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: 12. Juli 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 Experten[it-19], welche die bestehenden und neu designten Systeme auf Sicherheitslücken prüfen und Lösungsvorschläge zur deren Behebung präsentieren. Auch werden Experten gesucht, welche die im Unternehmen bestehenden Prozesse prüfen und neue Prozesse 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 auch in jüngster Vergangenheit deutlich, nachdem beispielsweise diverse Universitäten wie Gießen, Maastricht und Bochum Ende 2019 Ziele von Hackerangriffen geworden sind.[Sch20][WDR19][Ruh20] Aber nicht nur Universitäten sind betroffen, so ist neben Gerichten, Stadtverwaltungen und Krankenhäusern bereits der Deutsche Bundestag von Hackern angegriffen und kompromittiert worden.[HHH20][BH20][Wel19][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. Aus dieser Studie geht hervor, dass im Jahr 2019 von Datendiebstahl, Industriespionage oder Sabotage 75% der befragten Unternehmen[1] betroffen und 13% vermutlich betroffen waren. Die Zahlen der betroffenen Unternehmen ist steigend. Im Jahre 2015 waren „nur“ 51% betroffen und 28% vermutlich betroffen. Die Unternehmen beziffern den Schaden auf 102,9 Milliarden Euro pro Jahr.[BN19]

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

Aber 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 Fachbreich 03 Elektrotechnik und Informatik aufgenommen. So werden dort im fünften Semester in der

[1] Die Grundlage der Studie sind 1070 (2019) und 1074 (2015) befragte Unternehmen

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

1.1 Motivation

Neben diversen Meldungen zu erfolgreichen Angriffen auf Unternehmen und öffentliche Körperschaften und durch die Veranstaltung IT-Sicherheit im fünften Semester, besonders herauszuheben ist hier das Praktikum[2], bin ich auf das Thema IT-Sicherheit aufmerksam geworden.

Die zunehmenden Vorfälle zeigen, dass ein breites Bewusstsein für IT-Sicherheit geschaffen werden muss.

Der Versuch „Catch me, if you can“ versucht dieses Bewusstsein zu schaffen, in dem die Studierenden sowohl in die Rolle des Angreifers als auch die des Schützers schlüpfen.

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

1.2 Aufgabenstellung

Begleitend zu der Veranstaltung IT-Sicherheit für die Studiengänge Bachelor Informatik und den Bachelor Elektrotechnik des Fachbereichs 03 Elektrotechnik und Informatik der Hochschule Niederrhein werden 3 Versuche im Rahmen des Praktikums durchgeführt. Diese sollen den Studierenden praktische Erfahrungen ermöglichen.

Der zweite Versuch namens „Catch me, if you can“ stellt einen Vergleichswettbewerb dar. An diesem Wettbewerb nehmen mehrere Teams teil, welche sich alle in einem gemeinsamen Computernetzwerk befinden. Die Aufgabe der Teams besteht darin, festgelegte Programme/-Dienste abgesichert bereit zustellen, 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 logisch gesehen Passwörter oder private Bilder und werden durch sogenannte Flags repräsentiert. Eine Flag ist eine generierte Zeichenfolge mit fester Länge.

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

[2] Praktikum ist hierbei mit einer Pflichtübung vergleichbar

Ziel meiner Arbeit ist die Modernisierung und Verbesserung dieses Auswertungs- und Überwachungssystems.

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 verschieden Komponenten des Servers erstellt.

Anhand der abgeleiteten Anforderungen und des Entwurfs der verschiedenen Komponenten werden im Kapitel 4 verschiedene 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 folgende 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 Bedingungsanleitungen

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)
- Praktikum (1 Semesterwochenstunde)

Vorlesung

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

Übung

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

Praktikum

Die Versuche des Praktikums finden im monatlichen Turnus (3x im Semester) á 4 Stunden statt. Bei 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.

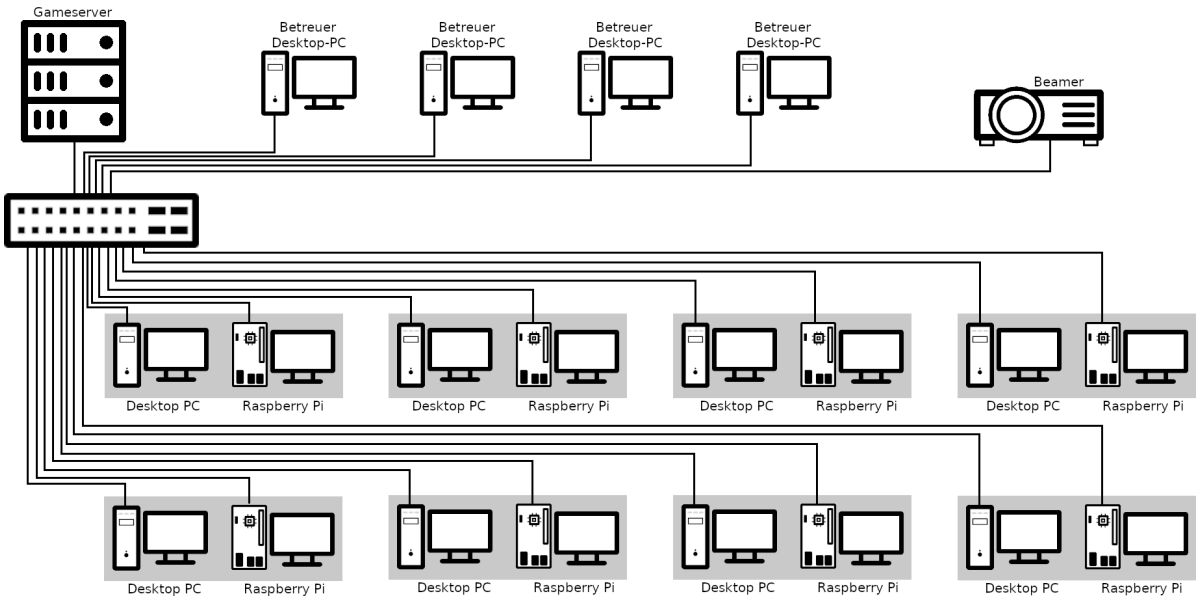

Abbildung 2.1: Übersicht über die Laborausstattung (Netzwerktopologie)

Wie in der Abbildung 2.1 zu sehen, ist das Labor mit acht Gruppenarbeitsplätzen für Studierende sowie Arbeitsplätzen für die Betreuer und Mitarbeiter ausgestattet. Ein Betreuerarbeitsplatz kann zu einem neunten Gruppenarbeitsplatz umfunktioniert werden.

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

[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, welche 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 das auf Inkompatibilitäten von Softwares der verschieden Versuche geachtet werden muss. Auch bietet VirtualBox die Möglichkeit sogenannte Snapshots anzulegen. Ein Snapshot ist eine Momentaufnahme eines aktuellen Systemzustandes. Diese stellt eine Komplettsicherung des Systems dar. Durch diese ist es möglich, die Systeme auf den gleichen Stand zu bringen. Dieses wird benötigt, um eine Vergleichbarkeit zwischen den Studierenden zu gewährleisten.[Ora20c]

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

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

Alle Rechner, 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 Spieler 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 mehrere 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.

Der Versuch ist in drei Phasen untergliedert.

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

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 vorhanden Diensten abgesichert werden 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 zu suchen. Da der Ablageort der Flags auf allen Systemen gleich ist, kann durch diese Information im Spielverlauf ein Angriff schneller Flags einbringen.

