortschrittlichste Open-Source-Datenbank, die es gibt.“ ([The20])

4.2.3 SQLite

Im Gegensatz zu den bereits vorgestellten Datenbanken handelt es sich bei SQLite um eine serverlose Datenbank, die nicht aufgesetzt oder verwaltet werden muss. SQLite speichert die Daten in einer gewöhnlichen Datei, die kopiert und verteilt werden kann. [SQL]

4.2.4 Wahl der Datenbanksoftware

Die PostgreSQL Datenbank wird den Alternativen vorgezogen, da es für den Anwendungsfall keine nennenswerten Unterschiede zwischen den Datenbanken gibt. Die erhöhte Komplexität der Installation von PostgreSQL im Vergleich zu SQLite kann durch die Nutzung eines Dockerimages reduziert werden. Die Studierenden haben sich mit der PostgreSQL Datenbank in der Veranstaltung *Datenbanksysteme* beschäftigt und im EZS-Labor besitzen bereits Mitarbeiter Kenntnisse von PostgreSQL.

4.3 Frontend

Für die Realisierung des Frontends sollte aus den in Kapitel 3 genannten Gründen auf ein bestehendes Framework zurückgegriffen werden. In der Betrachtung werden die drei beliebtesten JavaScript Frameworks, laut einer Stackoverflow (Frage- und Antwortseite für Programmierende) Umfrage, diskutiert. Die Umfrage, bei der circa 65.000 Menschen teilgenommen haben, wurde im Februar 2020 durchgeführt. [Sta20]

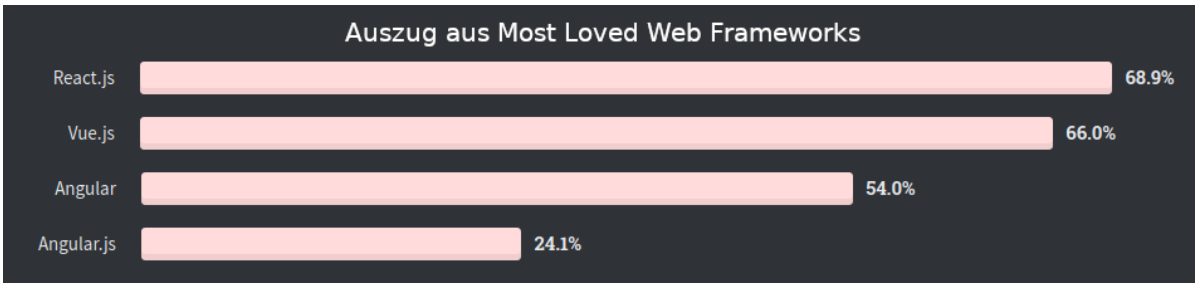

Abbildung 4.1: Weiterverwendung des benutzten Webframeworks (Screenshot) [Sta20]

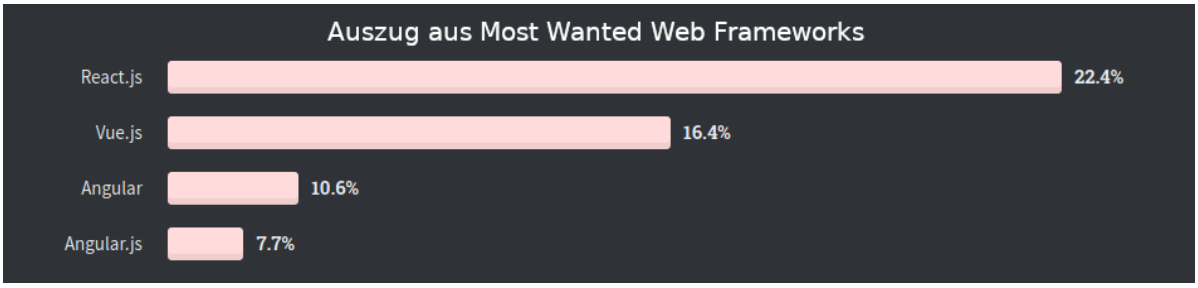

Abbildung 4.2: Interesse an einem neuen Webframework (Screenshot) [Sta20]

In der Abbildung 4.1 ist zu sehen, wie groß der Anteil der Entwickelnden (in Prozent) ist, die bereits mit der Technologie arbeiten und ein Interesse bekundet haben, diese für weitere Projekte zu nutzen. React.js führt diese Liste mit 69 % an. Darauf folgt Vue.js dicht mit 66 %. Bei Angular sieht es etwas anders aus, hier bekunden nur 54 % ein Interesse. Im Gegensatz dazu wollen ∼25 % der Befragten den Vorgänger Angular.js weiter nutzen.

Die Abbildung 4.2 zeigt den Prozentanteil der Entwickelnden, die ein Interesse an der Technologie, bisher aber noch nicht mit dieser gearbeitet haben. Auch hier ist React.js mit ∼22 % führend. Etwas abgeschlagen mit ∼16 % folgt Vue.js. Dahinter reihen sich Angular mit ∼11 % und Angular.js mit 8 % ein.

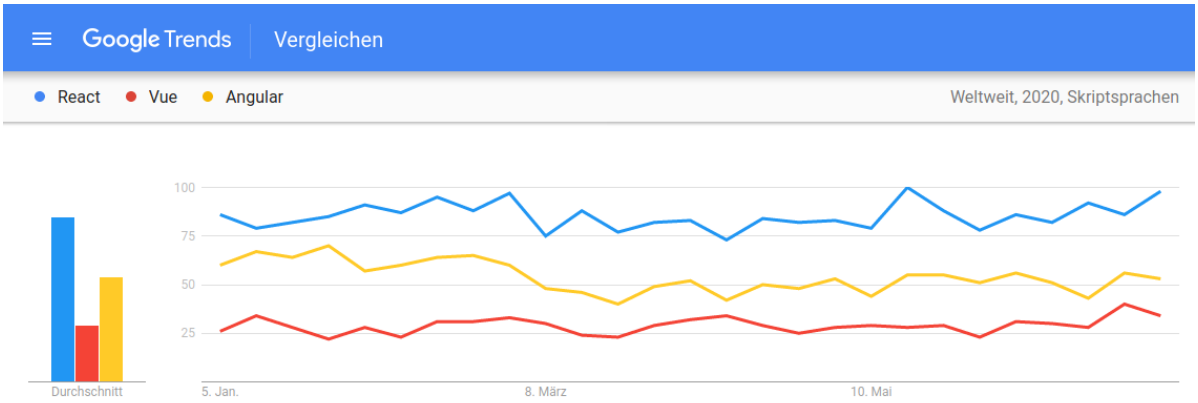

Abbildung 4.3: Google Trends (Screenshot) [Goo20b]

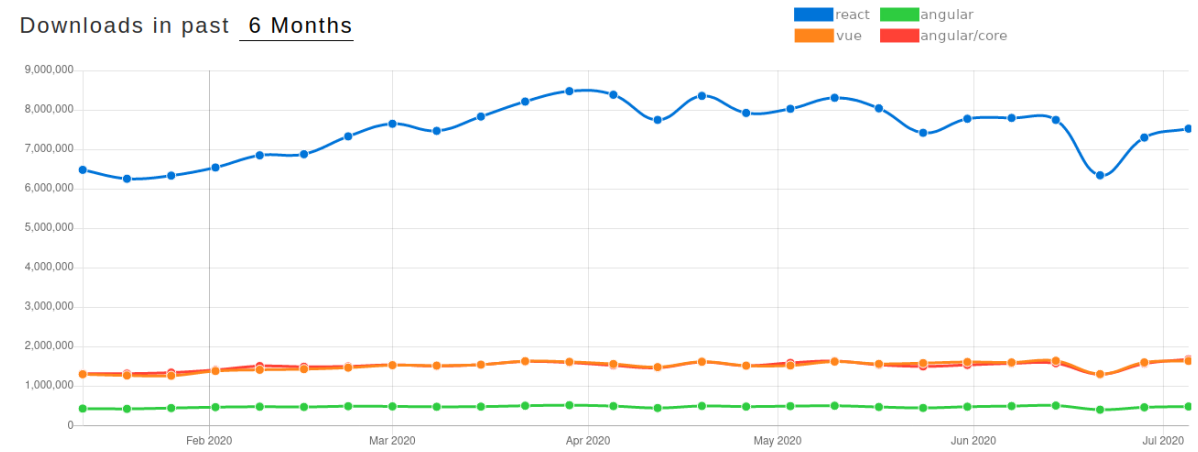

Abbildung 4.4: NPM Trends (Screenshot) [Pot20]

Die Google Trends in Abbildung 4.3 zeigen das aus Stichproben normalisierte Interesse an einem Begriff vom Januar 2020 bis Anfang Juli 2020. Das Interesse wird von Google stichpunktartig anhand von Google Suchanfragen gemessen. [Goo20c]

In der Abbildung 4.4 werden die wöchentlichen Downloads pro Framework grafisch dargestellt. In dieser ist erkennbar, dass Vue.js und Angular ungefähr gleich oft heruntergeladen werden. Generell ist eine Stagnation der Frameworks erkennbar, einzig bei React.js schwanken die Downloadzahlen zwischen 6,5 und 8,5 Millionen. Derzeitig scheinen sich die Downloads bei ungefähr 7 Millionen pro Woche eingependelt zu haben.

Anhand der Abbildung 4.1, Abbildung 4.3 und Abbildung 4.4 lässt sich schlussfolgern, dass React.js derzeitig das beliebteste und auch das am meisten verbreitete Framework ist. Das unbeliebteste Framework der Auswahl scheint Angular.js zu sein. Anhand der Grafiken fällt eine klare Positionierung von Vue.js und Angular zwischen React und Angular.js schwer.

Im Folgenden sollen die Frameworks genauer betrachtet werden, um eine Nutzung unter anderem anhand des Funktionsumfangs, der Möglichkeiten sowie der Komplexität zu beurteilen.

4.3.1 Angular

AngularJS wurde von einem Google Mitarbeiter (Misko Hevery) als Seitenprojekt gestartet. Er wollte dieses nutzen, um interne Webanwendungen einfacher zu entwickeln. Nachdem einige interne Anwendungen mit diesem Projekt erstellt worden sind, wurde AngularJS im Jahr 2010 als Open-Source-Projekt veröffentlicht. Es fand sowohl in Unternehmensprojekten als auch in Projekten der Open-Source Community Anwendung. [Gav18]

Einige Jahre nach der Veröffentlichung begann sich die Landschaft der Webentwicklung durch Weiterentwicklungen und neue Standards in JavaScript zu verändern. Neben dieser Veränderung ist das Team an Grenzen gestoßen, die die Verbesserung des Frameworks anging. Diese Entwicklung führte dazu, dass AngularJS über die Zeit an Modernität verlor. [Gav18]

AngularJS war im Entwurfsstadium nie für Mobile- oder Unternehmensanwendungen gedacht. So entschied sich das Team bei Google mit der Entwicklung der Version 2 einen Neuanfang zu wagen und Angular 2 für solche Anwendungsfelder zu entwerfen. [Gav18]

Um Verwirrungen zu vermeiden, wurde das bis dahin bekannte Angular in „AngularJS“ umbenannt und die neue Version „Angular 2“ genannt. Die Versionierung von Angular folgt seit Version 2 dem Semantic Versioning Prinzip, bei dem die Versionsnummer in „MAJOR.MINOR.PATCH“ ([Pre]) untergliedert wird. [Gav18]

In der Anfangsphase von Angular 2 war es umständlich ein neues Projekt anzulegen, da die Build-Größen vergleichsweise groß waren. So erarbeitete sich Angular den Ruf sperrig zu sein. Neben diesem wurde die Sprache von JavaScript zu TypeScript (Obermenge von JavaScript) umgestellt, das unter anderem eine andere Syntax und neue Konzepte zur Folge hatte. [Gav18]

Am 24.06.2020 erschien Angular 2 in der Version 10.0.0 [Goo20a] und zählt wie bereits zuvor dargestellt zu den beliebtesten Frontend-Frameworks.

Im Folgenden wird nur das aktuell beworbene Angular 2 dargestellt und Angular genannt.

Bei der Entwicklung von Angular wurde auf Konsistenz, Produktivität, Wartbarkeit, Modularität und frühzeitige Fehlererkennung gesetzt.

Angular basiert auf Komponenten und Diensten, die alle identische Strukturen besitzen. Auch gibt Angular vor, wie ein wiederverwendbarer Code angelegt wird, sodass hierfür keine Code Styles notwendig sind. Außerdem erhält die Angular Dokumentation einen Style-Guide und die Unternehmen müssen keinen eigenen Style-Guide erstellen, was eine Zeitersparnis bedeutet. Durch diesen konsistenten Aufbau wird auch eine bessere Wartbarkeit gewährleistet. [Wah17]

Durch die Konsistenz, die gegebenen Style-Guides und den wiederverwendbaren Code steigt die Produktivität, da die Entwickelnden eingeschränkter sind und nicht überlegen und bewerten müssen, ob das aktuell angewandte Vorgehen richtig ist. [Wah17]

Die Wartbarkeit der geschriebenen Anwendung profitiert von der Konsistenz und den Style Guides, da so jeder, der involviert ist, den Code leichter nachvollziehen und verändern kann. Derzeit ist durch das große Interesse der Community sowie der internen Nutzung bei Google die Wartbarkeit des Frameworks gesichert. [Wah17]

Bei Angular wird der Code in einzelnen Modulen organisiert. Durch die Nutzung des modularen Systems können diverse Teams derselben Anwendung arbeiten, ohne sich gegenseitig zu behindern. [Wah17]

Die frühzeitige Fehlererkennung wird durch die Nutzung von TypeScript ermöglicht. TypeScript bietet im Gegensatz zu JavaScript Unterstützung für die Typisierung. Die Nutzung der Typen ist in TypeScript nicht vorgeschrieben, aber zur Fehlererkennung empfohlen. Des Weiteren ermöglicht das Angular-CLI eine einfache Methode, um Unittests und End-zu-End-Tests durchzuführen; so können Fehler entdeckt und behoben werden. [Wah17]

Diese Vorgaben führen zwar zu einer besseren Wartbarkeit und einer konsistenten Anwendung, schränken aber gleichzeitig Entwickler ein. Das Gerüst ist unflexibel gehalten, um eine konsistente Anwendung zu ermöglichen.

Da Angular in Unternehmensanwendungen verwendet werden soll, besitzt es eine gewisse Komplexität und benötigt daher eine Einarbeitungszeit. Diese steile Lernkurve muss von neuen Programmierenden bewältigt werden. [Ven18]

4.3.2 React

Im Jahr 2010 startete React als Port von XHP (PHP-Erweiterung, die XSS Angriffe vorbeugen sollte) bei Facebook. Da XHP Probleme mit dynamischen Webanwendungen nicht löste, bat ein Facebook Software-Engineer um Zeit, damit er XHP unter Zuhilfenahme von JavaScript in den Browser bringen könne. [Daw14]

Anfänglich wurde React noch unter „BSD + patents“ lizenziert. Diese Lizenz gewährte eine Nutzungslizenz und eine Patentlizenz, solange von einer Klage gegen Facebook bei möglichen Patentverletzungen abgesehen wird. Am 25. September 2017 wurde die Lizenz nach Kritik auf die „MIT“-Lizenz umgestellt. [Kri17] [Lar17] Die MIT-Lizenz ist eine sehr wenig restriktive Lizenz und gibt die Erlaubnis zur Veränderung, Modifikation und Nutzung ohne Kosten. Auch beinhaltet diese kein Copyleft. Bei einem Copyleft muss das entwickelte Produkt (hier die Software), das die Software mit Copyleft nutzt, unter derselben Lizenz veröffentlicht werden. [Weh20]

React besteht aus Komponenten und nutzt eine Syntaxerweiterung von JavaScript namens JSX, um JavaScript und HTML zusammenzuführen. Die JSX-Dateien müssen vor der Auslieferung an den Client zu JavaScript transformiert werden. Reacts Komponenten sollten möglichst klein sein und nur eine Aufgabe erledigen. Dies führt zu einer besseren Test- und Wartbarkeit sowie einem besseren Verständnis der Anwendung. [Bez15]

Auch nutzt React für seine React Views einen sogenannten Virtual DOM (Document Object Model). Die Nutzung der React Views ermöglicht es, die Anwendungen auf dem ausliefernden Server zu rendern. Neben diesem können Unterschiede zwischen DOM und vDOM erkannt werden. Dadurch muss nicht mehr die gesamte Seite, sondern nur die veränderte Komponente neu gerendert werden. Dies resultiert in einer besseren Performance. [Bez15]

Um beispielsweise eine SPA in React zu entwickeln, muss auf Drittanbieter und deren Erweiterungen zurückgegriffen werden oder die entsprechenden Funktionen müssen selbst programmiert werden, da diese nicht in React enthalten sind. Dies stellt entweder Zeitaufwand oder Abhängigkeiten von weiteren Entwicklern dar.