Wettbewerb

Die Wettbewerbsphase selber dauert circa 140 Minuten. In dieser Zeit sind Angriffe auf fremde Gastsysteme erlaubt und ausdrücklich gewünscht. Eine weitere Absicherung ist weiterhin möglich. Das System sollte auf fremde Aktivitäten überwacht werden. Diese Aktivitäten sollten schnellstmöglich unterbunden werden, da die Angreifer Flags klauen können, und so dem Team Strafpunkte einbringen. Auch kann die Zeit für die Lösung von zur Verfügung stehender 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 möglich. Die Studierenden erstellen für ihren anzufertigen Versuchsbericht ein Screenshot der Punkteübersicht. Eine Nachbesprechung ist optional und mit maximal 30 Minuten angesetzt.

Während des Wettbewerbs gelten die aufgelisteten Regeln. Es handelt sich hierbei 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 benutzbar 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 und Verbesserungsmöglichkeiten 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 worden ist, stellt der Dienst für eine bestimmte Zeit (Timeout) keine weitere Flag 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 zu dem Port *12321* möglich ist, in dem die Operation eine Telnet Verbindung öffnet und prüft, ob die Verbindung erfolgreich war.

ScanWebUp Jedes Gastsystem stellt unter zuhilfenahme eines Apache Web Servers und php-Dateien Webseiten und Daten bereit, welche über einen Web Client abgerufen werden können. Dazu muss auf Port *80* der HTTP- und auf Port *443* der HTTPS-Dienst laufen und erreichbar sein. Dieses verifiziert die Scan-Operation in dem sie eine Socket Verbindung zu den Ports *80* und *443* geöffnet 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 einen 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 überein stimmen. Besonderheit hierbei ist, dass statt einer validen Kombination aus Nutzernamen und Passwort eine sogenannte SQL-Injection (wird in Kapitel 2.4.2 genauer erläutert) im Nuterznamen übergeben wird. Sollte die Anfrage alle gespeicherten Nutzerdaten zurück geben ohne das eine Authentifizierung statt findet, ist die SQL-Injection weiterhin möglich.

ScanXSSSave Diese Scan-Operation prüft, ob die auf dem Gastsystem implementierte XSS Schwachstelle behoben worden. Dazu wird die Webseite mit präparieren Inhalt aufgerufen. Auf die Vorgehensweise wird in Kapitel 2.4.2 eingegangen. In der Rückgabe wird geprüft, ob dieser ungefiltert auf der Webseite zu finden ist. Sollte diese 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. Sollte dieser möglich sein, haben die Studierenden dieses unsichere Passwort nicht geändert haben. Des Weiteren wird geprüft, ob das htaccess Passwort, welche die phpMyAdmin Anwendung schützen soll, geändert worden ist.

ScanFTPSave Auf dem Client System läuft ein FTP Server, welcher ohne Login (Nutzername & Passwort) Daten bereitstellt. Der Scan prüft, ob ein sogenannter Anonymous Login möglich ist, in dem eine FTP Verbindung ohne Login aufgebaut wird. Sollte die Verbindung erfolgreich sein, ist der Anonymous Login immer noch möglich.

ScanTelnetSave Ein Telnet Server horcht auf Verbindungen auf Port *23*. Da dieser Dienst nicht benötigt wird, soll er durch die Studierende 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 sogenanntem 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. Des Weiteren 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 Wort „*Aufgabe*“ und einem *Zähler* zusammen.

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 Überwachungssystems akzeptiert. Eine 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 im bestenfall zu verhindern.

Damit pro Team mehrere eindeutige Flags geniert 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 die GUI (Graphical User Interface) für die Studierenden und Betreuer dar. Hier kann der aktuelle Punktestand angesehen werden. Auch wird in der GUI dargestellt, welches Team welchen Service abgesichert hat, inklusive der negative Punkte für nicht abgesicherte Dienste, und wie viele Strafpunkte das jeweilige Team erhalten hat.

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

Die Betreuer haben die Möglichkeit über die Web-GUI ein neues Spiel anzulegen, 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 den Studierenden weitere Flags mit ihren Punkten zu kaufen. Der Kauf von Flags lohnt sich schon, da die verkauften Flags mehr Punkte bringen als sie kosten. Um einen Einkauf im Flagshop durchzuführen, müssen die Teams sich vorher einen Account erstellen. Die Registrierung erfragt neben dem benötigten Benutzernamen und Passwort auch für den Flagshop irrelevante Daten ab. Diese ähneln persönlichen Informationen, welche bei den meisten Onlineshop angegeben werden müssen. Das Format, hier die Repräsentation als Zahl oder String sowie die Länge, und die Erforderlichkeit der Daten wird nur im HTML-Formular festgelegt. Durch eine Manipulierung des Formulars kann dieses mit nicht konformen oder nicht vorhanden Daten abgesendet werden. Für jeder der nicht vorhanden 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 Studierende 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. Bei der ersten Art müssen sich die beiden Pakete gleichzeitig im Warenkorb befinden und die Identifikationsnummern (ID) dieser auf nicht vorhanden 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 Paket kostenlos zu erhalten. Dazu muss im Warenkorb, dass 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 der Browser betrachtet und verändert werden. [w3s]

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

Challenges Derzeitig sind fünf Challenges implementiert, welche vom System in zufälliger Reihenfolge an interessierte Teams verteilt werden. Eine abgeschlossene oder abgebrochene Challenge, durch das Neuladen der Webseite oder der Betätigen der Zurück-Taste, kann nicht wiederholt werden. Eine Challenge kostet 10 Punkte. Nach erfolgreichem Abschließen einer Challenge gibt es 10 Punkte plus eine gewisse Anzahl an Punkten für das Absolvieren der Aufgabe. Die folgenden Challenges sind implementiert.[Abt16, S.19-20]

Aufgabe 1: robots.txt Die Studierenden sollen in dieser Challenge lernen, dass die *robots.txt* Datei kein Zugriffsschutz darstellt. Diese bittet nur Suchmaschinen die angegeben 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 vorhanden Ordner finden. In diesem befindet sich die Lösung zur Challenge.

Aufgabe 2: JavaScript-Login-Bypass Bei dieser Challenge ist das benötigte Passwort zur Lösung der Challenge als Klartext im Quelltext versteckt. Das Verstecken ist 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 dieses nicht möglich, da die Leerzeilen entfernt werden und das Passwort daher oben im Quelltext zu sehen ist.

Aufgabe 3: Form-Modification In dieser Challenge sollen die Studierende 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 serverseitige überprüft werden, da hier die Regeln der Überprüfung nicht durch die Nutzer verändert werden können.

Die Aufgabe besteht darin einen bestimmten Login Namen aus einem Drop-Down-Menü auszuwählen. Da der Name nicht in dieser Liste ist, müssen die Studierenden das HTML Formular so manipulieren, dass diese den geforderten Namen auswählen können.

Aufgabe 4: JavaScript-Substrings Das Passwort, welches 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 durch die Studierenden gegebenen Passwort ist.

Aufgabe 5: URL-Hex-Injection Die Studierenden sollen an geheime Informationen in einem Ordner gelangen, welcher nach einem Wert aus dem Hexadezimalsystem (Basis 16 statt Basis 10, wie beim Dezimalsystem) benannt ist. 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 selber durch einen Hexadezimalwert dargestellt werden kann.

Abgabe von Flags

Um Flags abgeben zu können, müssen die Studierenden sich mit ihren Zugangsdaten, welche 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. Des Weiteren ist es nicht möglich mehrere Flags gleichzeitig abzugeben.

2.4.2 Komponenten des Clients

Da sich die Bachelorarbeit mit der Modernisierung des Auswertungs- und Überwachungssystem 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 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, welche 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 wird 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 selber erfolgreich durchgeführt hat.

Neben diesen Schwachstellen gibt es eine Registrierung für den Flagshop. Dieses erfordert einige Eingaben, wie Name, Alter, Postleitzahl und vieles mehr. Die Eingaben sind im HTML-Formular als Pflicht markiert und haben eine Vorgabe 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. 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: 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]

2.5 Schnittpunkte zwischen Server und Clients

Der Server und die Clients laufen auf getrennten Systemen. Da die Studierende 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 ist,
- der Webserver erreichbar ist,
- der Bubble-Server erreichbar ist,
- der Login zum Membersbereich erreichbar und abgesichert ist,
- die Kundenbewertung erreichbar und abgesichert ist,
- ob das SQL Passwort geändert worden ist,
- ob der FTP Server gegen unautorisierten Zugriff abgesichert ist und
- ob 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, welche 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 Betreuer)
- Dokumentation aller Events durch Protokollierung der einzelnen Aktionen des Systems
- Bereitstellung von Challenges, damit Studierende sich weiter Punkte erarbeiten können

- Bereitstellung eines (Flag-) Shops, bei dem mehrere Lücken genutzt werden können, um Flags zu erhalten
- Einstellungen des Spiels sollen durch Betreuer geändert werden können
- Verwaltung von Benutzern (Administratoren und Spielern)
- Zugangskontrolle für teilnehmende Studierende durch Prüfung der Hackits
- 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 sind alle Daten an einem Ort gebündelt. Zweitens kann 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 soll hierdurch die Verständlichkeit verbessert und die Challenges sowie der Flagshop besser platziert werden.

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

Module sparsam nutzen Bei der Implementierung der Software sollte, so weit dieses notwendig und sinnvoll ist, auf bereits vorhandene Module und Frameworks zurückgegriffen werden. Durch die Minimierung von Abhängigkeiten ist die Wartung von 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, damit nicht inaktive Module/Frameworks mit eventuellen Schwächen genutzt werden.

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

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

3.2 Containerisierung

(todo: Weiter ausarbeiten, mit quellen versehen, andere Position) Durch die Nutzung von Containerisierung ist es möglich die Anwendung agiler und skalierbarer zu betreiben. Auch werden so Abhängigkeiten zwischen den Komponenten reduziert und eine losere Kopplung erreicht, dies führt auch zu einer klareren Struktur und übersichtlicheren Programmierung.

Die Containerisierung wird mithilfe von Docker erreicht. Docker ist ein seit 2013 bestehende Open-Source Containerisierungssoftware, welche ein weite Verbreitung genießt. Sie war der de facto Standard für Containerisierung, wird in den letzten Jahren, jedoch durch Container-Orchestrierung Softwares, wie Kubernetes oder OpenShift, stückweise verdrängt.

Da eine Container-Orchestrierung viel mehr bietet als im Rahmen dieses Projektes benötigt wird und eine Verwaltung und Installation von solch einer Software nicht einfach ist, wird Docker, welches den Anforderung genügt, verwendet. So bietet Docker den Vorteil der einfacheren Bedienbarkeit und Installation sowie der benötigten Flexibilität.

Mithilfe von Docker Compose ist es möglich Applikationen, welche aus mehreren Containern bestehen auch Stack genannt, zu verwalten und zu betreiben. Auch ist einfacher Abhängigkeiten zwischen den Containern zu modellieren und Verbindungen zwischen Containern herzustellen.

links: https://cloud.google.com/containers?hl=de, https://entwickler.de/leseproben/containerisierung-der-it-579775782.html

3.3 Übersicht

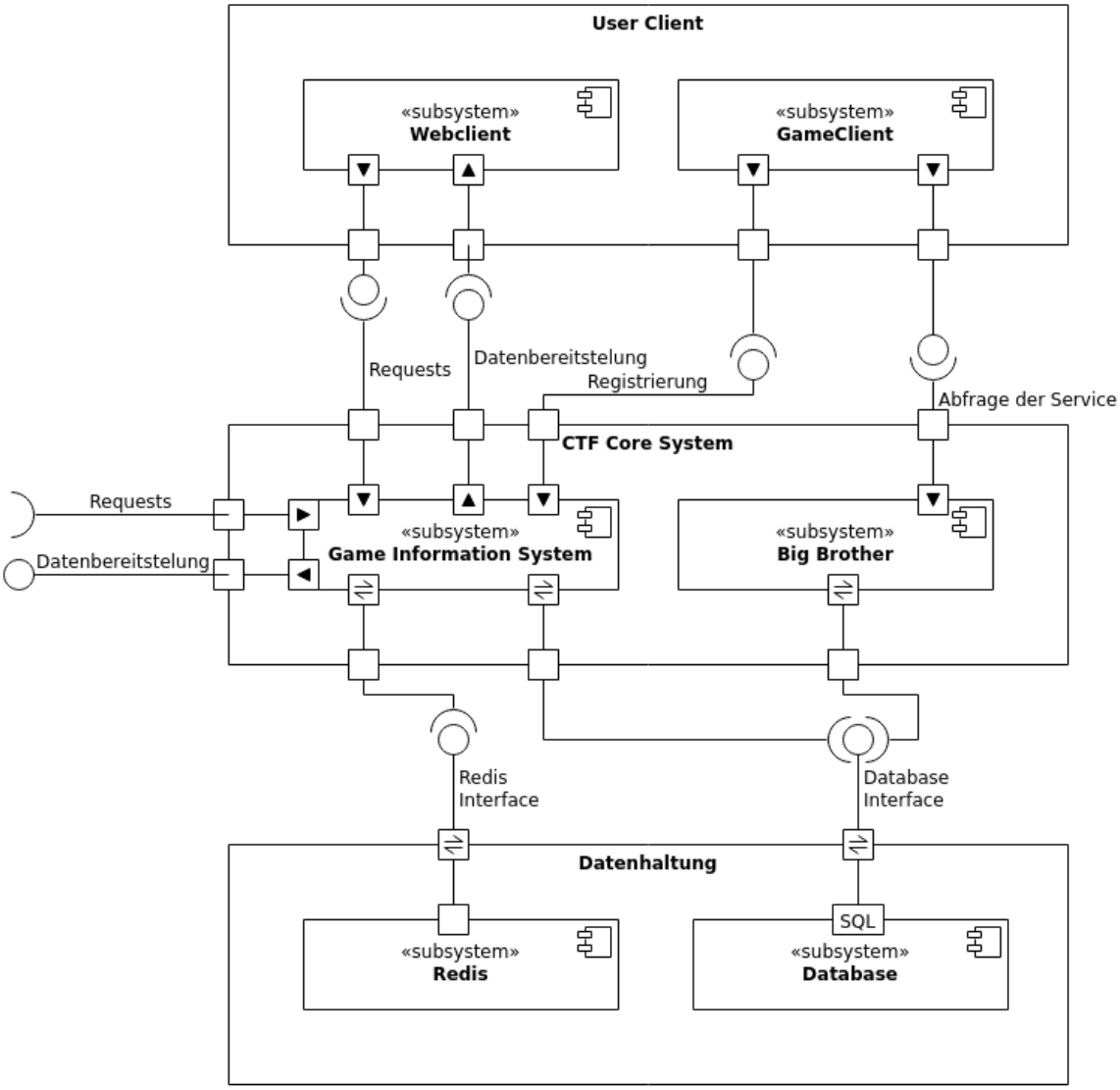

Abbildung 3.1: Übersicht über die Anwendung (Komponentendiagramm)

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 für den Versuch relevanten Komponenten.