4.3.3 Vue

Vue.js wurde durch einen damaligen Mitarbeiter von Google (Evan You) entworfen und entwickelt, nachdem er viel mit Angular gearbeitet hatte. Er wollte nur einen ganz kleinen Teil aus Angular nutzen. Er begann zunächst nur für sich zu entwickeln, veröffentlichte aber im Februar 2014 die Version 1.0.0 auf GitHub. [Teu18a]

Das Projekt gewann Beliebtheit auf GitHub und die Community beteiligte sich. Die ersten Plugins wurden geschaffen und es bildete sich ein Core Team.

Vue liefert im Gegensatz zu den anderen vorgestellten Frontend-Frameworks nur das Data Binding und die Möglichkeit, Komponenten zu programmieren, im Kern mit. Zusätzlich benötigte Funktionen können bei Bedarf als Module geladen werden. Dies macht Vue zu einem spezialisierten, aber auch universell nutzbaren Framework. Es ist nicht so universell nutzbar wie Angular, aber auch nicht so spezialisiert wie React. [Teu18a]

Die Dokumentation von Vue ist leicht verständlich und die Community hilft schnell bei Fragen. Anwendungen in Vue sind in einer hierarchischen Struktur von kleinen Komponenten untergliedert. Dies soll ermöglichen, dass die Komponenten und so das gesamte System besser zu verstehen sind. Redundanz ist dort zulässig, wo diese benötigt wird. Komponenten erhalten neben dem reinen HTML auch CSS und JavaScript und sollen atomar sein. Das heißt, dass diese alleine und ohne Bindungen zu einer Komponente auf einer logisch gesehen höher liegenden Ebene nutzbar sind. Durch diesen Ansatz ist die Software übersichtlicher, was zu einer besseren Test- und Wartbarkeit führt. [Teu18b]

Damit diese Trennung nicht nur in der Software, sondern auch auf der Dateiebene passiert, nutzt Vue hierfür die sogenannten Single File Components. Diese Dateien enden mit der Dateiendung *.vue* und sind erst nach einer „Kompilierung“ nutzbar. Eine Single File Component trennt HTML, CSS und JavaScript auf, sodass diese Teile übersichtlich betrachtetet werden können. [Teu18b]

Ein Nachteil von Vue ist, dass es von keiner großen Firma entwickelt wird, die die Entwicklung vorantreibt und die Verbreitung erhöht. Außerdem werden Angular und React intern selber genutzt, sodass das Framework eine langfristige Nutzung hat. Allerdings nutzen seit

circa 2016 immer mehr Unternehmen Vue. Hier sind beispielsweise GitLab (2016, [Sch16]), Laravel und Nintendo anzuführen. [tec18]

4.3.4 Wahl des Frontend-Frameworks

Für das Frontend sollte entweder Vue oder React genutzt werden, da diese einsteigerfreundlicher als Angular sind. Angular wird auf Grund der Komplexität und des Umfangs des zu schreibenden Frontends nicht in Betracht gezogen. Sowohl für Vue als auch für React sprechen ausreichend viele Gründe.

Vue ist schlanker und etwas leichter zu erlernen als React, da hier nur Kenntnisse in JavaScript, CSS und HTML benötigt werden. Bei React werden noch zusätzliche Kenntnisse in JSX benötigt.

Für React spricht die Verbreitung und die damit größere Community, die bei Problemen konsultiert werden kann. Auch ist es wahrscheinlicher, dass Problemstellungen bereits auf Frage-Antwort Plattformen wie StackOverflow behandelt worden sind.

Schlussfolgernd sollte für das Frontend React genutzt werden, da hier eine größere Community bei Fragen zur Verfügung steht.

5 Realisierung

Die Realisierung ist in drei Teile untergliedert. Zuerst werden die genutzten Datenbanktabellen und -views aufgezeigt. Der zweite Teil beschäftigt sich mit der Umsetzung des Big Brothers. Zuletzt wird die Implementierung des GIS erläutert. Diese Vorgehensweise wurde aufgrund der vorhandenen Abhängigkeiten gewählt.

Der Webclient wurde aus Zeitgründen nicht implementiert und wird daher auch nicht in der Realisierung aufgegriffen.

5.1 Datenbank

Die Realisierung des Datenbankschemas wird über Migrationen in der Komponente des Game Information Systems durchgeführt. Sie wird aus Verständnisgründen in diesem Kapitel erläutert.

Die im Entwurf (Abschnitt 3.6) vorgestellten Tabellen wurden ohne Anpassungen realisiert. Da die Punkteberechnung nicht im Game Information System durchgeführt werden soll, werden Views in der Datenbank genutzt, die die benötigten Punkte aus den verschieden Tabellen zusammenrechnen.

Eine SQL-Datei zur Erstellung des Datenbank-Schemas befindet sich auf dem mit der Bachelorarbeit abgegebenen Datenträger und wurde aber aus Platzgründen weder hier noch in den Anhang aufgenommen.

5.1.1 Punkteübersicht

Die folgenden Views wurden in der Datenbank erstellt, um die Punkteberechnung zu ermöglichen. Die Punkteberechnung wurde kleinschrittig gehalten, um eine Nachvollziehbarkeit zu gewährleisten.

Angriffspunkte

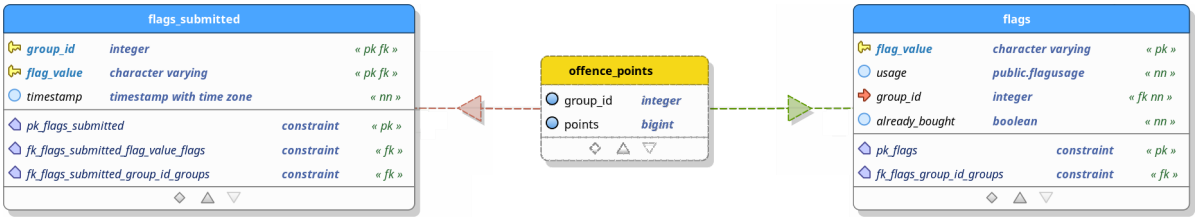

Abbildung 5.1: View Angriffspunkte (ER-Diagramm)

Wie in Abbildung 5.1 erkennbar, werden die Angriffspunkte (*offence_points*) über die Tabellen *flags* und *flags_submitted* berechnet. Dies erfolgt, indem alle abgegebenen Flags mit den im Spiel vorhandenen Flags verbunden werden (*JOIN*), um nur Flags zu beachten, die von einer anderen Gruppe als der Besitzenden abgegeben wurden. Im Anschluss werden diese pro Gruppe gezählt und als Offensivpunkte ausgewiesen.

```
1 SELECT flags_submitted.group_id,
    ↪ count(flags_submitted.flag_value) AS points
2 FROM (flags_submitted JOIN flags ON
    ↪ (((flags_submitted.flag_value)::text =
    ↪ (flags.flag_value)::text)))
3 WHERE (flags_submitted.group_id <> flags.group_id)
4 GROUP BY flags_submitted.group_id;
```

Listing 5.1: SQL View Angriffspunkte

Defensivpunkte

Ähnlich den Angriffspunkten (5.1.1) werden die Defensivpunkte (Strafe für verlorene Flags) berechnet. Anders als bei den Angriffspunkten werden alle vorhandenen Flags mit den abgegebenen Flags für jede Gruppe verbunden, wenn die besitzende Gruppe ungleich der abgebenden Gruppe ist. In dieser Liste treten Flags mehrfach auf, wenn mehrere Gruppen einer Gruppe dieselbe Flag gestohlen haben. Da im alten System ein Verlust einer Flag an mehrere Gruppen nur einmal mit Strafpunkten geahndet worden ist, werden in der Liste der Flags mit der SQL Klausel *DISTINCT* Duplikate entfernt. Im Anschluss werden dann die verlorenen Flags pro Gruppe gezählt und so die Defensivpunkte bestimmt.

```
1 SELECT flags.group_id, count(DISTINCT flags.flag_value) AS
    ↪ points
2 FROM (flags JOIN flags_submitted ON
    ↪ (((flags.flag_value)::text =
    ↪ (flags_submitted.flag_value)::text)))
3 WHERE (flags.group_id <> flags_submitted.group_id)
```

```
4 GROUP BY flags.group_id;
```

Listing 5.2: SQL View Denfensivpunkte

Erkundungspunkte

Die Erkundungspunkte werden nach dem gleichen Vorgehen wie die Angriffspunkte (5.1.1) bestimmt. Einziger Unterschied ist, dass nur Flags beachtet werden, bei denen die besitzende Gruppe identisch der abgebenden Gruppe ist.

```
1 SELECT flags_submitted.group_id,
    ↪ count(flags_submitted.flag_value) AS points
2 FROM (flags_submitted JOIN flags ON
    ↪ (((flags_submitted.flag_value)::text =
    ↪ (flags.flag_value)::text)))
3 WHERE (flags_submitted.group_id = flags.group_id)
4 GROUP BY flags_submitted.group_id;
```

Listing 5.3: SQL View Erkundungspunkte

Strafpunkte

Die View der Strafpunkte (*penalty_points*) summiert die in der *penalties*-Tabelle vorhandenen Strafpunkte pro Gruppe. Andere Informationen wie Grund der Strafe bleiben hierbei unberücksichtigt. Jede Gruppe (*group_id*) hat entsprechende Strafpunkte (*total_penalty*). Sollte eine Gruppe bisher noch keine Strafpunkte erhalten haben, wird die Gruppe in der View nicht aufgeführt.

Servicepunkte

In der View *group_service_points* werden pro Gruppe pro Service die Servicepunkte sowie die Serviceprozentzahl berechnet. Dazu werden Daten aus den Tabellen *services* und *group_service_status* benötigt. Aus der Tabelle *services* wird einzig die Gewichtung der einzelnen Services abgefragt. In der Tabelle *group_service_status* werden die Anzahl der durchgeführten sowie der erfolgreichen Scans pro Gruppe abgerufen.

Die Servicepunkte dienen als Leistungsindikator der Absicherung und der Instandhaltung des Systems durch die Studierenden. Dazu wird von der Anzahl der erfolgreichen Scans die Anzahl aller Scans abgezogen und das Ergebnis der Subtraktion gewichtet, um einen geringeren Einfluss auf die Gesamtpunkte zu ermöglichen.

$$\frac{Anzahl\ erfolgreicher\ Scanvorgaenge - Anzahl\ Scanvorgaenge}{Gewichtung} = Strafpunkte$$

Die Serviceprozentzahl stellt für die Studierenden und die für betreuenden Personen die prozentuale Erreichbarkeit dar. Für die Berechnung dieser Prozentzahl pro Gruppe und Service wird die folgende Formel verwendet:

$$\frac{100}{Anzahl\ erfolgreicher\ Scanvorgaenge} * Anzahl\ Scanvorgaenge = Wert\ in\ \%$$

Um das Ergebnis einer Division durch 0 zu verhindern, wird im SQL Quelltext die Funktion *COALESCE* verwendet. Diese nimmt den ersten Wert ungleich Null. Bei der Punkteberechnung kann durch eine Gewichtung von 0 die Division ein Ergebnis von Null zurückgeben. Deshalb wird die Gewichtung standardmäßig auf 1 gesetzt. Wenn eine Gruppe keine erfolgreichen Scanvorgänge zu melden hat, wird das Ergebnis der Formel für die Serviceprozentzahl auf 0 gesetzt.

```
1 COALESCE(NULLIF((services.weight)::integer, 0), 1)
2
3 COALESCE(((100 / group_service_status.scan_count) *
    ↪ group_service_status.online_count), 0)
```

Listing 5.4: SQL Abfang von Division durch 0

Flagshoppunkte

Bei den Flagshoppunkten handelt es sich um die für Käufe ausgegebenen Punkte einer Gruppe. Für die Berechnung wird die Summe der Kosten aller gekauften Pakete pro Gruppe gebildet. Wenn eine Gruppe keine Ausgaben getätigt hat, wird diese auch nicht in die View aufgenommen.

Challengepunkte

Im Gegensatz zum alten Überwachungs- und Auswertungssystem wird der Start einer Challenge nicht länger mit Minuspunkten bestraft. Diese Minuspunkte besitzen meiner Meinung nach eine abschreckende Wirkung auf die Studierenden, sodass diese die Challenges nicht versuchen. Im Rahmen einer Lehrveranstaltung sollten die Studierenden ermutigt werden, verschiedene themenbezogene Aufgaben zu lösen.

Die Challengepunkte werden mithilfe der Tabellen *challenges* und *group_has_challenges* in der View *challenge_points* bestimmt. Für jede Gruppe werden die Punkte der gelösten Challenges summiert.

Sollten Negativpunkte für gestartete Challenges verteilt werden, muss der SQL Code der View angepasst werden, damit für jede gestartete Challenge Negativpunkte verrechnet werden.

Gesamtpunkte

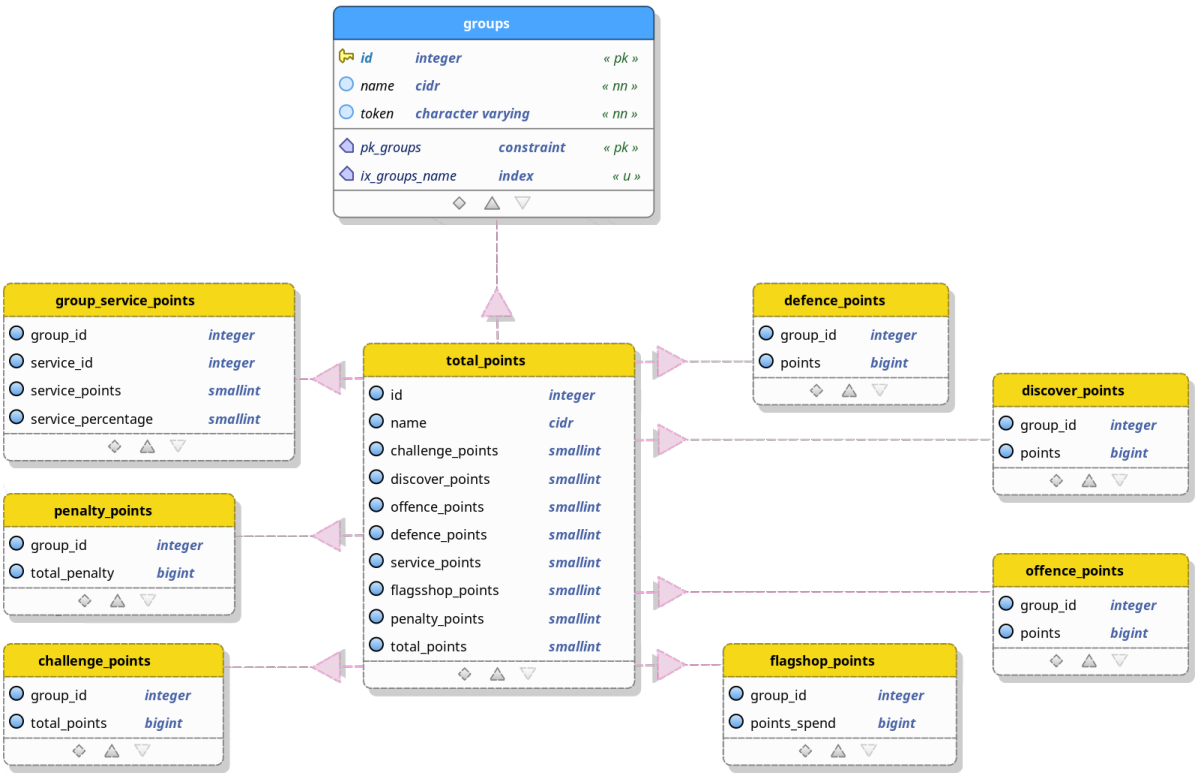

Abbildung 5.2: View Gesamtpunkte (ER-Diagramm)

In der View *total_points* werden pro Gruppe alle vorher genannten Punkte inklusive einer Gesamtpunkteanzahl dargestellt.