Bei der Komponente Webclient handelt es sich um einen geläufigen Webbrowser (häufig wird Firefox genutzt), welcher 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 installierten 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 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, welcher für die Überwachung des *GameClients* mit seinen Schwachstellen zuständig ist. Die Ergebnisse werden in der *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 der Komponente *Database* aufgenommen werden, die Trennung erfolgt, aber auf Grundlage der Verwendung innerhalb der Software Architektur. Die Redis-Datenbank wird als Cache verwendet und persistiert die Daten nicht.

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

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 alle 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, welcher das eigene System überwacht. Ein Master Scanner sammelt die von den Worker Scannern erstellten Ergebnisse ein und speichert diese in einer Datenbank ab.

Der Vorteile 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. Des Weiteren 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. Dem Ergebnis kann vertraut werden, da es auf einem System ohne Einfluss der Mitspielenden berechnet wird.

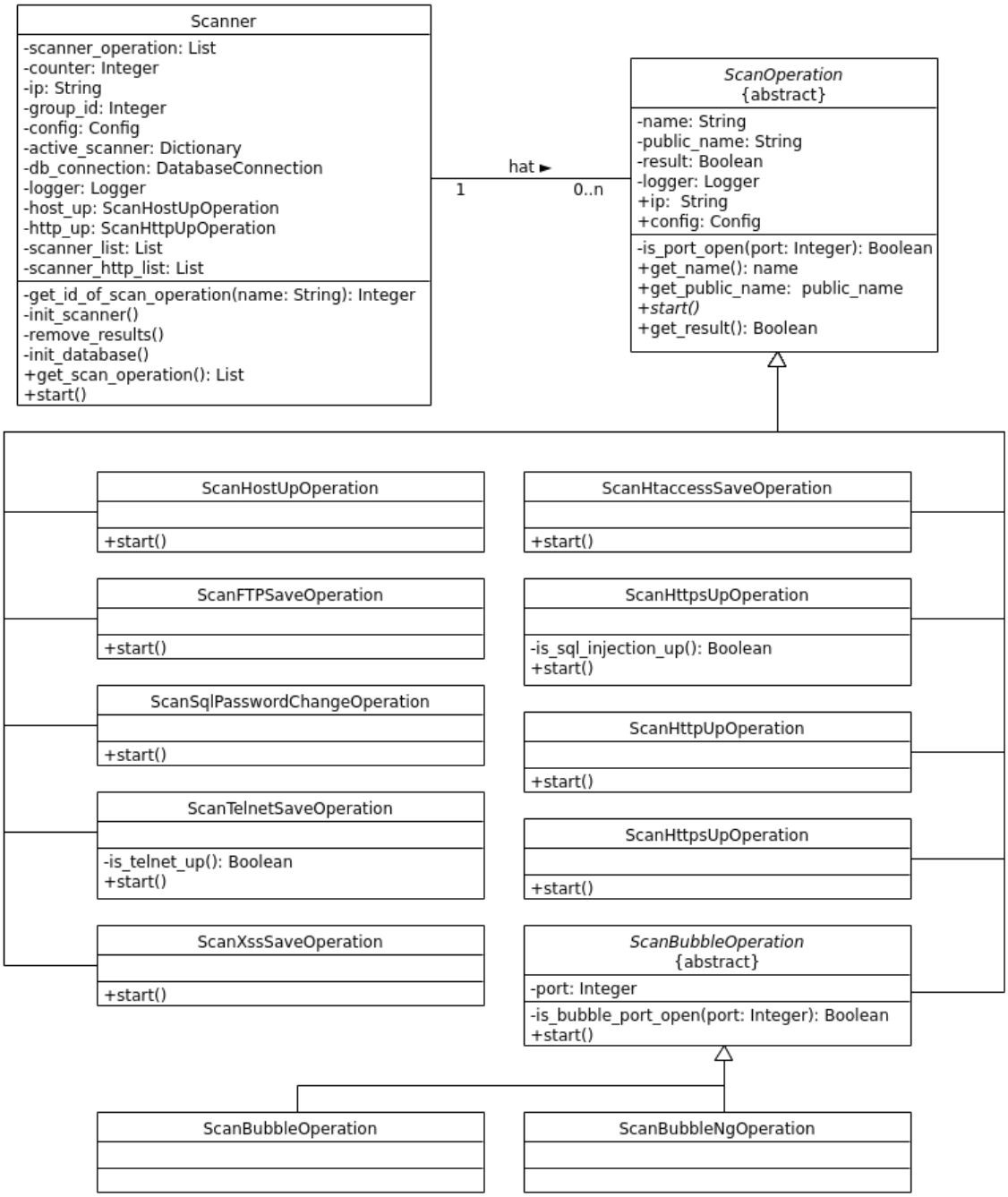

Abbildung 3.2: Klassen der Big Brother Komponente (Klassendiagramm)

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 Funktion *start()*. Diese wird von den abgeleiteten Klassen implementiert und ermöglicht das Starten der einzelnen Scan-Operationen.

Ebenfalls speichern alle Scan-Operationen ihr Ergebnis in der privaten Variable *result*. Mithilfe der von der abstrakten Klasse implementierten Funktion *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 / den aktiven Diensten. Des Weiteren stellt die abstrakte Klasse die Funktion *is_port_open* bereit, welche 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 max. eine Scan-Operation starten und beinhaltet / verwaltet alle Scan-Operationen für ein Game Client.

Um mehrere Game Clients zu überwachen, werden mehrere Objekte der Klasse Scanner benötigt.