Die Gesamtpunkte setzen sich aus den genannten Punkten zusammen. Die Gesamtpunkte werden anhand der folgenden Formel berechnet:

$$Challengepunkte + Erkundungspunkte + Offensivpunkte - Deffensivpunkte - \\ Servicepunkte - Flagshoppunkte - Strafpunkte = Gesamtpunkte$$

Die Punkte werden aus den vorher dargestellten Views bezogen. Sollten Punkte nicht vorhanden sein, da beispielsweise eine Gruppe keine Challenges gelöst hat, werden diese Punkte durch die Nutzung von *COALESCE* auf 0 gesetzt.

```
1 COALESCE(challenge.points, ((0)::smallint))
```

Listing 5.5: SQL Ersetzen nicht vorhandener Punkte

5.1.2 Servicestatus

Der aktuelle Servicestatus wird auch innerhalb einer View (*group_service_online_status*) ermöglicht. Diese View blendet nur Informationen der Tabelle *group_service_status* aus, deshalb hätte auch die Tabelle selbst genutzt werden können.

Die View stellt pro Gruppe und pro Service die Informationen zur Verfügung, inwieweit der Scan im letzten Versuch erfolgreich war.

5.2 Big Brother

Im Folgenden wird die Implementierung des Scanners namens Big Brother beschrieben.

Für die Realisierung wurde die Programmiersprache Python gewählt, da diese bereits im Game Information System eingesetzt wird und zum anderen in der Lehre der Hochschule Niederrhein vorkommt. Eine Programmierung in beispielsweise C oder C++ wäre auch denkbar gewesen, wurde aber aufgrund der unterschiedlichen Syntaxen zwischen C und Python nicht gewählt.

Big Brother baut zu Beginn eine Verbindung mithilfe der in der Konfiguration (Listing 4) hinterlegten Informationen zur Datenbank auf. In der Datenbank befinden sich die Daten, die von Big Brother und dem GIS gemeinsam genutzt werden. Diese sind beispielsweise für Big Brother die auszuführende Aufgabe und die zu überwachenden GameClients und für das GIS die Ergebnisse der Scans.

Um die Verbindung zur PostgreSQL Datenbank aufzubauen, wird das Python-Paket *psycopg2* verwendet. Dieses dient als Adapter zwischen PostgreSQL Datenbanken und Python Programmen.

Nach dem erfolgreichen Aufbau der Verbindung zur Datenbank wird die Aufgabe des Scanners bestimmt. Sollte in der Datenbank nicht explizit die Aufgabe der Initialisierung spezifiziert werden, wird angenommen, dass die GameClients überwacht werden sollen. (siehe Listing 5.6)

```
cursor.execute(f"SELECT value FROM {SETTING_TABLE_NAME} WHERE
    ↪ key = 'scanner.task'")
task_tuple = cursor.fetchone()

try:
        task = task_tuple[0].upper()
except TypeError:
        task = "SCAN"
...
if task != "INIT":
        start_scan(connection)
else:
        run_init(connection)
```

Listing 5.6: Aufgabe des Scanners

5.2.1 Implementierung der Initialisierung

Bei der Initialisierung wird zuerst die bestehende Datenbanktabelle mit den vorhanden Services zurückgesetzt. Im Anschluss werden alle im Big Brother implementierten Services in die Datenbanktabelle geschrieben. Dazu wird ein Dummy-Scanner angelegt und die Scan-Operationen ausgelesen.

Durch diese Vorgehensweise können alte Datenbestände keine Fehler verursachen.

```
cursor.execute(f"DELETE FROM {SERVICE_TABLE_NAME}")
```

Listing 5.7: Löschen der Services

Bei dem Einfügen der Services werden pro Service der interne und der angezeigte Name sowie die Gewichtung und die Aktivierung in die Tabelle (*services*) geschrieben. Bei der Gewichtung und der Aktivierung werden Standardwerte verwendet, damit dieser Vorgang automatisch durchgeführt werden kann. Standardmäßig wird jeder Service mit 1 gewichtet und ist deaktiviert.

```
cursor.execute(f"INSERT INTO {SERVICE_TABLE_NAME}(name,
    ↪ public_name, weight, active) VALUES('{name}',
    ↪ '{public_name}', 1, FALSE);")
```

Listing 5.8: Einfügen eines Services

Die Standardwerte können pro Service über das GIS überschrieben werden.

Außerdem werden Standardwerte für Einstellungen in die Datenbanktabelle (*settings*) geschrieben, sofern diese nicht bereits vorhanden sind.

```
cursor.execute(f"INSERT INTO {SETTING_TABLE_NAME}(key, value)
    ↪ VALUES ('{default_value[0]}', '{default_value[1]}') ON
    ↪ CONFLICT (key) DO NOTHING;")
```

Listing 5.9: Einfügen einer Einstellung

5.2.2 Implementierung der Scan-Operations

Die im alten System vorhandene Überwachung ist in der Programmiersprache PHP implementiert und wurde im Rahmen dieser Arbeit in das Python-Programm überführt.

Die verschiedenen Aufgaben der Überwachung wurden, wie im Entwurf ausgearbeitet, in einzelne sogenannte Scan-Operationen getrennt. Dies soll das Verständnis fördern und den Aufwand einer Wartung reduzieren. Durch diesen Ansatz können einzelne Scan-Operationen einfacher ersetzt oder verändert werden. Außerdem ist es so möglich, leichter neue Scan-Operationen zu integrieren.

Basisklasse der Scan-Operationen

Alle implementierten Klassen werden von der Basisklasse *AbstractScanOperation* abgeleitet. Diese selbst wird von der in Python implementierten abstrakten Basisklasse *ABC* abgeleitet.

In der Basisklasse *AbstractScanOperation* wird die Funktion *_is_port_open* implementiert. Mithilfe dieser kann geprüft werden, ob ein bestimmter Port auf einem Rechner geöffnet ist. Hierzu wird mit der Bibliothek *socket* eine Socket-Verbindung zum Rechner gestartet. Schlägt die Verbindung fehl, wird angenommen, dass der Port auf dem Rechner geschlossen ist.

```
try:
        socket_connection = socket.socket(socket.AF_INET,
            ↪ socket.SOCK_STREAM)
        socket_connection.connect((self._ip, port))
        socket_connection.shutdown(socket.SHUT_RDWR)
        return True
except ...
        return False
finally:
        if socket_connection is not None:
                socket_connection.close()
```

Listing 5.10: Big Brother Funktion is_port_open

Die Funktion wird in der Basisklasse zur Verfügung gestellt, da verschiedene Scan-Operationen diese benötigen und nutzen.

Darüber hinaus definiert die Basisklasse die abstrakte Funktion *start*, die von allen abgeleiteten Scan-Operationen implementiert wird. Dadurch ist es möglich, die Scan-Operationen einheitlich zu starten.

Die Ergebnisse der Scan-Operationen werden ebenfalls einheitlich im privaten Attribut *_result* abgespeichert. Das Ergebnis kann über das getter-Property *result* abgerufen werden. Zum Abruf des internen und des Anzeigenamens werden ebenfalls getter-Properties eingesetzt.

Durch die Nutzung einer Art getter-Methode wird anderen Entwickelnden mitgeteilt, dass keine Änderungen an dem privaten Attribut durchgeführt werden sollen. Mithilfe des in Python vorhandenen Property Decorators kann die getter-Methode als Variable und nicht als Funktion verwendet werden.

```
1 @property
2 def result(self) -> bool:
3         return self._result
```

Listing 5.11: Big Brother Ergebnis Getter-Property

Host

In der HostUp-Operation wird geprüft, ob der entfernte Rechner erreichbar ist. Hierfür wird ein einfaches ICMP-Paket (Ping) gesendet. Wird eine Antwort auf das ICMP-Paket empfangen, ist der entfernte Rechner erreichbar.

Hierfür wird die Python Bibliothek *ping3* genutzt. Diese stellt ähnlich der *_is_port_open* eine Socketverbindung her. Die Besonderheit ist, dass statt einem TCP-Paket ein ICMP-Paket versendet wird. Da die Erstellung eines ICMP-Paketes nicht ohne Root-Rechte funktioniert, benötigt das Programm eben diese.

```
1 def start(self) -> None:
2 result = ping(self._ip, ttl=5,
    ↪ timeout=Config.operations['base']['ping_timeout'])
3 if result is None:
4         self._result = False
5 else:
6         self._result = True
```

Listing 5.12: Big Brother HostUp Ping

Für das Versenden eines ICMP-Paketes hätte auch die in Linux implementierte ping-Funktion per Syscall verwendet werden können. Diese Alternative wurde nicht gewählt, da das Programm unabhängig vom unterliegenden Betriebssystem agieren soll.

Bubble und Bubble-Ng

Für die beiden Bubble Scan-Operationen wurde eine extra Klasse geschaffen, die eine Methode bereitstellt, mit der die Erreichbarkeit des Bubble-Servers geprüft wird. Diese Klasse wurde geschaffen, da sich die beiden Scan-Operationen nur minimal voneinander unterscheiden.

Die Methode öffnet eine Telnet-Verbindung zum Bubble-Server und führt einen Befehl des Bubble-Servers aus. Sollte die Antwort das erwartete Ergebnis nicht beinhalten oder sollte keine Verbindung aufgebaut werden, wird der Bubble-Server als nicht erreichbar oder benutzbar angesehen.

```
1  def _is_bubble_port_up(self) -> bool:
2    try:
3      with Telnet(self._ip, self._port) as connection:
4        connection.write(b'help\n')
5          if connection.read_until(b"commands:",
               ↪ 1).decode('ascii') == "commands:":
6            return True
7          else:
8            return False
9          except ...
10           return False
```

Listing 5.13: Big Brother Buble Port Prüfung

Auch implementiert diese Klasse die *start*-Funktion, indem nur die *_is_bubble_port_up()*-Funktion aufgerufen und das Ergebnis zwischengespeichert wird.

```
1  def start(self) -> None:
2          self._result = self._is_bubble_port_up()
```

Listing 5.14: Big Brother Bubble Scan-Operation

Die eigentlichen Scan-Operationen *BubbleUp* und *BubbleNgUp* setzen in der Konfiguration der abstrakten Klasse jeweils den zu prüfenden Port. Derzeit läuft auf den GameClients auf Port *12321* der Bubble- und auf Port *12322* der BubbleNg-Server.

Web

Die Scan-Operationen *ScanHttpUpOperation* und *ScanHttpsUpOperation* unterscheiden sich nur im zu prüfenden Port. Beide nutzen zur Überprüfung des überwachten Dienstes die von der Basisklasse bereitgestellten Funktion *_is_port_open(PORT)*.

```
1  def start(self) -> None:
2          result = self._is_port_open(HTTPS_PORT)
3          ...
```

```
4           self._result = result
```

Listing 5.15: Big Brother HTTP(S) Scan-Operation

Die gegenwärtige Implementierung könnte ähnlich der Bubble Realisierung verändert werden, sodass für die HTTP-Dienste ebenfalls eine abstrakte Klasse definiert wird. Da im Gegensatz aber keine zusätzliche Funktion wie *_is_bubble_port_up()* benötigt wird, ist von der Realisierung einer allgemeinen HTTP-Klasse abgesehen worden.

FTP

In der FTP Scan-Operation soll gepürft werden, ob der FTP-Server nutzbar und ob der anonyme Login abgeschaltet ist. Dazu wird mit der Python Bibliothek *ftplib* eine Verbindung zum FTP-Server ohne Angabe von Nutzerdaten aufgebaut. Ist die Verbindung erfolgreich, wird der Dienst durch die Studierenden nicht abgesichert. Sollte die Verbindung fehlerhaft sein, wird angenommen, dass der Dienst nicht zur Verfügung steht.

Einzig eine Verbindung, die mit dem Fehler „anonymous access disabled“ beendet wird, zählt als erfolgreich, da hier die Verbindung an fehlenden Nutzerdaten gescheitert ist.

```
1  def start(self) -> None:
2    try:
3      with FTP(self._ip) as connection:
4        connection.login()
5        self._result = False
6      except error_perm as error_msg:
7        if "anonymous access disabled" in str(error_msg):
8          self._result = True
9        else:
10         self._result = False
11     except ...
12       self._result = False
```

Listing 5.16: Big Brother FTP Scan-Operation

SQL-Injection

Bei dieser Scan-Operation müssen zwei Dinge geprüft werden. Erstens muss sichergestellt werden, dass die Webseite mit der SQL-Injection erreichbar ist. Zweitens muss geprüft werden, ob die SQL-Injection behoben worden ist.

Die Erreichbarkeit muss geprüft werden, da ansonsten bei der Prüfung der SQL-Injection fehlerhafte Ergebnisse produziert werden, wenn die Seite nicht erreichbar ist.

Die Erreichbarkeit wird mithilfe einer validen Kombination aus Nutzername und Passwort geprüft. Sollte der Server nicht oder mit einer nicht erwarteten Nachricht antworten, wird die SQL-Injection als nicht nutzbar angesehen. Die Anfrage wird unter Zuhilfenahme der *request*-Bibliothek erstellt und gesendet.

```
def _is_sql_injection_up(self) -> bool:
 try:
  request_data = {
   "user":
     ↪ Config.operations['sql_injec']['admin']['username'],
   "pass": Config.operations['sql_injec']['admin']['password']
  }

  response = requests.post(f"http://{self._ip}/{URL_PATH}",
     ↪ data=request_data)

  if Config.operations['sql_injection']['control']['flag'] in
     ↪ response.text and
     ↪ Config.operations['sql_injection']['control']['value']
     ↪ in response.text:
   return True
  else:
   return False
 except ...
  return False
```

Listing 5.17: Big Brother SQL-Injection UP

Ob die SQL-Injection abgesichert ist, wird ähnlich wie bei der Nutzbarkeit geprüft. Es werden aber anstatt einer validen Kombination manipulierte Daten gesendet. Sollte nicht die implementierte Fehlermeldung „Login failed, you n00b!!!“ in der Antwort enthalten sein, ist die SQL-Injection nicht ausreichend abgesichert worden.

```
def start(self) -> None:
  if not self._is_sql_injection_up():
    self._result = False
  try:
    request_data = {
      "user": "' OR 1=1  #  ",
      "pass": "not used"
    }

    response = requests.post(f"http://{self._ip}/{URL_PATH}",
       ↪ data=request_data)

```

```
12      if response.status_code == 500:
13        self._result = False
14      elif "Login failed, you n00b!!!" in response.text:
15        self._result = True
16      else:
17        self._result = False
18    except ...
19      self._result = False
```

Listing 5.18: Big Brother SQL-Injection Save

XSS

Bei der XSS Scan-Operation wird geprüft, ob das Bewertungsformular auf den GameClients weiterhin für XSS-Angriffe offen ist.

Hierzu werden anstatt einer realen Bewertung der Name und die Bewertung mit JavaScript ersetzt und an den Server gesendet. Wird das gesendete JavaScript ungefiltert in das HTML-Dokument übernommen, wurde die XSS-Schwachstelle nicht behoben.

```
1  def start(self) -> None:
2    try:
3      request_data = {
4        "name": "<script>alert('XSS via name
           ↪ injected');</script>",
5        "bewertung": "<script>alert('XSS via bewertung
           ↪ injected');</script>"
6      }
7
8      response = requests.post(f"http://{self._ip}/{URL_PATH}",
         ↪ data=request_data)
9
10     if request_data['bewertung'] in response.text or
         ↪ request_data['name'] in response.text:
11       self._result = False
12     else:
13       self._result = True
14   except ...
15     self._result = False
```

Listing 5.19: Big Brother XSS Save

Telnet

Die Klasse der Telnet Scan-Operation implementiert eine Hilfsfunktion, die prüft, ob der Telnetport (*23*) erreichbar ist. Hierzu wird die von der Basisklasse implementierte Methode zum Prüfen offener Ports verwendet.

```
1 def _is_telnet_up(self) -> bool:
2         return self._is_port_open(23)
```

Listing 5.20: Big Brother Telnet

Die eigentliche Scan-Operation ruft nur die Hilfsfunktion auf und speichert das invertierte Ergebnis ab, da die Studierenden den Telnet-Server abschalten sollen.

```
1 def start(self) -> None:
2         result = self._is_telnet_up()
3         self._result = not result
```

Listing 5.21: Big Brother Telnet

Htaccess

Bei dieser Scan-Operation wird geprüft, ob die auf allen Systemen voreingestellten Nutzerdaten für den Htaccess-Schutz geändert worden sind. Zur Überprüfung wird die geschützte Seite mit den Standardnutzerdaten aufgerufen. Sollte der Statuscode ungleich 401 (Unauthorized) sein, waren die Nutzerdaten korrekt oder der Webserver war nicht erreichbar.

```
1  def start(self) -> None:
2    try:
3      response =
         ↪ requests.head(f"http://{self._ip}/phpmyadmin/",
         ↪ auth=(Config.operations['htaccess']['username'],
         ↪ Config.operations['htaccess']['password']))
4
5      if response.status_code == 401:
6        self._result = True
7      elif response.status_code ...
8        self._result = False
9    except ...
10     self._result = False
```

Listing 5.22: Big Brother Htaccess

SQL-Passwort

Im Gegensatz zum alten System wird das SQL-Passwort auf dem entfernten Rechner selbst geprüft. Dazu wird eine SSH-Verbindung mit den Nutzerdaten, die in der *settings*-Tabelle in der Datenbank hinterlegt sind, aufgebaut. Sollte die Verbindung fehlschlagen, wird angenommen, dass das SQL-Passwort nicht geändert worden ist.

Über die SSH Verbindung kann die lokale Datenbank angesprochen werden. Sollte die Authentifizierung mit der Standardkennung aufgrund falscher Nutzerdaten fehlschlagen, wurde das SQL-Passwort geändert.

```
1  def start(self) -> None:
2    connection = None
3
4    if self._ssh_username is None or self._ssh_username.upper()
       ↪ == "NONE" or self._ssh_password is None or
       ↪ self._ssh_password.upper() == "NONE":
5      self._result = False
6      return
7
8    try:
9      connection = paramiko.SSHClient()
10     ...
11     connection.connect(self._ip, username=self._ssh_username,
         ↪ password=self._ssh_password,
         ↪ timeout=Config.operations['base']['ssh_timeout'])
12
13     _, _, stderr = connection.exec_command(f"mysql -u
         ↪ {Config.operations['sql']['username']}
         ↪ -p{Config.operations['sql']['password']} -e 'quit'")
14
15     for line in stderr:
16       if "Access denied for user 'root'@'localhost' (using
           ↪ password: YES)" in line.strip('\n'):
17         self._result = True
18         return
19     self._result = False
20   except ...
21     self._result = False
22   finally:
23     if connection is not None:
24       connection.close()
```

Listing 5.23: Big Brother SQL-Passwort

5.2.3 Konfiguration

Konstanten

In der Datei *modules/helper/constant.py* sind Variablen abgelegt, die sich selten bis gar nicht ändern. Diese sind deshalb auch nicht in die Einstellungen übernommen worden.

Zu diesen Variablen zählen die Namen der benutzen Datenbanktabellen und Standardeinstellungen. Sollte beispielsweise in der Datenbank eine Tabelle umbenannt worden sein, kann diese Änderung in der Datei vollzogen werden. Durch diese Vorgehensweise muss die Änderung nicht manuell im nutzenden Code geschehen.

Konfiguration

Die Konfiguration in der Datei (*modules/config/config.py*) beinhaltet die Verbindungsdaten zur Datenbank, die URL des GIS – an die erfolgreiche Scans gemeldet werden sollen – sowie Einstellungen und Daten für die verschiedenen Scan-Operationen.

Die Standardkonfiguration ist sowohl im Anhang (Listing 4) als auch im Big Brother Repository vorhanden.

Einstellungen, die sich meiner Meinung nach häufiger ändern, sind in der Datenbanktabelle *settings* abgelegt, damit diese durch das Hochschulpersonal leichter verändert werden können.

5.2.4 Implementierung der Scan-Funktion

Die eigentliche Scan-Funktion wird mithilfe der Klasse *ScanGuard* implementiert, da für jede teilnehmende Gruppe ein Scanner gestartet wird. Bei der Erstellung eines Objektes dieser Klasse werden die Einstellungen, die teilnehmenden Gruppen und die aktiven Scan-Operationen übergeben.

Der ScanGuard legt für jede Gruppe ein Objekt der Klasse *Scanner* mit den benötigten Informationen wie IP, aktive Scan-Operationen und Einstellungen an.

Danach wird in einer Endlosschleife der nächste Scan-Zeitpunkt errechnet und im folgenden alle Scanner parallel mit der Bibliothek *concurrent* gestartet. Im Anschluss wird auf die Beendigung der Scanner sowie das Erreichen des nächsten Scan-Zeitpunktes gewartet.

```
1 while not self._stop_scanning:
2   sleep_until = datetime.now() +
      ↪ timedelta(seconds=self._timeout)
3
4   future_list = []
5   for scanner in scanner_list:
```

```
6       future_list.append(pool.submit(scanner.start))
7
8   wait(future_list)
9   ...
10  while not self._stop_scanning and datetime.now() <
        ↪ sleep_until:
11      sleep(500 / 1000)
```

Listing 5.24: Big Brother ScanGuard

Das Signal für die Beendigung des Programms wird abgefangen, damit die Endlosschleife abgebrochen, und so das Programm gracefull beendet wird.

5.2.5 Implementierung eines Scanners

Bei der Erzeugung eines Objektes der Klassse *Scanner* werden von allen benötigten Scan-Operationen Objekte angelegt.

Wie im Quellcode Listing 6 wird beim Starten eines Scanners anfangs geprüft, ob der entfernte Rechner erreichbar ist. Sollte der entfernte Rechner nicht erreichbar sein, werden keine weiteren Scan-Operationen durchgeführt. Im Anschluss wird noch sequentiell geprüft, ob der HTTP-Dienst erreichbar ist. Ist er erreichbar, werden die Scan-Operationen, die einen funktionierenden HTTP-Dienst voraussetzen, der Liste der durchzuführenden Scan-Operationen hinzugefügt.

Danach werden die restlichen Scan-Operationen parallel gestartet. Nachdem alle Scan-Operationen abgeschlossen sind, wird das Ergebnis ausgewertet und in der Datenbank festgehalten. Hierzu wird die Anzahl der durchgeführten Scans und bei Erfolg auch die Anzahl der erfolgreichen Scans um 1 inkrementiert. Außerdem werden der Zeitpunkt des Scans sowie der Erfolg oder Misserfolg vermerkt.

5.3 Game Information System

Das Game Information System wurde als REST-Schnittstelle implementiert. Sie kann über einen REST-Client wie cURL[1], Insomnia[2] oder einen eigenen Webclient angesprochen werden. Für die Realisierung wurde aufgrund der in Abschnitt 4.1 angeführten Argumente Flask als Framework gewählt.

[1] https://curl.haxx.se/
[2] https://insomnia.rest/

Zuerst wird die Flask-Anwendung mithilfe der Konfigurationsdatei eingestellt. Danach werden alle verwendeten Erweiterungen mit der Flask-Anwendung vertraut gemacht. Im Folgenden werden die Befehle des Command-line Interfaces angelegt. Zum Schluss werden die Routen registriert und nicht vorhandene Einstellungen mit Standardwerten in die *settings*-Tabelle der Datenbank geschrieben.

In der Produktionsumgebung wird zur Betreibung der Web Server Gateway Interface (WSGI) HTTP Server *gunicorn* verwendet, da der von Flask mitgebrachte WSGI Server nur für Entwicklungszwecke genutzt werden sollte. Ein WSGI Server wird benötigt, um Anfragen an eine in Python geschriebene Webanwendung weiterzuleiten.

5.3.1 Konfiguration

Damit das CTF-Spiel durch die betreuenden Personen angepasst werden kann, gibt es neben einer Konfigurationsdatei auch Einstellungen, die in der Datenbank gespeichert werden.

In der Konfigurationsdatei werden die Informationen gespeichert, die selten oder niemals geändert werden müssen. Die in der Datenbank gespeicherten Einstellungen können durch die betreuenden Personen über das REST-Interface geändert werden.

Konfigurationsdatei

In der Konfigurationsdatei sind mehrheitlich Einstellungen für die Flask Anwendung sowie die Erweiterungen definiert. Dazu zählen *Secret Keys*, aber auch die Verbindungsdaten zur PostgreSQL- und Redis-Datenbank. Die Secret Keys werden genutzt, um die gesendeten Informationen, beispielsweise JWTs, zu signieren und bei einer Antwort zu validieren.

```
1  FLAG_SECRET = "same_on_client"
2  FLAGSHOP_FLAG_SECRET = "another_secret"
3  SECRET_KEY = 'secret_key'
4  BEHIND_PROXY = True
5  PROXY_SECURE_VALUE = "http_header_pw"
6  JWT_SCRET_KEY = 'jwt_secret_key'
7  JWT_ACCESS_TOKEN_EXPIRES = datetime.timedelta(minutes=15)
8  JWT_REFRESH_TOKEN_EXPIRES = datetime.timedelta(hours=8)
9  REDIS_URL = "redis://localhost:6379"
10 SQLALCHEMY_DATABASE_URI =
     ↪ "postgresql://postgres:password@localhost:5432/postgres"
```

Listing 5.25: GIS Auszug aus der Konfiguration

Einstellungen

Über die Einstellungen lässt sich das Spiel leicht durch die betreuenden Personen verändern. Dazu gehört beispielsweise, ob der Login für die Studierenden mit validen Hackit-Zugangsdaten geschehen muss oder ob ein sogenannter anonymer Login erlaubt ist. Außerdem kann eingestellt werden, ob das Spiel am Ende der Angriffszeit automatisch beendet und aufgeräumt werden soll.

Nachdem ein Spiel beendet worden ist, ist es den Studierenden, anders als im derzeitig genutzten System, nicht mehr möglich weitere Aktionen durchzuführen. Für den Fall, dass dies nicht gewünscht ist, kann über die Einstellung das automatische Beenden deaktiviert werden.

Ist die Einstellung gesetzt, dass das Spiel automatisch aufgeräumt werden soll, werden alle Tabellen kurz nach der Beendigung zurückgesetzt. Tabellen, in denen Informationen hinterlegt sind, die über mehrere Spiele verwendet werden, bleiben unberührt.

```
1 setting.require_hackit: true
2 setting.end_game_automatically: true
3 setting.backup_game_automatically: true
4 setting.cleanup_game_automatically: true
5 setting.flag_submitting_penalty_points: 10
6 setting.flags_per_group: app.config['FLAG_COUNT']
7 game.discover: 0
8 game.attack: 0
```

Listing 5.26: GIS Auszug aus den Einstellungen

5.3.2 Object-Relational Mapping

Mit dem Object-Relational Mapping (ORM) können Daten aus der Datenbank mithilfe eines objektorientierten Paradigmas abgefragt oder manipuliert werden. Hierbei wird eine Verbindung zwischen einem Objekt in Python und einer Relation in der Datenbank geschaffen.

Über die Manipulation des Objekts lassen sich Änderungen am Datenbestand in der Datenbank durchführen.

Für das ORM wird das Python-Toolkit *SQLAlchemy* verwendet und durch die Flask Erweiterung *Flask-SQLAlchemy* für die Nutzung innerhalb der Flask-Anwendung konfiguriert.

Die Klassen werden von der durch *SQLAlchemy* bereitgestellten Klasse *Model* abgeleitet. Für jede zu erzeugende Spalte in der Datenbank wird im ORM-Modell ein Attribut vom Typ *Column*-Objekt erstellt. Das *Column*-Objekt wird ebenfalls von *SQLAlchemy* zur Verfügung gestellt und erhält Informationen wie den Datentyp und Constraints (Primary Key / Foreign Key). Das Vorgehen wird exemplarisch in Listing 5.27 dargestellt.

```
class Backup(db.Model):
  __tablename__ = 'backups'

  id = db.Column(db.Integer, primary_key=True,
    ↪ autoincrement=True)
  created_at = db.Column(db.DateTime(timezone=True),
    ↪ server_default=func.now(), nullable=False)
  data = db.Column(db.JSON, nullable=False)
```

Listing 5.27: GIS Beispiel eines ORM Models

Auch implementieren die ORM-Modelle eine Klassenmethode, über die Objekte anhand von bestimmten Informationen aus der Datenbank abgerufen werden können. Alternativ könnte anstatt der Verwendung der *get*-Methode auch die *query*-Methode aus der Basisklasse *Model* verwendet werden. Dies wurde aus Konsistenzgründen und der häufigen Nutzung im Quellcode nicht verwendet.

```
class Backup(db.Model):
@classmethod
def get(cls, id: int = None, select_all: bool = False):
  if id is not None:
    return cls.query.get(id)
  elif select_all:
    return cls.query.all()
  else:
    return None
```

Listing 5.28: GIS Beispiel einer Get-Methode des ORM Models

Hash des Account Passwortes

Das Accountmodell speichert das Passwort im privaten Attribute *_password*. In der Datenbank ist das Passwort aber in der Spalte *password* abgelegt. Über die *setter*-Methode kann das Passwort im Programmcode manipuliert werden.

Durch die Nutzung von *getter*- und *setter*-Methoden wird sichergestellt, dass alle Passwörter auf die gleiche Art und Weise gehasht werden und niemals im Klartext vorliegen.

Für das Passworthashing wird der Algorithmus *bcrypt* verwendet, da dieser sequentiell abgearbeitet wird und daher langsamer als beispielsweise *sha* berechnet wird. Für die Erstellung eines Hashwertes wird zwingend ein Salt benötigt. Aus IT-Sicherheitsgründen ist ein Salt unverzichtbar und eine gute Methode Rainbow Tables vorzubeugen. Außerdem ist *bcrypt* zukunftssicher, da der Algorithmus durch die Anzahl der verwendeten Rounds langsamer gemacht werden kann. Je mehr Rounds verwendet werden, desto länger dauert die Erzeugung

eines Passwort-Hash. Diese Eigenschaften sollen einen Brute-Force-Angriff verlangsamen, beziehungsweise teurer machen. [Ari18]

Die bei der Erstellung verwendeten Rounds des Bcrypt-Passworthash mussten von 15 auf 10 gesetzt werden. Ein testweise durchgeführter Import einer bestehenden Hackit-Zugangsdaten-Datei hat gezeigt, dass bei einer Nutzung von 15 Rounds der Import mehr als 5 Minuten gedauert hat.

```
1  class User(db.Model):
2  ...
3  _password = db.Column('password', db.String, nullable=False)
4  ...
5  def get_password(self) -> str:
6  return self._password
7
8  def set_password(self, password):
9  custom_bcrypt = bcrypt.using(rounds=10)
10 self._password = custom_bcrypt.hash(password)
11
12 password = db.synonym('_password',
       ↪ descriptor=property(get_password, set_password))
13
14 def check_password(self, password: str) -> bool:
15 return bcrypt.verify(password, self.password)
```

Listing 5.29: GIS Passwort des Accounts

Flaggenerierung

Für die Flaggenerierung muss derselbe Algorithmus und Seed wie auf dem GameClient verwendet werden, damit dieselben Flags auf dem Server und dem Client generiert werden. Dazu wird ein Seed mithilfe des Hash-Algorithmus *MD5* gehasht. Näheres zum Seed ist in der Flaggenerierung in Unterabschnitt 5.3.5 beschrieben.

Damit alle Flags als MD5-Hash in der Datenbank abgelegt werden, werden auch hier eine *getter*- und eine *setter*-Methode sowie ein privates Attribut genutzt. Die *setter*-Methode erzeugt einen md5-Hash und speichert diesen im privaten Attribut ab. Sollte die Hashfunktion geändert werden, kann diese wie im Kommentar in Listing 5.30 erkennbar ausgetauscht werden.

```
1  class Flag(db.Model):
2    ...
3    _value = db.Column('flag_value', db.String,
       ↪ primary_key=True)
4    ...
5    def get_value(self) -> str:
```

```
6       return self._value
7
8     def set_value(self, value: str) -> None:
9       # self._value = sha512(value.encode('utf-8')).hexdigest()
10      self._value = md5(value.encode('utf-8')).hexdigest()
11
12    value = property(get_value, set_value)
```

Listing 5.30: GIS Hashen der Flags

5.3.3 Migrationen

Die Datenbankmigrationen werden aus den vorhandenen ORM-Modellen automatisch mit dem Datenbank-Migrationswerkzeug *Alembic* und der Erweiterung *Flask-Migrate* erzeugt. Einzig für die in Abschnitt 5.1 eingeführten Views mussten manuell eigene Migrationsskripts angelegt werden. Das Vorgehen kann im Quellcode nachvollzogen werden.

Ein solches Skript kann nach Änderungen an einem oder mehreren ORM Models durch den in 5.31 gezeigten Befehl erzeugt werden.

```
1 flask db migrate
```

Listing 5.31: GIS Erzeugung eines Migrationsskripts

Mit den Skripts ist es möglich, die Datenbank auf ältere Iterationen zurückzusetzen, da bei Erstellung eines Iterationsschrittes auch der Rückweg festgehalten wird.

Um die vorliegenden Migrationsskripts auf die Datenbank anzuwenden, werden die in Listing 5.32 gezeigten Befehle genutzt.

```
1 flask db upgrade
2 flask db downgrade <Version>
```

Listing 5.32: GIS Nutzung eines Migrationsskripts

Die realisierten Tabellen und Views wurden in Abschnitt 5.1 erläutert und werden deshalb hier nicht weiter aufgegriffen.