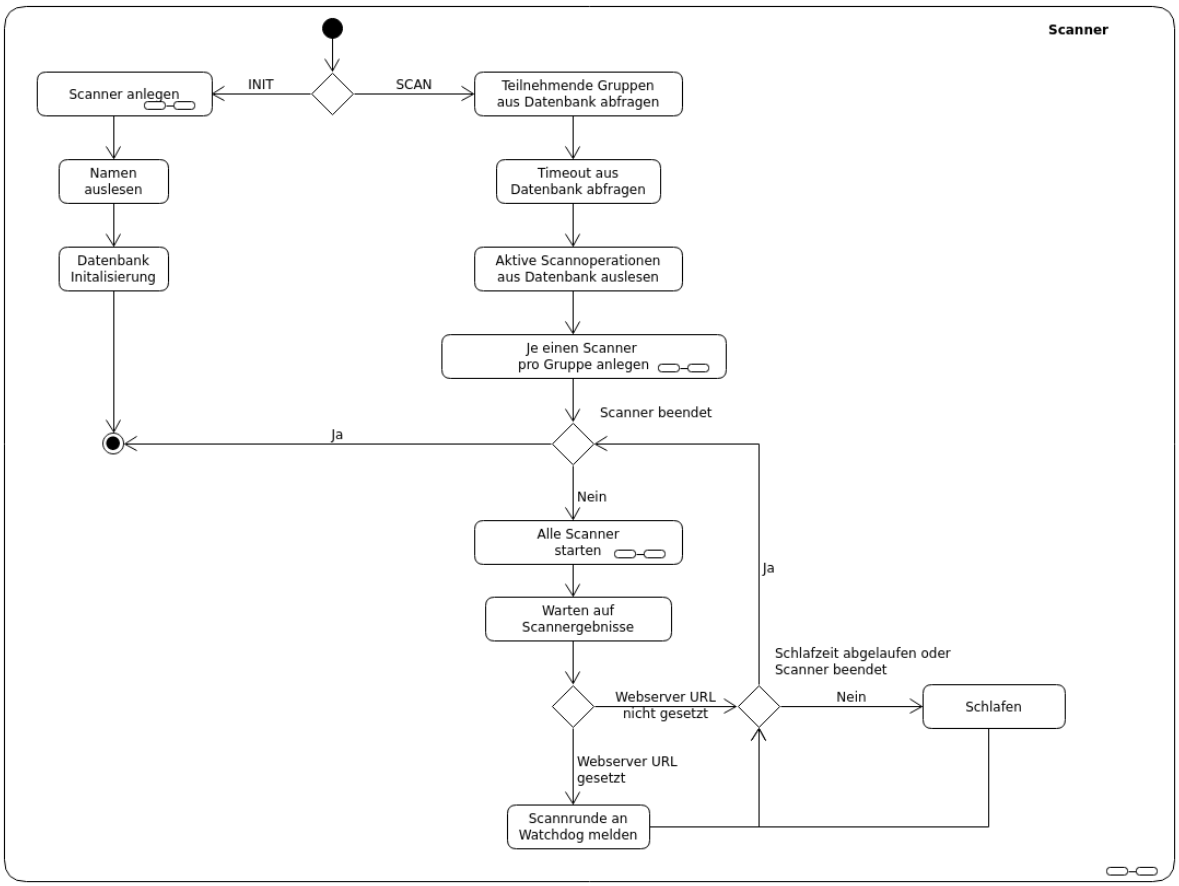

Abbildung 3.3: Ansicht des Scanners (Zustandsdiagramm)

Bei Starten des Scanners wird die zu bearbeitende Aufgabe spezifiziert.

Bekommt der Scanner die Aufgabe „INIT“, soll dieser die Service Datenbank mit den implementieren Scan-Operationen füllen, sodass Administratoren diese an- oder ausschalten können. Um die Service Datenbank zu füllen wird zunächst ein Dummy der Klasse Scanner angelegt. Aus diesem Dummy Objekt werden von allen Scan-Operationen der Anzeigenamen und

der interne Name ausgelesen. Nach dem Auslesen alle Operationen werden die erhalten Daten gebündelt in die Service 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 Game Clients gestartet. Hierzu werden die teilnehmenden Gruppen und das Intervall der Scannerdurchgänge aus der Datenbank ausgelesen. Neben diesem werden die aktiven Scanner aus der Datenbank abgefragt. Sind all Informationen vorhanden, wird für jede Gruppe ein Scanner Objekt der Klasse Scanner erstellt. Bei der Erstellung werden die aktiven Scan-Operationen sowie die zu überwachende Gruppe übergeben. Das Scanner Objekt legt dann für die benötigten Scan-Operationen die jeweiligen Objekte an.

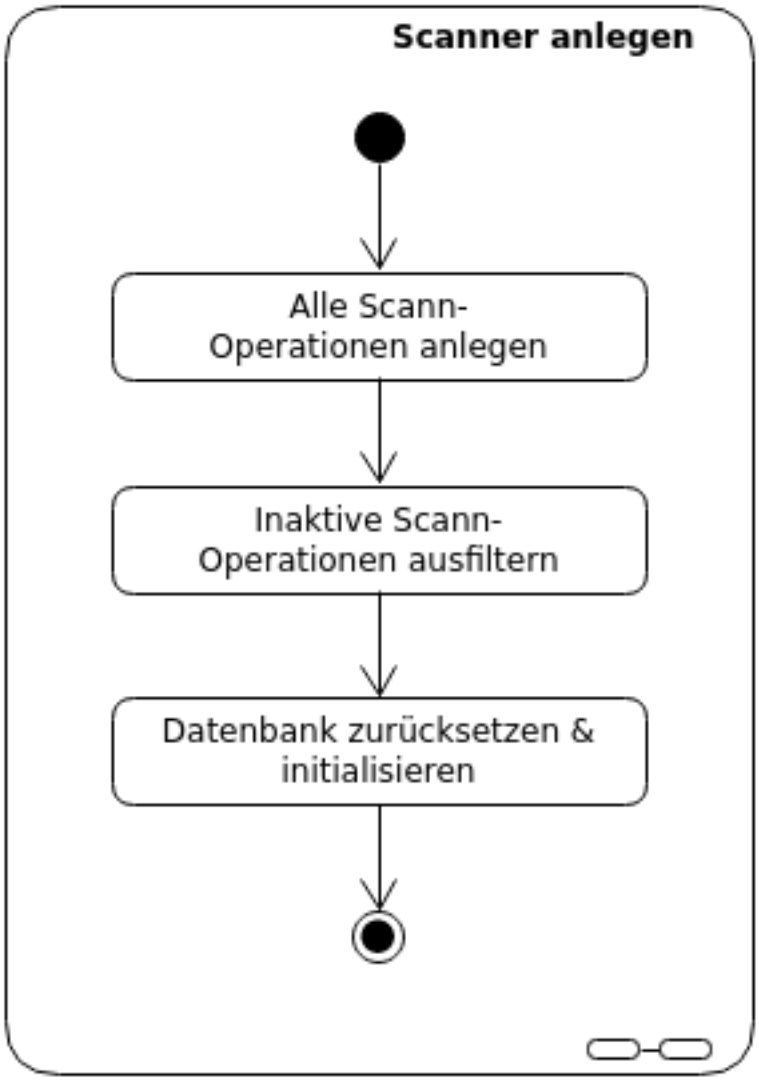

Abbildung 3.4: Erstellung eines Scanners (Zustandsdiagramm)