5.3.4 Authentifizierung

Für die Authentifizierung werden, wie im Entwurf angedacht, Tokens, spezieller JSON Web Tokens (JWT), verwendet. Es werden ein Access- und ein Refresh-Token nach einem erfolgreichen Login für die Nutzenden erzeugt. Diese Tokens werden in der Redis-Datenbank für die Dauer ihrer Gültigkeit abgespeichert, um sie bei Bedarf invalidieren zu können. Bei der

Authentifizierung wird zuerst geprüft, ob der vom Client gesendete Token mit dem richtigen Secret Key signiert ist. Danach wird geprüft, ob der Token als gültig in der Redis-Datenbank hinterlegt ist und innerhalb seiner Lebensspanne verwendet wird.

Der Refresh-Token hat eine Gültigkeit von 8 Stunden und kann verwendet werden, um weitere Access-Tokens ohne Angabe der Nutzerdaten zu erstellen. Die Lebensdauer des Refresh-Tokens wurde auf 8 Stunden gesetzt, da es vorkommen kann, dass der Versuch zweimal hintereinander durchgeführt wird. Das betreuende Personal muss sich dann nicht erneut anmelden.

Der Access-Token besitzt eine Gültigkeit von 15 Minuten und wird zur Authentifizierung und Autorisierung verwendet. Die Gültigkeit ist auf 15 Minuten begrenzt, damit bei einem Verlust des Tokens, ein Angreifer maximal 15 Minuten Schaden anrichten kann. Um Anfragen an die SQL-Datenbank zu minimieren, werden Informationen wie Rolle oder Gruppenzugehörigkeit eines Accounts mit in den Access-Token übernommen. Änderungen an der Rolle eines Accounts werden so aber erst nach einem erneuten Ausstellen des Access-Tokens gültig.

Decorator für Routen

Um die Routen mit Berechtigungen zu versehen und redundanten Code zu verhindern, wird bei der Absicherung auf Decorator zurückgegriffen. Sollte eine Route mit einem Decorator abgesichert werden, wird vor dem Aufruf der implementierten Funktionalität geprüft, ob der JWT die benötigten Berechtigungen besitzt.

In Tabelle 2/3 sind die Routen mit ihren benötigten Berechtigungen dargestellt.

Es werden zwei Decorator zur Verfügung gestellt. Beide prüfen, ob der Token im Feld *role* die benötigte Rolle gesetzt hat. Der Unterschied zwischen *role_required* und *roles_required* ist, dass bei *roles_required* eine Liste von erlaubten Rollen angegeben werden kann. Dies wird auch im Code deutlich, da anstatt eines Vergleiches von Variablen geprüft wird, ob die Rolle, die im JWT codiert ist, in der Liste der erlaubten Rollen existiert.

```
1  def roles_required(roles: list):
2    def _roles_required(fn):
3      @wraps(fn)
4      def wrapper(*args, **kwargs):
5      ...
6      if get_jwt_claims()['role'] not in roles:
7        raise Forbidden(f"You␣don't␣have␣any␣role␣of␣{roles}")
8      else:
9        return fn(*args, **kwargs)
10   return wrapper
11 return _roles_required
```

Listing 5.33: GIS Prüfung der Rollen

Der Decorator *flagshop_route* wird für die Routen verwendet, die einen gültigen Flagshop-User voraussetzen. Er nutzt den implementierten *role_required* Decorator mit der Flagshop-User Rolle.

5.3.5 Routen

Im Folgenden werden die implementierten Routen aufgegriffen. Eine Übersicht mit den implementierten Routen und den dazugehörigen Methoden befindet sich im Anhang in Tabelle 1.

Die Erweiterung *Flask-RESTful* wird verwendet, um die verschiedenen Methoden eines Endpoints einer Klasse zuzuordnen. Die Klasse wird von der Klasse *Resource* abgeleitet. Andernfalls wäre die API aus vielen verschiedenen Funktionen entstanden.

```
1 class EndpointName(Resource):
2   def get(self):
3     ...
4   def post(self):
5     ...
6   def put(self):
7     ...
8   def delete(self):
9     ...
```

Listing 5.34: GIS Beispiel eines Endpoints

Außerdem werden auftretende Fehler im JSON-Format und nicht als Plain-Text versendet.

Wie bereits im Entwurf erwähnt, erhalten alle Routen für die Versionierung den Präfix */v1*.

Index

Der Endpoint **/** stellt über die *GET*-Methode Informationen zur API im JSON-Format bereit. Denkbar, aber nicht implementiert ist eine Liste der verfügbaren Endpoints und deren Methoden.

Um die Gültigkeit von Access-Tokens zu prüfen, kann der Endpoint */secure* mit der *GET*-Methode aufgerufen werden. Sollte der Token valide sein, wird eine kurze Erfolgsnachricht, andernfalls eine Fehlernachricht, zurückgegeben.

Login

Für die Authentifizierung werden mehrere Endpoints benötigt.

Der Endpoint */auth/login* stellt nach einem erfolgreichem Login, mit einer validen Kombination aus Nutzernamen und Passwort, den Access- und der Refresh-Token aus. Dazu müssen die Nutzerdaten mit der *POST*-Methode an den Server übermittelt werden. Die Tokens werden dann im JSON-Format zurück gesendet.

```
1 {
2   'access_token': Base64-Access-Token,
3   'refresh_token': Base64-Refresh-Token
4 }
```

Listing 5.35: GIS Access- und Refresh-Token

Die Gültigkeit kann – wie bereits in Unterabschnitt 5.3.4 erwähnt – in der Konfiguration geändert werden und ist derzeitig auf 8 Stunden für den Refresh- und auf 15 Minuten für den Access-Token begrenzt.

Über den Endpoint */auth/refresh* kann ein valider Refresh-Token genutzt werden, um ohne Angabe der Nutzerdaten einen neuen Access-Token zu erhalten.

Da die Access- und Refresh-Tokens anders validiert werden, müssen für das Zurückrufen der Tokens zwei Endpoints zur Verfügung stehen. Access-Tokens können an den Endpoint */auth/revoke/access* und Refresh-Tokens an den Endpoint */auth/revoke/refresh* gesendet werden. Sollten die angekommenen Tokens gültig sein, werden sie dann in der Redis-Datenbank als „zurückgerufen“ markiert.

Da der Access-Token für den Flagshop anders als der normale Access-Token aufgebaut ist, wird der Enpoint */auth/flagshop* benötigt. Dieser ist nur mit einem gültigen normalen Access-Token nutzbar. Für den Flagshop wird nur ein Access-Token mit einer Gültigkeit von 15 Minuten ausgestellt.

Ein Login ist für eine spielende Person nur dann möglich, wenn die eigene IP-Adresse in der Liste der Gruppen oder deren Mitglieder vorhanden ist. Andernfalls scheitert der Login mit einer entsprechenden Fehlermeldung. Dies ist notwendig, damit das System bestimmen kann, für welche Gruppe eine Aktion ausgeführt worden ist.

Account

Für die Verwaltung der Accounts werden zwei Endpoints bereitgestellt.

Der Endpoint */user* stellt über die Methode *GET* für die betreuenden und administrierenden Personen eine Liste der im System vorliegenden Login-Informationen bereit. Über die Methode *POST* können besagte Personen neue Accounts anlegen. Accounts mit den Rollen *supervisor* und *admin* können nur durch administrierende Personen angelegt werden. Die Methode

DELETE kann genutzt werden, um alle Accounts der Rolle *player* gleichzeitig zu löschen. Theoretisch könnten mit dieser Methode auch Accounts anderer Rollen gelöscht werden. Die Methode wurde limitiert, um ein unabsichtliches Löschen der Accounts des Hochschulpersonals zu verhindern.

```
1 if args['role'] != 'player':
2   raise Conflict('Currently␣only␣mass␣deletions␣of␣players␣
      ↪ are␣supported.')
```

Listing 5.36: GIS Löschen auf player-Accounts begrenzen

Die Funktion wurde geschaffen, um die Hackit-Zugangsdaten nach Ende des Semesters aus der Datenbank zu entfernen.

Über die Methoden *GET*, *PUT* und *DELETE* des Endpoints */user/<int:user_id>* wird eine Verwaltung einzelner Accounts ermöglicht. Hierzu muss bei der Anfrage die Account-ID übergeben werden. Die Methode *GET* zeigt alle Informationen zu dem bestimmten Account an. Über die Methode *PUT* können einzelne Informationen verändert und über die Methode *DELETE* kann der Account gelöscht werden.

Die betreuenden Personen dürfen auch hier nur Accounts der Rolle *player* sowie ihre eigenen verwalten. Die eigene Rolle kann nur auf die Rolle *player* geändert werden. Dies soll unerlaubte Rechteausweitung verhindern. Auch ist es nicht möglich, den aktuellen Account zu löschen. Sollte die Löschung eines Accounts nötigt sein, muss dafür ein anderer Account genutzt werden.

Außerdem ist ein dritter Endpoint, erreichbar unter */user/import*, implementiert, der das Importieren von Hackit-Zugangsdaten ermöglicht. Hierfür wird eine Datei per *POST*-Anfrage an den Server gesendet. Die Login-Informationen werden zeilenweise ausgelesen und mit der Rolle *Player* angelegt. Ein Import von beispielsweise Accounts der Rolle *admin* kann auf ähnliche Art realisiert werden, wurde jedoch nicht implementiert, da dies keine Anforderung an die Software war.

Gruppen

Für die Verwaltung der teilnehmenden Gruppen werden zwei Endpoints zur Verfügung gestellt.

Der Endpoint */client* ermöglicht das Ansehen und Anlegen von teilnehmenden Gruppen. Die Liste der Gruppen wird mit der *GET*-Methode ausgeliefert und beinhaltet ID und IP der Gruppen. Über die *POST*-Methode können neue Gruppen angelegt werden. Neben administrierenden und betreuenden Personen können sich GameClients selbst registrieren. Die Registrierung ist für die GameClients nur vor einem Spiel möglich. Damit nur GameClients sich selber registrieren können, besitzen GIS und die GameClients ein Secret, über das validiert wird, ob die Anfrage von einem GameClient kommt.

Bei der Registrierung einer Gruppe werden die dazugehörigen Flags über den in Abschnitt 2.4.1 vorgestellten Algorithmus mit den in Abschnitt 3.8 erwähnten Veränderungen generiert. Das Hashen erfolgt – wie bereits in Abschnitt 5.3.2 erwähnt – über die *setter*-Methode des *value*-Attributes.

```
1  flag_secret = current_app.config['FLAG_SECRET']
2  flagshop_secret = current_app.config['FLAGSHOP_FLAG_SECRET']
3
4  # Flags on client
5  seed = f"{group.token}{group_ip}{flag_secret}"
6  new_flag = Flag(value=f"{seed}{counter}", usage='on_client',
       ↪ group_id=group.id)
7
8  # Flagshop Flags
9  seed = f"{group.token}{group_ip}{flagshop_secret}flag_reg"
10 new_flag = Flag(value=f"{seed}{counter}",
       ↪ usage='flagshop_registration', group_id=group.id)
11
12 seed = f"{group.token}{group_ip}{flagshop_secret}flag_buy"
13 new_flag = Flag(value=f"{seed}{counter}",
       ↪ usage='flagshop_buy', group_id=group.id)
```

Listing 5.37: GIS Flaggenerierung

Über den Endpoint */client/<int:group_id>* können die administrierenden und betreuenden Personen eine teilnehmende Gruppe verwalten. Neben dem Anzeigen der Gruppeninformationen inklusive der Gruppenmitglieder ist es ihnen möglich, die Gruppe mit allen verbundenen Informationen zu löschen.

Gruppenmitglieder

Die Verwaltung der Gruppenmitglieder steht den administrierenden und betreuenden Personen sowie den über die IP-Adresse der Gruppe angemeldeten Spielenden zur Verfügung.

Der Endpoint */associate* ermöglicht über die *POST*-Methode das Anlegen neuer Gruppenmitglieder. Die administrierenden und betreuenden Personen können weiterhin die *GET*-Methode für die Auflistung aller Gruppenmitglieder, inklusive der zugehörigen Gruppe, nutzen.

Über den Endpoint */associate/<int:associate_id>* können einzelne Gruppenmitglieder bei Bedarf gelöscht werden.

Einstellungen

Über den Endpoint */setting* ist es den betreuenden und administrierenden Personen möglich, die im Spiel genutzten Einstellungen zu ändern. Dazu können alle Einstellungen (siehe Abschnitt 5.3.1) über die *GET*-Methode angefragt und mithilfe der *PUT*-Methode verändert werden.

Flag

Über den Endpoint */flag* können die Studierenden die gefundenen und erhaltenen Flags abgeben. Für die Punkteverrechnung wird die im Access-Token codierte Gruppe benötigt. Eine Abgabe von Flags wird verhindert, falls das Spiel nicht aktiv ist (noch nicht gestartet, beendet oder pausiert).

Die Validierung der Flag wird mithilfe der *flags*-Tabelle durchgeführt, indem geprüft wird ob die abgegebene Flag in dieser Tabelle vorhanden ist.

Im Anschluss wird geprüft, ob es sich bei der abgegebenen Flag um eine Flag handelt, die nicht auf den GameClients verteilt ist und durch eine andere Gruppe als der Besitzenden abgegeben wird. Dies verhindert die Abgabe von bspw. Flagshop-Flags durch andere Gruppen.

```
1 if flag.group_id != group_id and flag.usage != "on_client":
2   raise BadRequest("Private␣flags␣cannot␣be␣submitted")
```

Listing 5.38: GIS Abgabe privater Flags verhindern

Sollten fremde Flags während der Discover-Zeit abgegeben werden, bestraft das System die abgebende Gruppe für ihr regelverletzendes Verhalten automatisch. Die Flag muss von der Gruppe nach Start der Attack-Time erneut abgegeben werden, um Punkte für diese zu erhalten.

```
1 if game_status_setting.value == "discover" and flag.group_id
    ↪ != group_id:
2   penalty = Penalty(reason="Submitted␣an␣foreign␣flag␣during␣
      ↪ discovery", points=penalty_points,
3   group_id=group.id)
4   raise Conflict("You␣submitted␣an␣alien␣flag␣during␣
      ↪ discovery.␣The␣flag␣is␣not␣accepted.␣You'll␣be␣
      ↪ punished.")
```

Listing 5.39: GIS Strafe für das Abgeben fremder Flags

Sollten alle Überprüfungen erfolgreich durchgeführt worden sein, wird die Flag inklusive der abgebenden Gruppe in die *flags_submitted*-Tabelle aufgenommen. Die Berechnung der Punkte erfolgt wie in Abschnitt 5.1 beschrieben über die Views.

Der Endpoint ist mit einem Rate Limiter versehen, der eine Brute-Force-Abgabe von Flags verhindern soll. Dies soll die Abgabe von durch Studierende generierte Flags verhindern oder bestrafen. Pro Gruppe werden in der Minute nicht mehr als 60 Flags akzeptiert. Der Wert kann nach Bedarf verändert werden.

Spielsteuerung

Der Endpoint */match/control* ermöglicht es den administrierenden und betreuenden Personen, das Spiel zu verwalten. Mit der Methode *POST* kann das Spiel angelegt werden. Die Methode *PUT* ermöglicht das Verwalten des Spiels. Über die *DELETE*-Methode kann das aktuell laufende Spiel beendet werden.

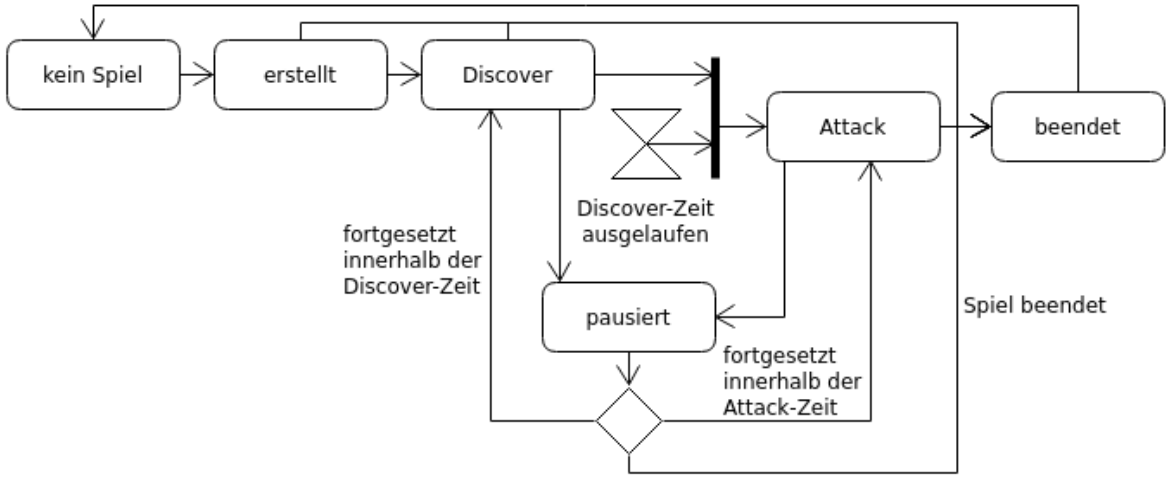

Abbildung 5.3: Spielstatus (Zustandsdiagramm)

In Abbildung 5.3 sind mögliche Übergänge zwischen den Spielstatus zu sehen. Ein Spiel muss angelegt werden, damit GameClients eine Teilnahme am Spiel bekunden können. Wenn sich alle GameClients registriert haben, kann das Spiel gestartet werden. Es beginnt mit der Discover-Zeit und wechselt nach Überschreitung dieser automatisch in die Attack-Zeit.

Während der Discover- und Attack-Phase kann das Spiel pausiert werden. Ein angehaltenes Spiel wird in der Phase fortgesetzt, in der es pausiert wurde. Die weiterzuführende Phase wird durch den Vergleich zwischen Start der Pause und Ende der Phasen bestimmt.

```
1 if pause_start < discover_end:
2   ...
3 elif pause_start < attack_end:
4   ...
```

Listing 5.40: GIS Spiel fortsetzen

Die neuen Endzeitpunkte der Phasen werden berechnet, indem der derzeitige Endzeitpunkt mit der Differenz von Start und Beendigung der Pause addiert wird. Wird das Spiel in der Attack-Phase fortgesetzt, bleibt das Ende der Discover-Zeit unberührt.

```
1 time_delta = datetime.now() - pause_start
2 new_discover_end = discover_end + time_delta
3 new_attack_end = attack_end + time_delta
```

Listing 5.41: GIS Neues Ende berechnen

Spiele, die sich im Status *erstellt*, *discover*, *attack* oder *pausiert* befinden, können beendet werden. Ist ein Spiel beendet, können die Studierenden keine weiteren Aktionen am GIS durchführen.

Nach dem Beenden kann das Spiel bereinigt werden. Hierbei werden alle nicht länger benötigten Daten aus der Datenbank gelöscht. Zu diesen Daten gehören beispielsweise die teilnehmenden GameClients, inklusive der dazugehörigen Informationen wie etwa Gruppenmitglieder, Serviceinformationen und abgegebene Flags. Angelegte Notizen werden nicht gelöscht, da diese über mehrere Versuchstermine Relevanz haben könnten.

```
1 scheduler.add_job(id="change_game_status_attack",
    ↪ func=change_gamestatus, trigger='date',
    ↪ run_date=discover_end_datetime)
```

Listing 5.42: GIS Scheduler Jobs

Die automatische Spielsteuerung sowie die Statusänderung des Spiels nach Ende der Discover-Zeit werden mit der Erweiterung *Flask APScheduler* umgesetzt. Dazu werden Jobs verwendet, die zu einer angegebenen Zeit eine bestimmte Funktion ausführen.

Scanner-Steuerung

Der Scanner wird durch die Spielsteuerung gesteuert. Jedoch erhalten auch die administrierenden und betreuenden Personen die Möglichkeit, diesen über den Endpoint */scanner* zu bedienen. Er kann über die *POST*-Methode gestartet und über die *DELETE*-Methode beendet werden. Die Methode *GET* stellt die Logs des Containers bereit.

Das GIS steuert den Big Brother Container über den Docker-Deamon an. Deshalb muss der Scanner beim Starten des Stacks mit hochgefahren und im Anschluss direkt abgeschaltet werden. Die Steuerung wird unter Zuhilfenahme der Python Bibliothek *docker* ermöglicht. Damit der Container gesteuert werden kann, wird der Container-Name benötigt. Dieser sowie der aktuelle Status des Containers werden in der *settings*-Tabelle abgelegt.

Spielstände

Der aktuelle Spielstand kann über den Endpoint */match/score* abgerufen werden. Dazu werden die Informationen aus der Datenbank aufbereitet. Neben den Gruppenpunkten aus der *total_points*-View werden auch die aktiven Services, inklusive ihren Namen, sowie die Überschriften für die Punktetabelle übermittelt.

Über den Endpoint */match/info* können die aktuellen Spielinformationen abgerufen werden. Zu den Spielinformationen zählt der aktuelle Status des Spiels und des Scanners sowie der Endzeitpunkt der Discover- und Attack-Zeit.

Alte Spielstände können über die Endpoints */backup* und */backup/<int:backup_id>* abgerufen werden. Der Endpoint */backup* gibt nur eine Liste der vorhandenen Spielstände inklusive des Zeitpunktes der Speicherung zurück. Mithilfe der Liste kann eine Spielstand-ID ausgesucht und der Spielstand über den Endpoint */backup/<int:backup_id>* abgerufen werden.

Bei alten Spielständen werden die Daten im JSON-Format in der Variablen *data* übermittelt.

Strafe

Über den Endpoint */penalty* können alle ausgesprochenen Strafen über die *GET*-Methode angesehen werden. Administrierende und betreuende Personen können über die *POST*-Methode neue Strafen erstellen. Hierzu sind Angaben zu Grund, Anzahl der Strafpunkte sowie die zu bestrafende Gruppe notwendig.

Der Endpoint */penalty/<int:penalty_id>* kann von den administrierenden und betreuenden Personen genutzt werden, um einzelne Strafen anzusehen (*GET*), zu bearbeiten (*PUT*) und zu löschen (*DELETE*).

Flagshop

Der Flagshop setzt verschiedene Funktionen um und besteht aus drei getrennten Bereichen. Die Bereiche Accountverwaltung und Paketverwaltung implementieren jeweils zwei Endpoints. Zusätzlich existiert noch einen Endpoint für die Transaktionsverwaltung.

Der Endpoint */flagshop/user* stellt die Methoden *GET* und *POST* bereit. Über die *POST*-Methode können administrierende, betreuende und spielende Personen neue Flagshop-User anlegen. Die *GET*-Methode ist für Spielende nicht nutzbar und listet alle angelegten Flagshop-User, inklusive der zugehörigen Gruppe, auf.

Die administrierenden und betreuenden Personen können über den Endpoint */flagshop/user/<user_name>* angelegte Flagshop-User verändern oder löschen.

Damit Flagshop-User Pakete kaufen können, müssen die Pakete zuerst durch betreuende oder administrierende Personen angelegt werden. Hierfür stellt der Endpoint */flagshop/package* eine *POST*-Methode zur Verfügung. Auch kann die Liste der verfügbaren Pakete mittels einer *GET*-Methode von angemeldeten Flagshop-Usern sowie den oben genannten Personengruppen abgerufen werden.

Sollten Pakete geändert oder gelöscht werden, muss der Endpoint */flagshop/package/<int:package_id>* verwendet werden. Die Methode *PUT* ermöglicht das Verändern und die Methode *DELETE* das Löschen.

Damit die Flagshop-User eine Transaktion durchführen können, wird der Endpoint */flagshop/transaction* benötigt. Über die *POST*-Methode kann eine Bestellung aufgegeben werden. Der Preis der Bestellung wird aus den in Abschnitt 2.4.1 erwähnten Gründen in der Anfrage mit übermittelt und nicht auf dem Server anhand des Warenkorbs berechnet.

Über diesen Endpoint können sich administrierende und betreuende Personen außerdem alle Transaktionen anzeigen lassen und diese bei Bedarf löschen.

Challenge

Über den Endpoint */challenge* können angemeldete Personen alle verfügbaren Challenges mit der *GET*-Methode einsehen. Es werden hierbei nur der Name, die ID und der Wert der Challenge angezeigt. Über die *POST*-Methode wird den administrierenden und betreuenden Personen die Möglichkeit gegeben, neue Challenges anzulegen.

Der Endpoint */challenge/<int:challenge_id>* stellt die Methoden *GET*, *PUT* und *DELETE* zur Verfügung. Mithilfe der *GET*-Methode können die Informationen der Challenge abgerufen werden. Sollte eine spielende Gruppe dies machen, wird der Start der Challenge in der entsprechenden Datenbanktabelle festgehalten.

Eine Challenge kann über die *PUT*-Methode verändert und über die *DELETE*-Methode gelöscht werden.

Für die Abgabe der Challenges werden weitere Endpoints benötigt. So stellt der Endpoint */challenge/solve/<int:challenge_id>* über die *POST*-Methode die Möglichkeit der Abgabe von Challenges für die Studierenden zur Verfügung.

Administrierende und betreuende Personen können über die *DELETE*-Methode einen Start oder eine Abgabe einer Challenge zurücknehmen. Außerdem können sie über den Endpoint */challenge/solve* alle gestarteten und beendeten Challenges nachvollziehen.

Notizen

Über die *POST*-Methode des Endpoints */note* sind administrierende oder betreuende Personen in der Lage, neue Notizen zu erstellen. Notizen können über den Endpoint */note/<int:note_id>* verändert (*PUT*) oder gelöscht (*DELETE*) werden.

Die hinterlegten Notizen können als Liste über die *GET*-Methode des Endpoints */note* oder einzeln über die *GET*-Methode des Endpoints */note/<int:note_id>* abgerufen werden.

Log

Um die Aktionen des Scanners nachvollziehen zu können, werden Log-Einträge abgespeichert. Diese können durch betreuende und administrierende Personen über den Endpoint */log* eingesehen werden. Außerdem ist es diesen Personen möglich, eigene Log-Einträge mithilfe der *POST*-Methode zu erstellen.

Log Informationen des letzten Spiels können über den Endpoint */log/old* abgerufen werden.

5.3.6 CLI Befehle

Die Erweiterungen *Flask-Migrate* und *Flask-RESTful* stellen CLI-Befehle zur Verfügung.

Flask-Migrate stellt die in Unterabschnitt 5.3.3 vorgestellten Befehle bereit. Über den Befehl *flask routes* zeigt *Flask-RESTful* alle registrierten Ressourcen mit den entsprechenden Methoden und Routen an.

Außerdem ermöglicht es ein selbst programmierter Befehl, ohne Angabe von Login-Informationen, Accounts zu verwalten. Dies wird zum einen benötigt, um den ersten Benutzer anzulegen. Zum Anderen kann bei einem Verlust aller Adminkennungen eine bestehende Kennung geändert oder eine neue angelegt werden.

Werden bei einem Befehl nicht alle benötigten Informationen angegeben, erfragt die Anwendung diese über die Befehlszeile.

```
1 flask user create username --role admin/supervisor/player
    ↪ --password secure_pw
2 flask user delete username
3 flask user create username --role admin/supervisor/player
    ↪ --password secure_pw
```

Listing 5.43: GIS CLI

6 Fazit

6.1 Zusammenfassung

Diese Arbeit hat sich mit dem Entwurf und der Realisierung eines Capture the Flag Core Systems beschäftigt. Ziel war es, ein System zu erschaffen, das im Rahmen des zweiten Praktikumsversuchs der Lehrveranstaltung IT-Sicherheit an der Hochschule Niederrhein eingesetzt werden kann. Es soll das bestehende Überwachungs- und Auswertungssystem ersetzen. Das CTF Core System soll die gleichen Basisfunktionalitäten wie das vorherige System besitzen und ein Spiel für mehr als acht Gruppen ermöglichen.

Durch die Lehrveranstaltung und die Ausstattung des EZS-Labors, in dem der Versuch stattfindet, wird ein Rahmen für das zu konzipierende System gegeben. Die funktionalen Anforderungen, die mindestens implementiert werden sollten, wurden vom bestehenden System vorgegeben.

Zu diesen gehörten:

- Scan der GameClients
- Flaggenerierung zur Prüfung der Flags
- Flagabgabe inklusive Verrechnung
- Verwaltung des Spiels
- Ansicht des Spielstatus
- Flagshop
- Challegnes.

Die Scans der GameClients wurden übernommen und in Scan-Operationen untergliedert. Jede Scan-Operation prüft genau eine Schwachstelle oder einen Dienst.

Die Scan-Operationen bestehen auszugsweise aus den Prüfungen von:

- GameClient erreichbar
- HTTP-Dienst erreichbar
- Bubble(-NG)-Server beantwortet definierten Befehl
- XSS im Bewertungsformular behoben

- Login nutzbar und SQL-Injection behoben
- Anonymer Login des FTP-Servers abgeschaltet
- Telnet deaktiviert.

Die einzelnen Scan-Operationen werden parallel von einem Scanner abgearbeitet. Pro Gruppe wird ein Scanner zur Überwachung parallel gestartet. Nach einer Scan-Runde wird eine durch die betreuenden Personen festgelegte Zeit gewartet, bevor eine neue Runde gestartet wird.

Die Ergebnisse werden in der Datenbank für die Auswertung festgehalten und durch eine Reihe von Views als Punkte berechnet. Die Gewichtung der einzelnen Scan-Operationen kann durch die betreuenden Personen individuell eingestellt werden.

Neben den Diensten sollen die Studierenden eigene Flags schützen und fremde abgreifen. Flags stellen geheime Informationen dar und sind eindeutige Strings, die durch das Hashen eines Seeds mit einem Hash-Algorithmus entstehen. Sie werden bei der Registrierung eines GameClients am GIS erstellt. Derzeit wird der *MD5*-Algorithmus verwendet und der Seed setzt sich aus Token + IP-Adresse der Gruppe + Geheimnis + Zähler zusammen.

Eine Abgabe von Flags ist über die in der Komponente Game Information System implementierte REST-Schnittstelle für angemeldete Studierende möglich. Die Gültigkeit der abgegebenen Flags wird mithilfe der im System vorliegenden Flags validiert. Eine Abgabe wird nur innerhalb der vorgesehenen Zeiten akzeptiert. Eventuelle Regelverstöße bei der Abgabe werden automatisch mit Strafpunkten geahndet.

Neben der Abgabe ermöglicht es die REST-Schnittstelle den betreuenden Personen das Spiel zu verwalten und zu steuern. Es unterstützt spielende und betreuende Personen, indem es den durch die Datenbank berechneten Spielstand, inklusive der einfließenden Punkte, anzeigt.

Die Schnittstelle stellt auch einen Flagshop, in dem mit einem Flagshop-Account weitere Flags gekauft werden können, und Challenges zur Verfügung.

Die Komponenten werden containerisiert, um die Verwaltung zu erleichtern und eine Portierbarkeit des Core Systems auf andere Computer zu ermöglichen.

Nach Aufschlüsselung verfügbarer Technologien wurde für das Backend *Flask*, für die Datenbank *PostgreSQL* und für das Frontend *React* gewählt.

Die Komponente der Weboberfläche (Frontend) konnte aus Zeitgründen nicht umgesetzt werden und liegt deshalb nur im Entwurf vor.

6.2 Ausblick

Im Folgenden wird ein Ausblick auf mögliche Veränderungen und Erweiterungen gegeben.

Implementierung Webseite

Die angedachte Single Page Applikation konnte aus Zeitgründen nicht in dieser Bachelorarbeit implementiert werden. Die Webseite kann anhand des vorgestellten Entwurfs und der diskutierten Technologie in einer weiteren Bachelorarbeit aufgegriffen werden. Hierbei sollte in jedem Fall ein eigener Vergleich aktueller Technologien sowie eine kritische Auseinandersetzung mit dem in dieser Arbeit erarbeiteten Entwurf für das Frontend durchgeführt werden. Es ist aber auch möglich, die Idee einer SPA zu verwerfen und eine klassische Multi Page Applikation, welche die API verwendet, zu entwickeln.

Implementierung weiterer Scan-Operationen

Durch den modularen Aufbau des Scanners ist es leicht möglich, neue Scan-Operationen zu erstellen. Dies kann genutzt werden, um auf dem Client weitere Schwachstellen und/oder Dienste zu implementieren, die dann durch den Big Brother überwacht und durch die Studierenden behoben oder online gehalten werden müssen. Herr Abts hat in seiner Bachelorarbeit[1] Ideen für Schwachstellen im Kapitel Ausblick genannt.