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

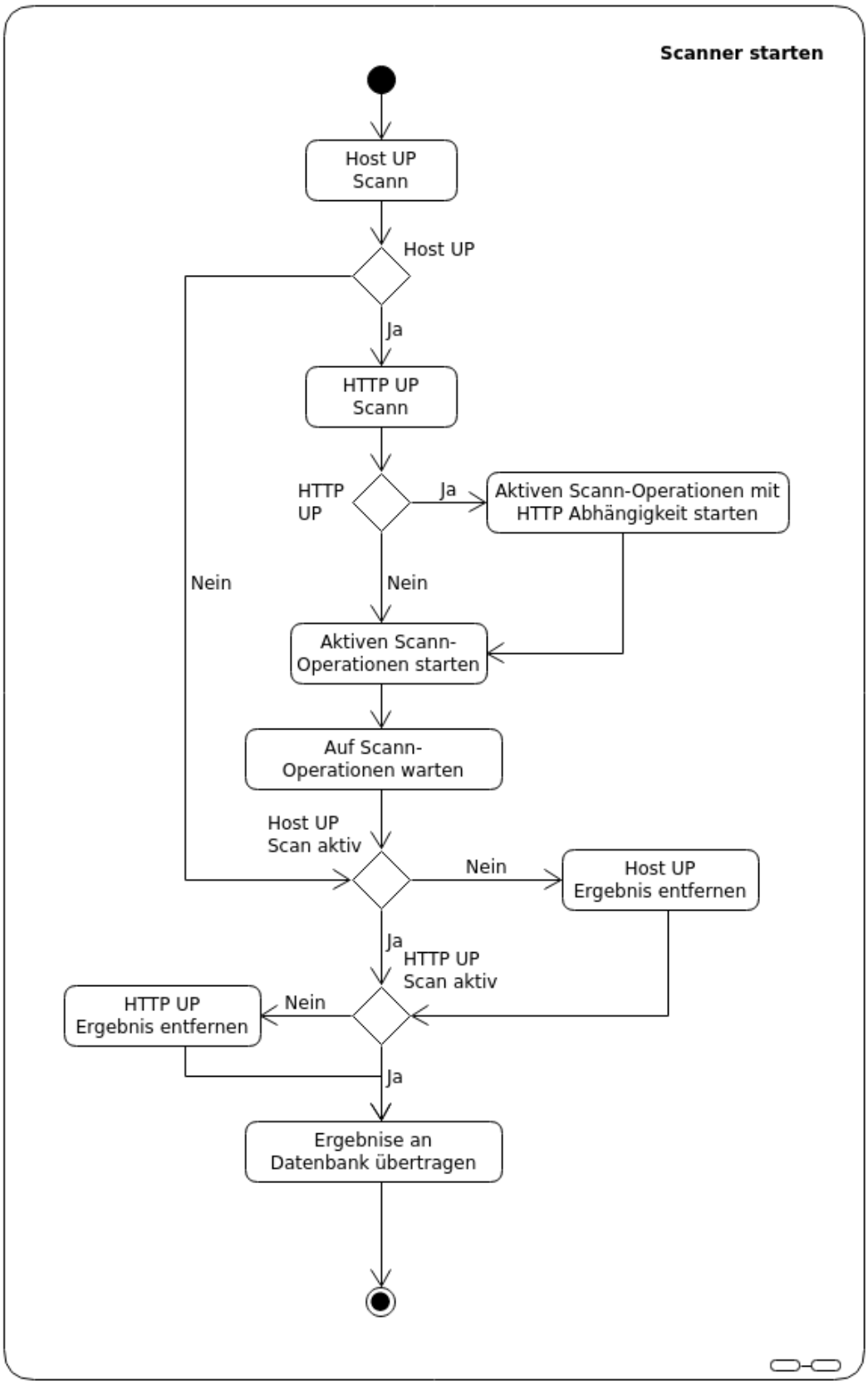
Scanner starten
Host UP Scann
Host UP
Ja
HTTP UP Scann
HTTP UP
Ja
Aktiven Scann-Operationen mit HTTP Abhängigkeit starten
Nein
Nein
Aktiven Scann-Operationen starten
Auf Scann-Operationen warten
Host UP Scan aktiv
Nein
Host UP Ergebnis entfernen
Ja
HTTP UP Scan aktiv
HTTP UP Ergebnis entfernen
Nein
Ja
Ergebnise an Datenbank übertragen

Abbildung 3.5: Starten eines Scanners (Zustandsdiagramm)

Beim Ausführen der Funktion *start()* des Scanners wird zu nächst geprüft, ob das entfernte System erreichbar ist. Sollte dies nicht der Fall sein, werden alle nachfolgenden Scan-Operationen nicht durchgeführt, da diese fehlschlagen werden. Im Anschluss 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, welche auf dem HTTP Dienst basieren, mit in die Liste der abzuarbeiten Scan-Operationen aufgenommen. Danach werden alle verbleibenden Scan-Operationen nebenläufig 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 ist, werden diese aus dem Ergebnis entfernt. Danach übermittelt der Scanner die gesammelten Ergebnisse zur Datenbank und beendet seine Scan-Runde.

3.4.3 Scan-Operationen

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

- Host-Up
  Prüft, ob der entfernte Rechner mit Hilfe von ICMP Paketen erreichbar ist
- Bubble-Up
  Prüft, ob der Bubble Server erreichbar ist und ob die Telnet Steuerung funktioniert
- BubbleNg-Up
  Siehe Bubble-Up
- FTP-Save
  Prüft, ob der FTP Server erreichbar ist und ob die Nutzung des Anonymous Login unterbunden worden ist
- Htaccess-Save
  Prüft, ob die Kombination aus Nutzername / Passwort des Htaccess Schutz geändert worden ist
- SQL-Injection-Save
  Prüft, ob die SQL-Injection im Login zum Membersbereich verhindert worden ist
- SQL-Password-Save
  Prüft, ob das lokale Passwort des SQL-Nutzers root geändert worden ist

- Telnet-Save
  Prüft, ob der Telnet Server deaktiviert / deinstalliert worden ist

- 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 worden ist.

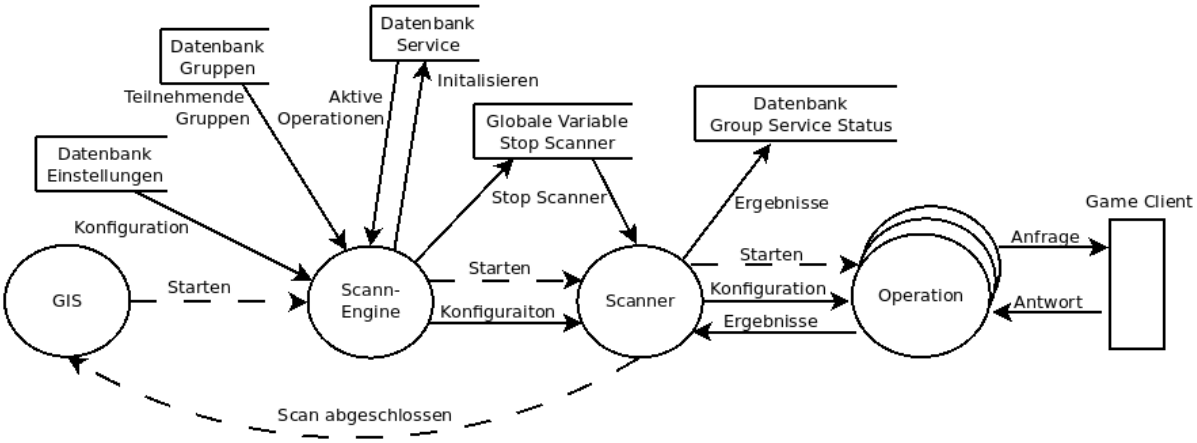

Abbildung 3.6: Datenfluss in der Scanner Komponente (Datenflussdiagramm)

Die im Datenflussdiagramm 3.6 zu sehenden, aber bisher nicht beschrieben Datenflüsse finden zwischen dem Webserver und dem Scanner oder einer Scan-Operation und dem Game Client statt. Administratoren können über den Webserver den Scanner an- und abschalten. Die Scan-Operationen fragt bei dem Game Client ihren überwachten Dienst / 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 unabhängig Komponenten, welche, 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 von einander und können leichter weiterentwickelt oder ersetzt werden.

Die Verwendung dieses Architekturmuster würde bei der aktuellen Aufgabenstellung zu hohen Aufwand für zu wenig Nutzen bedeuten.