Veränderungen in der Anwendung

In der derzeitigen Realisierung wird zum Parsen einer Anfrage der im verwendeten Modul Flask-RESTful implementierte *reqparse (request parser)* verwendet. Dieser soll mit der Flask RESTful Version 2.0.0 entfernt und müsste daher gegen beispielsweise Marshmallow[2] oder den Flask internen Anfragen-Parser ausgetauscht werden. Es besteht derzeitig keine Dringlichkeit, da Flask-RESTful am 6. Februar 2020 erst in der Version *0.3.8* erschien. Sollte die Anwendung nicht weiterentwickelt werden, ist auch keine Änderung notwendig, da dann die Flask-RESTful Version unverändert bleibt.

Dummy Client

Im Versuch werden bei Bedarf GameClients gestartet, die nicht durch die Studierenden geschützt werden. Diese Dummy Clients werden als zusätzliches Angriffsziel benötigt und derzeit gleichwertig behandelt. Eine mögliche Änderung wäre, dass bei dem Dummy Client ein Bool-Wert in der Datenbank bei der Registrierung gesetzt wird. Dieser Wert könnte genutzt

[1] Abt16.
[2] `https://marshmallow.readthedocs.io/en/stable/`

werden, um diese GameClients von der Überwachung auszuschließen und in der Weboberfläche gesondert darzustellen.

Tokengenerierung für Flags

Es ist zu prüfen, ob der Token nicht auf dem Server generiert werden sollte und dem Client nur mitgeteilt wird. Bei der jetzigen Implementierung teilt der GameClient dem Server den Token mit, mit dem die Flags generiert werden sollen. Wenn die Studierenden es schaffen, diesen Token in der Mitteilung an den Server zu verändern, kann dieser auf einen Token gesetzt werden, bei dem die generierten Flags aus alten Versuchen bekannt sind.

Eine Tabelle aller historisch genutzten Tokens kann in der Datenbank angelegt werden, um sicherzustellen, dass jede Gruppe einen historisch einzigartigen Token besitzt. Damit wäre gewährleistet, dass Flags aus alten Versuchsterminen nicht verwendet werden können.

Lokale Webseiten überarbeiten

Die auf dem Client betriebenen Webseiten sollten nach der Implementierung der vom Server ausgelieferten Webseite an das Design angepasst werden. Dies erzeugt ein einheitliches Bild und verdeutlicht den Zusammenhang zwischen der lokalen Webseite und dem Versuchssystems.

Die Challenge- und Shopseiten sowie die entsprechenden Verweise können auf den lokalen Webseiten entfernt werden, da diese auf der vom Server ausgelieferten Webseite platziert werden sollen. Wird von einer Platzierung auf der Server Webseite abgesehen, müssen die angesprochen Seiten weiterhin vom GameClient ausgeliefert werden. Damit die bereitgestellte API des GIS verwendet wird, ist eine Änderung der Seiten auf dem GameClient notwendig.

Flagshop Flags

Es ist zu überlegen, zu bewerten und zu prüfen, ob eine Abgabe gegnerischer Flagshop Flags sinnvoll ist. Sollte Auswertung einer Sinnhaftigkeit ergeben, muss die implementierte Limitierung rückgängig gemacht werden.

Anhang

1 Installationsanleitung

Beide Programme sind unter Zuhilfenahme des Versionsverwaltungssystems *git* entwickelt worden und sind auf der GitLab Instanz des Gemeinschaftslabors Informatik (GLI) des Fachbereiches Elektrotechnik und Informatik der Hochschule Niederrhein im zugangsbeschränkten Repository *Its2* abgelegt. Dies ist erreichbar unter:
`https://gl.kr.hsnr.de/ezslabor/abschlussarbeiten/its2`.

Um die Anwendung nutzen zu können, muss das Repository heruntergeladen werden. Über die Weboberfläche kann dies als ZIP- oder TAR-Archiv heruntergeladen werden. Das Klonen mit der Methode `git clone` sollte dem Herunterladen vorgezogen werden, da Änderungen am Repository nur als Differenz herunter- oder hochgeladen werden müssen.

```
1 $ git clone
    ↪ https://gl.kr.hsnr.de/ezslabor/abschlussarbeiten/its2.git
```

Listing 1: git clone (bash)

Zur einfachen Nutzung der Anwendung ist im Repository neben einer *docker-compose.yaml* Datei auch ein *Makefile* angelegt. Die *docker-compose.yaml* Datei sorgt sich um das Zusammenspiel sowie die Konfiguration der einzelnen Container. Falls die Dockerimages für Big Brother und GIS nicht vorhanden sein sollten, werden diese automatisch erzeugt. Dies funktioniert nur, wenn die Ordner- und Dateistruktur unverändert bleibt.

In der *docker-compose.yaml* Datei ist darauf zu achten, dass Docker Socket in die Anwendung übergeben wird, da ansonsten die Steuerung des Scanners nicht möglich ist.

Im Makefile sind drei Befehle (*init, start, stop*) hinterlegt, die zur einfachen Nutzung von *docker-compose* beitragen sollen.

Der Befehl `init` führt eine Initialisierung der Anwendung aus. Dazu wird der bash Befehl (Listing 2) im GIS-Container ausgeführt wird. Dieser sorgt dafür, dass das Datenbankschema durch die hinterlegten Migrationen auf den benötigten Zustand gebracht wird. Danach wird ein Nutzer mit dem Namen *admin*, der Rolle *admin* und dem Passwort *admin* angelegt. Über diesen Benutzer können weitere Accounts angelegt werden. Der Account kann nach der Erstellung eines weiteren Administratoren-Accounts gelöscht werden.

```
$ pipenv run flask db downgrade base &&
  pipenv run flask db upgrade &&
  pipenv run flask user create admin --role admin --password
    ↪ admin
```

Listing 2: Initialisierung REST-Interface (bash)

Danach wird der Scanner gestartet und die Initialisierung der Service-Datenbank wird ausgeführt. Nachdem alle Services eingetragen worden sind, beendet sich der Container selbst und der *docker-compose* Befehl (Listing 3) wird ausgeführt. Dieser sorgt dafür, dass alle Container und Netzwerke, die durch *docker-compose* angelegt worden sind, entfernt werden.

```
$ docker-compose down
```

Listing 3: Aufräumen mit docker-compose down (bash)

Nach der Initialisierung ist die Anwendung einsatzbereit und kann über `make start` und `make stop` gestartet und beendet werden.

Bei `start` werden alle Container angelegt und im Hintergrund gestartet. Danach wird der Scanner Container beendet. Dies ist notwendig, damit das GIS den Scan-Container starten, pausieren und stoppen kann.

Bei `stop` wird nur der Befehl *docker-compose down* ausgeführt, der die bereits beschriebene Wirkung hat.

Sollten die Anwendungen einzeln verwendet oder installiert werden, muss die Dokumentation der jeweiligen Anwendung konsultiert werden.

2 Bedienungsanleitung

Das Game Information System ist ein RESTful-Interface und kann daher mit einem REST-Client angesprochen werden. Dazu kann die entworfene SPA implementiert werden. Es können aber auch Programme wie cURL („Werkzeug zur Übertragung von Daten von oder zu einem Server“ ([cUR])) oder Insomnia[1] verwendet werden. Insomnia wurde während der Entwicklung eingesetzt.

Über einen REST-Client können alle implementierten Routen (Tabelle 1) angesprochen werden. Für die Authentifizierung muss ein Access-Token über die Login-Schnittstelle abgeholt und in den nächsten Anfragen mitgesendet werden.

In der Dokumentation des GIS wird unter anderem auf die implementierten Routen eingegangen. Dort und in Tabelle 2/3 kann eingesehen werden, ob eine Authentifizierung notwendig ist. Welche Daten mitgesendet werden müssen und wie die Antwort aufgebaut ist, kann der Dokumentation entnommen werden.

[1] `https://insomnia.rest/`

Route	Methods
/	GET
/associate	GET, POST
/associate/<int:associate_id>	DELETE
/auth/flagshop/login	POST
/auth/login	POST
/auth/refresh	POST
/auth/revoke/access	DELETE
/auth/revoke/refresh	DELETE
/backup	GET
/backup/<int:backup_id>	GET
/challenge	GET, POST
/challenge/<int:challenge_id>	DELETE, GET, PUT
/challenge/solve	GET
/challenge/solve/<int:challenge_id>	DELETE, POST
/client	GET, POST
/client/<int:group_id>	DELETE, GET
/flag	POST
/flagshop/package	GET, POST
/flagshop/package/<int:package_id>	DELETE, PUT
/flagshop/transaction	DELETE, GET, POST
/flagshop/user	GET, POST
/flagshop/user/<user_name>	DELETE, PUT
/log	GET, POST
/log/old	GET
/match/control	DELETE, POST, PUT
/match/info	GET
/match/score	GET
/note	GET, POST
/note/<int:note_id>	DELETE, GET, PUT
/penalty	GET, POST
/penalty/<int:penalty_id>	DELETE, GET, PUT
/scanner	DELETE, GET, POST
/scanner/notify	POST
/secure	GET
/service	GET
/service/<int:service_id>	DELETE, GET, PUT
/setting	GET, PUT
/user	DELETE, GET, POST
/user/<int:user_id>	DELETE, GET, PUT
/user/import	POST

Tabelle 1: Übersicht über die implementierten Routen

Methode	Route	Admin	Supervisor	Player	Flagshop
GET	/	√	√	√	√
GET	/associate	√	√	χ	χ
POST	/associate	√	√	√	χ
DELETE	/associate/<int:associate_id>	√	√	√	χ
POST	/auth/flagshop/login	χ	χ	√	χ
POST	/auth/login	√	√	√	χ
POST	/auth/refresh	√	√	√	χ
DELETE	/auth/revoke/access	√	√	√	χ
DELETE	/auth/revoke/refresh	√	√	√	χ
GET	/backup	√	√	√	√
GET	/backup/<int:backup_id>	√	√	√	√
GET	/challenge	√	√	√	χ
POST	/challenge	√	√	χ	χ
GET	/challenge/<int:challenge_id>	√	√	√	χ
PUT	/challenge/<int:challenge_id>	√	√	χ	χ
DELETE	/challenge/<int:challenge_id>	√	√	χ	χ
GET	/challenge/solve	√	√	χ	χ
POST	/challenge/solve/<int:challenge_id>	χ	χ	√	χ
DELETE	/challenge/solve/<int:challenge_id>	√	√	χ	χ
GET	/client	√	√	√	√
POST	/client	√	√	√	χ
GET	/client/<int:group_id>	√	√	χ	χ
DELETE	/client/<int:group_id>	√	√	χ	χ
POST	/flag	χ	χ	√	χ
GET	/flagshop/package	√	√	χ	√
POST	/flagshop/package	√	√	χ	χ
PUT	/flagshop/package/<int:package_id>	√	√	χ	χ
DELETE	/flagshop/package/<int:package_id>	√	√	χ	χ
GET	/flagshop/transaction	√	√	χ	χ
POST	/flagshop/transaction	χ	χ	χ	√
DELETE	/flagshop/transaction	√	√	χ	χ
GET	/flagshop/user	√	√	χ	χ
POST	/flagshop/user	√	√	√	χ
PUT	/flagshop/user/<user_name>	√	√	χ	χ
DELETE	/flagshop/user/<user_name>	√	√	χ	χ
GET	/log	√	√	χ	χ
POST	/log	√	√	χ	χ
GET	/log/old	√	√	χ	χ

Tabelle 2: Berechtigungsmatrix der Routen 1/2

Methode	Route	Admin	Supervisor	Player	Flagshop
POST	/match/control	√	√	χ	χ
PUT	/match/control	√	√	χ	χ
DELETE	/match/control	√	√	χ	χ
GET	/match/info	√	√	√	√
GET	/match/score	√	√	√	√
GET	/note	√	√	√	√
POST	/note	√	√	χ	χ
GET	/note/<int:note_id>	√	√	√	√
PUT	/note/<int:note_id>	√	√	χ	χ
DELETE	/note/<int:note_id>	√	√	χ	χ
GET	/penalty	√	√	√	√
POST	/penalty	√	√	χ	χ
GET	/penalty/<int:penalty_id>	√	√	χ	χ
PUT	/penalty/<int:penalty_id>	√	√	χ	χ
DELETE	/penalty/<int:penalty_id>	√	√	χ	χ
GET	/scanner	√	√	χ	χ
POST	/scanner	√	√	χ	χ
DELETE	/scanner	√	√	χ	χ
POST	/scanner/notify	χ	χ	χ	χ
GET	/secure	√	√	√	χ
GET	/service	√	√	χ	χ
GET	/service/<int:service_id>	√	√	χ	χ
PUT	/service/<int:service_id>	√	√	χ	χ
DELETE	/service/<int:service_id>	√	√	χ	χ
GET	/setting	√	√	χ	χ
PUT	/setting	√	√	χ	χ
GET	/user	√	√	χ	χ
POST	/user	√	√	χ	χ
DELETE	/user	√	√	χ	χ
GET	/user/<int:user_id>	√	√	χ	χ
PUT	/user/<int:user_id>	√	√	χ	χ
DELETE	/user/<int:user_id>	√	√	χ	χ
GET	/user/import	√	√	χ	χ

Tabelle 3: Berechtigungsmatrix der Routen 2/2

```
class Config(object):
  database = {
    "username": "USERNAME",
    "password": "PASSWORD",
    "database": "DATABASE",
    "host": "HOST",
    "port": 5432
  }
  webserver = {
    "url": "webserver notify url",
    "auth": "Authentication Header"
  }
  operations = {
    "base": {
      "ping_timeout": 4,
      "ssh_timeout": 4
    },
    "sql": {
      "username": "SQL Username",
      "password": "SQL Password"
    },
    "sql_injection": {
      "admin": {
        "username": "ADMIN_SQL_USER",
        "password": "ADMIN_SQL_PASSWORD"
          },
          "control": {
            "flag": "FLAG",
            "value": "FLAG_VALUE"
          }
    },
    "htaccess": {
      "username": "HTACCESS_USER",
      "password": "HTACCESS_PASSWORD"
    },
    "bubble": {
      "bubble_port": 12345,
      "bubble_ng_port": 12345
    }
  }
```

Listing 4: Config Vorlage Big Brother

```
CREATE VIEW public.total_points AS
```

```
SELECT grp.id, grp.name,
(COALESCE(cha.total_points,
    ↪ ((0)::smallint)::bigint))::smallint AS challenge_points,
(COALESCE(dis.points, ((0)::smallint)::bigint))::smallint AS
    ↪ discover_points,
(COALESCE(ofe.points, ((0)::smallint)::bigint))::smallint AS
    ↪ offence_points,
(COALESCE(def.points, ((0)::smallint)::bigint))::smallint AS
    ↪ defence_points,
(COALESCE(tmp.service_points,
    ↪ ((0)::smallint)::bigint))::smallint AS service_points,
(COALESCE(fla.points_spend,
    ↪ ((0)::smallint)::bigint))::smallint AS flagsshop_points,
(COALESCE(pen.total_penalty,
    ↪ ((0)::smallint)::bigint))::smallint AS penalty_points,
((((((COALESCE(cha.total_points,
    ↪ ((0)::smallint)::bigint))::smallint +
    ↪ (COALESCE(dis.points,
    ↪ ((0)::smallint)::bigint))::smallint) +
    ↪ (COALESCE(ofe.points,
    ↪ ((0)::smallint)::bigint))::smallint) -
    ↪ (COALESCE(def.points,
    ↪ ((0)::smallint)::bigint))::smallint) -
    ↪ (COALESCE(tmp.service_points,
    ↪ ((0)::smallint)::bigint))::smallint) -
    ↪ (COALESCE(fla.points_spend,
    ↪ ((0)::smallint)::bigint))::smallint) -
    ↪ (COALESCE(pen.total_penalty,
    ↪ ((0)::smallint)::bigint))::smallint) AS total_points
FROM ((((((groups grp
LEFT JOIN challenge_points cha ON ((grp.id = cha.group_id)))
LEFT JOIN discover_points dis ON ((grp.id = dis.group_id)))
LEFT JOIN offence_points ofe ON ((grp.id = ofe.group_id)))
LEFT JOIN defence_points def ON ((grp.id = def.group_id)))
LEFT JOIN flagshop_points fla ON ((grp.id = fla.group_id)))
LEFT JOIN penalty_points pen ON ((grp.id = pen.group_id)))
JOIN (SELECT grp_1.id, sum(gsp.service_points) AS
    ↪ service_points
FROM (groups grp_1 LEFT JOIN group_service_points gsp ON
    ↪ ((grp_1.id = gsp.group_id)))
GROUP BY grp_1.id) tmp ON ((grp.id = tmp.id)));
```

Listing 5: SQL View Gesamtpunkte

```
self._host_up.start()
results['host_up'] = self._host_up.result
results['http_up'] = False
for scan_operation in self._scanner_list:
  results[scan_operation.name] = False

for scan_operation in self._scanner_list_http_up:
  results[scan_operation.name] = False
if results['host_up']:
  self._http_up.start()
  results['http_up'] = self._http_up.result

  future_list = []
  for scan_operation in self._scanner_list:
    future_list.append(pool.submit(scan_operation.start))

  if results['http_up']:
    for scan_operation in self._scanner_list_http_up:
      future_list.append(pool.submit(scan_operation.start))

  wait(future_list)
  for scan_operation in self._scanner_list:
    results[scan_operation.name] = scan_operation.result

  if results['http_up']:
    for scan_operation in self._scanner_list_http_up:
      results[scan_operation.name] = scan_operation.result

for key in results:
  if results[key]:
    online_count_add = 1
    else:
  online_count_add = 0

  try:
    self._cursor.execute( f"UPDATE {GROUP_SERVICE_TABLE_NAME}
      ↪ SET online_count = online_count +
      ↪ {online_count_add}, scan_count = scan_count + 1,
      ↪ was_online = {results[key]}, last_scan =
      ↪ CURRENT_TIMESTAMP WHERE group_id = {self._id} AND
      ↪ service_id = {self._get_id_of_scan_operation(key)}")
```

Listing 6: Big Brother Scanner

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Literatur

[Abt16] Benjamin Abts. „Überarbeitung und Erweiterung eines Client- / Server-Systems zur Durchführung von ITSicherheitsschulungen (Capture the Flag)“. Bachelor Arbeit. Hochschule Niederrhein, Juni 2016. 85 S.