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 / ein HTML Dokument eingebettet. Danach wird dieses zurück an den Nutzer 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 Nutzer sich mit einem extra angelegten Account am Flagshop anmelden und diesen verwenden dürfen. Dieses 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 dem 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 ersten ist es möglich den Client und Server unabhängig voneinander zu entwickeln, zu verändern und zu verbessern. Damit kann in Zukunft eine Iteration der Software einfacherer geschehen. Zweitens können die Studierenden eigene Clients programmieren, um mit der Anwendung zu interagieren.

3.5.3 Thin Webserver

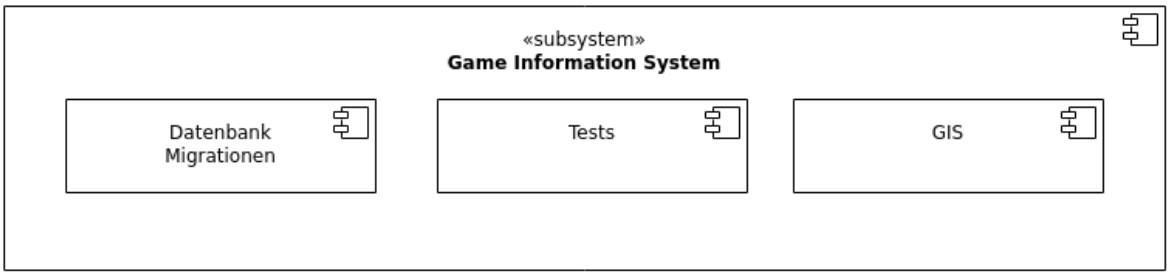

Abbildung 3.7: Rest Interface im Überblick (Komponentendiagramm)

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

Die Komponente *GIS* implementiert die gesamte Server Logik. In diesem Modul wird der eigentliche Thin Server, über das die Spieler und Betreuer mit der Anwendung interagieren können, implementiert.

In der Komponente Datenbank Migrationen sind Migrationsskripts hinterlegt, welche die Datenbank Iterationen festhalten. Diese Skripts sollen genutzt werden, um verwendete Datenbanken einfach auf den gleichen Soll-Zustand bringen zu können.

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

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

API

Um bei der Implementierung der API einem Standard / einem Vorgehen zu folgen, wird der de facto Standard für HTTP-APIs Represntational 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. Des Weiteren werden einheitliche Schnittstellen genutzt. Dazu müssen Standardmethoden und -repräsentationen genutzt werden.

Durch diese Anforderungen biete das REST Interface die Möglichkeiten alle zur Verfügung stehenden Ressourcen auf die gleiche Art und Weise zu verwalten. Mit dieser Herangehensweise wird die HTTP Methode GET nicht länger zur Veränderung oder Erschaffung von Ressourcen verwendet, sondern die dafür ausgelegten HTTP-Methoden. Die für die Verwendung

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

benötigten Daten werden auch nicht länger als Paramter der Anfrage beigefügt, sondern im Body der Anfrage übertragen.[Bei14]

Alle Routen werden mit dem Präfix */v1* versehen, um bei späteren Iterationen der API Kollisionen zu verhindern und die Nutzung der API v1 weiterhin zu 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.

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

(todo: Module / Komponenten des REST-Interface)

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 3.2: Übersicht über die zu implementierenden Routen

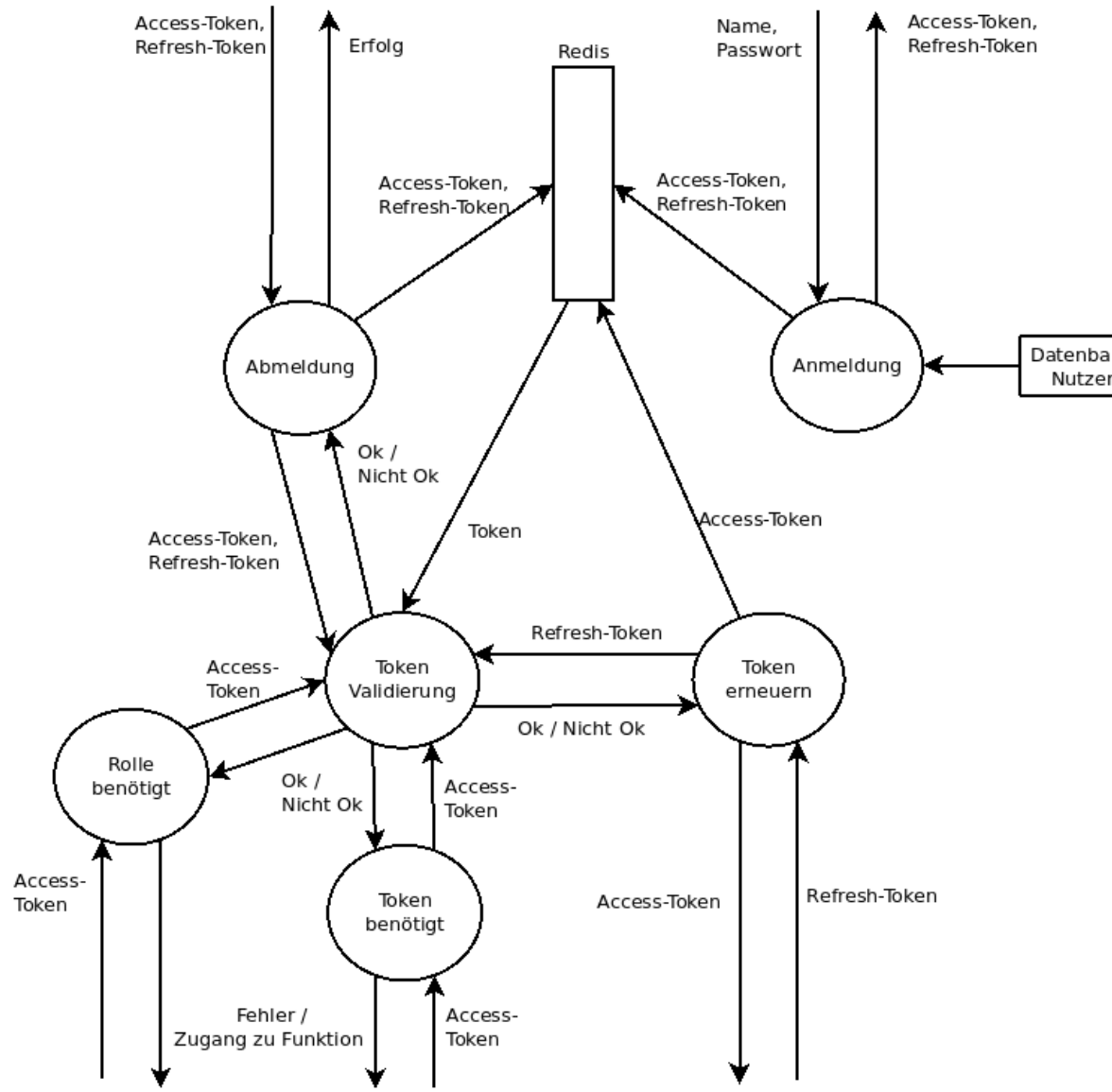

Abbildung 3.8: Datenfluss in der Security Komponente (Datenflussdiagramm)

Authentifizierung

Flaggengenierung - Änderung des Algo - Änderung des Vorgehens -> Generierung auf dem Server -> Versand an den Client (Secret nicht auf dem Client, Abfangen aller Flags möglich, Verschlüsselung bringt nix, da Nutzer root Rechte hat)

Challenges todo: Challenge Informationen und Lösung der Challenge über API, Darstellung / Inhalt der Challenge über anderen Webserver (da wo auch der Webclient ausgelieftert wird), da gedanke ist nur Daten zu übertragen und keine HTML Seiten etc.