[Ari18] Dan Arias. *Hashing in Action: Understanding Bcrypt*. 31. Mai 2018. URL: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ (besucht am 30. 07. 2020).

[Bac04] Daniel Bachfeld. *Giftspritze*. 6. Jan. 2004. URL: https://www.heise.de/security/artikel/Giftspritze-270382.html (besucht am 06. 07. 2020).

[BB] Cornelia Boenigk und Ralf Burger. *Was Ist PostgreSQL? | PostgreSQL*. URL: http://postgresql.de/was-ist-postgresql (besucht am 10. 07. 2020).

[Bei14] Hans Dieter Beims. *Web-Applikationen / REST*. Revision 2. 10. Dez. 2014.

[Bez15] Roberto Bez. *JavaScript: Einführung in React*. 12. Juni 2015. URL: https://www.heise.de/developer/artikel/JavaScript-Einfuehrung-in-React-2689175.html (besucht am 15. 07. 2020).

[BH20] Valerie Barsig und Jana Haase. *Cyber-Attacke auf das Potsdamer Rathaus*. 22. Jan. 2020. URL: https://www.pnn.de/potsdam/hacker-nutzten-sicherheitsluecke-cyber-attacke-auf-das-potsdamer-rathaus/25462398.html (besucht am 16. 05. 2020).

[BN19] Achim Berg und Michael Niemeier. „Wirtschaftsschutz in der digitalen Welt“. In: (11. Juni 2019), S. 13.

[Boe19] Eric Boersma. *Containerization: A Definition and Best Practices Guide - Plutora.Com*. 8. Juli 2019. URL: https://www.plutora.com/blog/containerization-best-practices (besucht am 18. 07. 2020).

[cM19] MDN contributors und Mozilla. *Django Introduction*. 30. Nov. 2019. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (besucht am 09. 07. 2020).

[cM20a] MDN contributors und Mozilla. *HTTP Headers*. 27. Apr. 2020. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers (besucht am 08. 07. 2020).

[cM20b] MDN contributors und Mozilla. *Server-Side Web Frameworks*. 3. Juni 2020. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks (besucht am 08. 07. 2020).

[Col20] Clifford Colby. *Windows 10 Dark Mode Is Here. Turn It on Now*. 15. Feb. 2020. URL: https://www.cnet.com/how-to/windows-10-dark-mode-is-here-turn-it-on-now/ (besucht am 19. 07. 2020).

[cUR] cURL. *Curl - How To Use*. URL: https://curl.haxx.se/docs/manpage.html (besucht am 15. 07. 2020).

[Daw14] Chris Dawson. *JavaScript's History and How It Led To ReactJS*. 25. Juli 2014. URL: https://thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/ (besucht am 15. 07. 2020).

[DB-] DB-Engines. *DB-Engines Ranking*. URL: https://db-engines.com/en/ranking (besucht am 09. 07. 2020).

[Dja] Django. *Django Documentation | Django Documentation | Django*. URL: https://docs.djangoproject.com/en/3.0/ (besucht am 09. 07. 2020).

[DO17] Thomas Drilling und Ulrike Ostler. *Was ist eine Datenbank?* 3. Aug. 2017. URL: https://www.datacenter-insider.de/was-ist-eine-datenbank-a-630652/ (besucht am 22. 07. 2020).

[Doc20] Docker Inc. *Overview of Docker Compose*. 15. Juli 2020. URL: https://docs.docker.com/compose/#features (besucht am 18. 07. 2020).

[FM20] Florian Flade und Georg Mascolo. *Cyberangriff auf Bundestag: Haftbefehl gegen russischen Hacker*. 5. Mai 2020. URL: https://www.tagesschau.de/investigativ/ndr-wdr/hacker-177.html (besucht am 16. 05. 2020).

[Gav18] Dave Gavigan. *The History of Angular*. 25. Mai 2018. URL: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7 (besucht am 14. 07. 2020).

[Goo] Google. *Dark Theme*. URL: https://material.io/design/color/dark-theme.html#usage (besucht am 19. 07. 2020).

[Goo20a] Google. *Angular - Angular Versioning and Releases*. 2020. URL: https://angular.io/guide/releases (besucht am 14. 07. 2020).

[Goo20b] Google. *Google Trends*. 13. Juli 2020. URL: https://trends.google.com/trends/explore?cat=733&date=2020-01-01%202020-12-31&q=React,Vue,Angular (besucht am 13. 07. 2020).

[Goo20c] Google. *Häufig Gestellte Fragen Zu Google Trends-Daten - Google Trends-Hilfe*. 2020. URL: https://support.google.com/trends/answer/4365533?hl=de&ref_topic=6248052 (besucht am 13. 07. 2020).

[HHH20] Simon Hurtz, Jan Heidtmann und Max Hoppenstedt. *Hacker-Angriff auf Gericht massiver als bislang bekannt*. 28. Jan. 2020. URL: https://www.sueddeutsche.de/digital/berlin-kammergericht-hacker-angriff-emotet-1.4775305 (besucht am 16. 05. 2020).

[Hoc] Hochschule Niederrhein. *Flyer Institut Clavis*. URL: https://www.hs-niederrhein.de/fileadmin/dateien/Institute_und_Kompetenzzentren/Clavis/Flyer_Institut_Clavis__5_.pdf (besucht am 16. 05. 2020).

[Hoc19] Hochschule Niederrhein. *Modulhandbuch Vollzeit BA Informatik*. 9. Dez. 2019. URL: https://www.hs-niederrhein.de/fileadmin/dateien/FB03/Studierende/Bachelor-Studiengaenge/PO2013/modul__bi.pdf (besucht am 16. 05. 2020).

[Hoc20] Hochschule Niederrhein. *Hackern die rote Karte zeigen - Neuer Studiengang Cyber Security Management*. 7. Feb. 2020. URL: https://www.hs-niederrhein.de/startseite/news/news-detailseite/?tx_news_pi1%5Bnews%5D=18990&cHash=e849d260ecd92cf53fc9c98f6dc9edaa (besucht am 16. 05. 2020).

[ION20] IONOS. *MariaDB vs. MySQL*. 10. März 2020. URL: https://www.ionos.com/digitalguide/hosting/technical-matters/mariadb-vs-mysql/ (besucht am 10. 07. 2020).

[it-19] it-daily.net. *IT-Security-Experten Werden Händeringend Gesucht - It-Daily.Net*. 3. März 2019. URL: https://www.it-daily.net/analysen/20773-it-security-experten-werden-haenderingend-gesucht (besucht am 16. 05. 2020).

[Kri17] Raúl Kripalani. *If You're a Startup, You Should Not Use React (Reflecting on the BSD + Patents License)*. 21. Nov. 2017. URL: https://medium.com/@raulk/if-youre-a-startup-you-should-not-use-react-reflecting-on-the-bsd-patents-license-b049d4a67dd2 (besucht am 15. 07. 2020).

[Kuc] A.M. Kuchling. *PEP 206 – Python Advanced Library*. URL: https://www.python.org/dev/peps/pep-0206/ (besucht am 09. 07. 2020).

[kul17] kulturbanause-Team. *Material Design – Die Designsprache von Google*. 26. Dez. 2017. URL: https://blog.kulturbanause.de/2016/01/material-design-die-designsprache-von-google/ (besucht am 19. 07. 2020).

[Lar17] Quincy Larson. *Facebook Just Changed the License on React. Here's a 2-Minute Explanation Why.* 28. Sep. 2017. URL: https://www.freecodecamp.org/news/facebook-just-changed-the-license-on-react-heres-a-2-minute-explanation-why-5878478913b2/ (besucht am 15. 07. 2020).

[Mel20] Ian Melnik. *Single Page Application (SPA) vs Multi Page Application (MPA): Pros and Cons - Merehead*. 17. Apr. 2020. URL: https://merehead.com/blog/single-page-application-vs-multi-page-application/ (besucht am 04. 06. 2020).

[MOV96] Alfred J. Menezes, Paul C. van Oorschot und Scott A. Vanstone. *Handbook of Applied Cryptography*. 1 edition. Boca Raton: CRC Press, 16. Dez. 1996. 780 S. ISBN: 978-0-8493-8523-0.

[Ora20a] Oracle Corporation. *MySQL :: MySQL 8.0 Reference Manual :: 1.3.1 What Is MySQL?* 2020. URL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html (besucht am 10. 07. 2020).

[Ora20b] Oracle Corporation. *MySQL :: MySQL 8.0 Reference Manual :: 1.3.2 The Main Features of MySQL*. 2020. URL: https://dev.mysql.com/doc/refman/8.0/en/features.html (besucht am 10. 07. 2020).

[Ora20c] Oracle Corporation. *Oracle® VM VirtualBox® User Guide*. Version 6.1.10. 5. Juni 2020. URL: https://www.virtualbox.org/manual/UserManual.html#features-overview (besucht am 06. 07. 2020).

[Pal10a] "Pallets. *Design Decisions in Flask — Flask Documentation (1.1.x)*. 2010. URL: https://flask.palletsprojects.com/en/1.1.x/design/#design (besucht am 09. 07. 2020).

[Pal10b] Pallets. *Foreword — Flask Documentation (1.1.x)*. 2010. URL: https://flask.palletsprojects.com/en/1.1.x/foreword/ (besucht am 09. 07. 2020).

[Pot20] John Potter. *React vs Vue vs Angular | Npm Trends*. 13. Juli 2020. URL: https://www.npmtrends.com/react-vs-vue-vs-angular-vs-@angular/core (besucht am 13. 07. 2020).

[Pre] Tom Preston-Werner. *Semantic Versioning 2.0.0*. URL: https://semver.org/lang/de/ (besucht am 14. 07. 2020).

[Qua17] Jürgen Quade. *Praktikum IT-Security*. Revision 2. 25. Sep. 2017.

[RG07] Christiane Rütten und Tobias Glemser. *Sicherheit von Webanwendungen*. 25. Jan. 2007. URL: https://www.heise.de/security/artikel/Sicherheit-von-Webanwendungen-270870.html (besucht am 06. 07. 2020).

[Ruh20] Ruhr24. *Hacker-Angriff legt IT-Systeme der Uni Bochum lahm - Klausuren ausgefallen*. 7. Mai 2020. URL: https://www.ruhr24.de/ruhrgebiet/bochum-rub-uni-hacker-angriff-webmail-moodle-news-universitaet-systeme-it-studierende-13753554.html (besucht am 16. 05. 2020).

[Sch16] Jacob Schatz. *Why We Chose Vue.Js*. 20. Okt. 2016. URL: https://about.gitlab.com/blog/2016/10/20/why-we-chose-vue/ (besucht am 14. 07. 2020).

[Sch20] Dennis Schirmacher. *Uni Gießen nähert sich nach Hacker-Attacke wieder dem Normalbetrieb*. 1. Juni 2020. URL: https://www.heise.de/newsticker/meldung/Uni-Giessen-naehert-sich-nach-Hacker-Attacke-wieder-dem-Normalbetrieb-4628715.html (besucht am 16. 05. 2020).

[Sos10] Alexander Sosna. „Konzeption und Realisierung eines modular aufgebauten Auswertungs- und Überwachungssystems zur Durchführung von IT-Sicherheitsschulungen.“ Bachelor Arbeit. Hochschule Niederrhein, Juni 2010. 98 S.

[SQL] SQLite. *Features Of SQLite*. URL: https://www.sqlite.org/features.html (besucht am 10. 07. 2020).

[Sta20] Stack Exchange. *Stack Overflow Developer Survey 2020*. The survey was fielded from February 5 to February 28. Feb. 2020. URL: https://insights.stackoverflow.com/survey/2020/#technology-most-loved-dreaded-and-wanted-web-frameworks (besucht am 13. 07. 2020).

[Tan20] Aaron Tan. *What Is CTF and How to Get Started!* 7. Mai 2020. URL: https://dev.to/atan/what-is-ctf-and-how-to-get-started-3f04 (besucht am 06. 07. 2020).

[tec18] techuz. *Top 9 Websites Built Using Vue.JS Front-End Framework*. 30. Aug. 2018. URL: https://www.techuz.com/blog/top-9-websites-built-using-vue-js/ (besucht am 14. 07. 2020).

[Teu18a] Marc Teufel. *Vue.Js Tutorial: Einführung in Das JavaScript-Framework*. 23. Juli 2018. URL: https://entwickler.de/online/javascript/tutorial-vue-js-einfuehrung-579851571.html (besucht am 14. 07. 2020).

[Teu18b] Marc Teufel. *Vue.Js Tutorial: So Entwickelt Man Komponenten Mit Vue.Js*. 25. Juli 2018. URL: https://entwickler.de/online/javascript/einfuehrung-vuejs-vue-579851816.html (besucht am 14. 07. 2020).

[The20] The PostgreSQL Global Development Group. *PostgreSQL: Documentation: 12: 2. A Brief History of PostgreSQL*. 2020. URL: https://www.postgresql.org/docs/current/history.html (besucht am 10. 07. 2020).

[Ven18] Ventzke Media. *Angular vs. React 2020 - Ein Vergleich Der Bibliotheken*. 5. Juni 2018. URL: https://www.ventzke-media.de/blog/angular-vs-react-vergleich.html (besucht am 14. 07. 2020).

[w3s] w3schools. *HTML Hidden Input*. URL: https://www.w3schools.com/tags/att_input_type_hidden.asp (besucht am 06. 07. 2020).

[Wah17] Dan Wahlin. *Die 5 wesentlichen Vorteile von Angular und TypeScript - Blog - t2informatik*. Ins deutsche Übersetzt von t2informatik am 19.01.2019. 26. Aug. 2017. URL: https://t2informatik.de/blog/softwareentwicklung/die-5-wesentlichen-vorteile-von-angular-und-typescript/ (besucht am 14. 07. 2020).

[WDR19] WDR. *Cyberattacke: Hackerangriff auf Universität Maastricht legt Wissenschaftsbetrieb lahm*. 27. Dez. 2019. URL: https://www1.wdr.de/nachrichten/rheinland/hacker-angriff-uni-maastricht-100.html (besucht am 16. 05. 2020).

[Weh20] Cornelia Wehner. *Software unter MIT Lizenz rechtssicher verwenden*. 2. Apr. 2020. URL: https://www.haerting.de/neuigkeit/software-unter-mit-lizenz-rechtssicher-verwenden (besucht am 15. 07. 2020).

[Wel19] Bianca Wellbrock. *IT-Sicherheit im Krankenhaus: Hack bringt Krankenhäuser zum Stillstand - PSW GROUP Blog*. 10. Sep. 2019. URL: https://www.psw-group.de/blog/it-sicherheit-im-krankenhaus-hack-bringt-krankenhaeuser-zum-stillstand/7175 (besucht am 16. 05. 2020).

[Wol15] Eberhard Wolff. *Microservices: Grundlagen flexibler Softwarearchitekturen*. 1., Auflage. Heidelberg: dpunkt.verlag GmbH, 1. Okt. 2015. 386 S. ISBN: 978-3-86490-313-7.