3.5.4 Reverse Proxy

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

Der Reverse Proxy muss bei der Weiterleitung der Anfrage an das GIS die eigentliche IP-Adresse der Anfrage weiterleiten. Dieses wird benötigt um die IP-Adresse, und so die zugehörige Gruppe, bei der Registrierung und dem Login zu bestimmen. Die originale IP-Adresse der Anfrage soll in einen HTTP Header gespeichert werden.

In einen 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 Geheimnis kann das GIS entscheiden, ob die Anfrage über einen vertrauenswürdigen Proxy weitergeleitet oder ob ein Spoofing-Angriff durchgeführt worden ist. Dieses 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 kann dieser im Namen andere agieren und diesen beispielsweise Minuspunkte verschaffen.

3.6 Datenbank

- Daten in Verbindung setzen, keine redundanz - Ordnung der Daten - zentrale Speicherort - Atomare Operation -> keine unvollständigen Daten - Daten müssen erhalten bleiben - Abfragen besser optimiert als auf Dateien - Views zur Berechnung von Punkten

Daten haben Schema, Keine großen Datenmengen

-> relationale db

3.6.1 ASDF-Tabellen

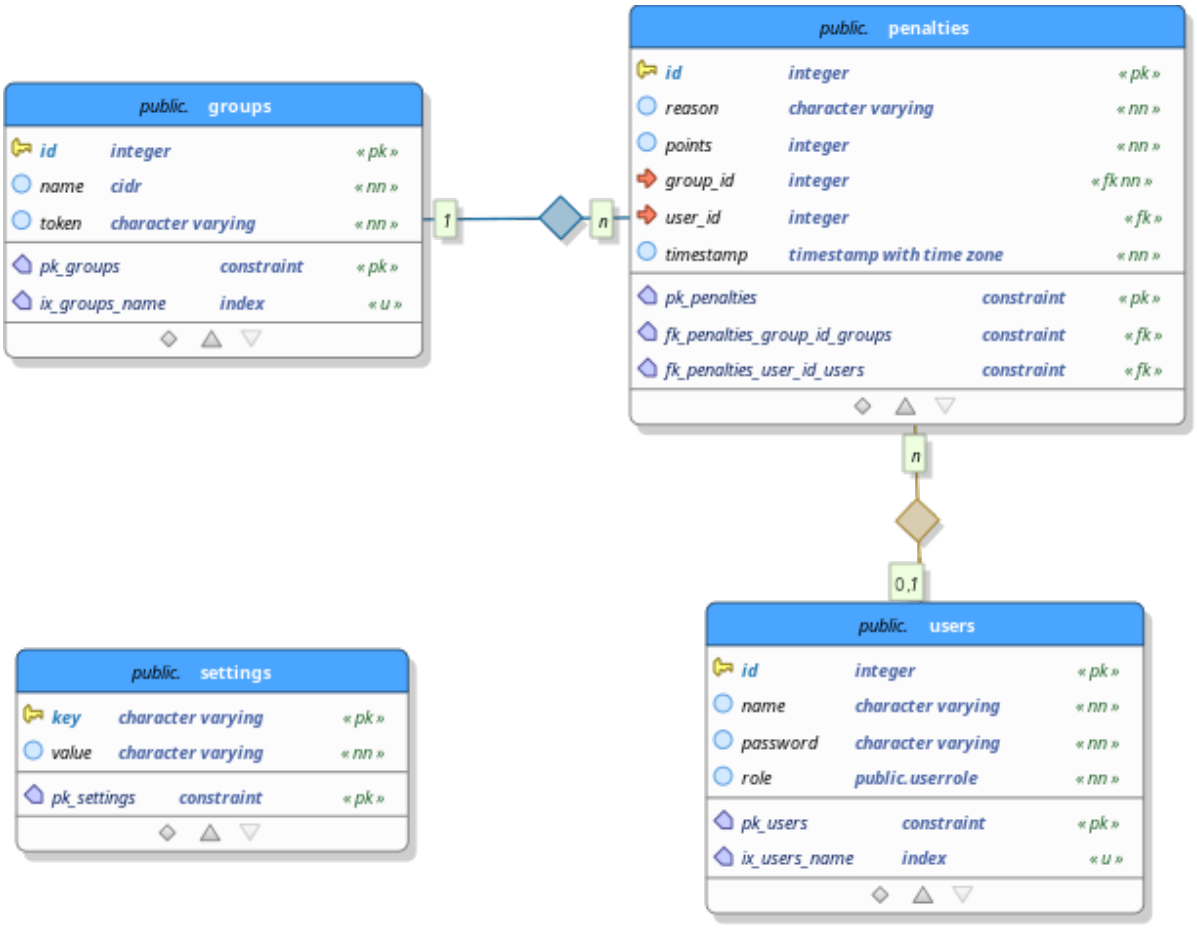

Abbildung 3.9: Ansicht der ASDF-Tabellen (ER-Diagramm)

In der Abbildung 3.9 ist neben der alleinstehenden Tabelle ***settings*** die Tabelle ***users*** sowie die n zu m Beziehung zwischen ***users*** und ***groups*** dargestellt. Die n zum m Beziehung wird über eine Referenztabelle (***penalties***) mit einer 1 zu n Beziehung zwischen ***groups*** und ***penalties*** und einer 0,1 zu n Beziehung zwischen ***users*** und ***penalties*** modelliert. In der n zu m Beziehung werden die ausgesprochen Strafen gespeichert. Jede Gruppe kann von mehreren Betreuern oder dem System mehrmals verwarnt werden.

Einstellungen

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

Nutzer

In der Tabelle ***users*** sind die Log-In Informationen der Administratoren, Betreuer und Spieler 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 Nutzers wird mithilfe eines Enums (*role*) (Liste von benannten Werten) definiert.

Strafen

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

3.6.2 Gruppen-Tabellen

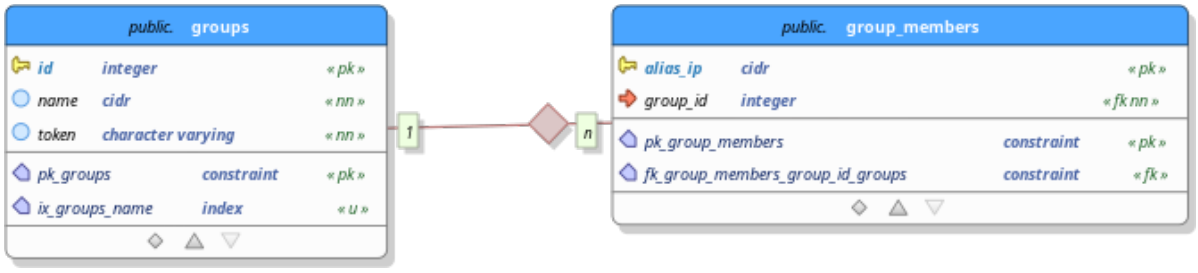

Abbildung 3.10: Ansicht der Gruppen-Tabellen (ER-Diagramm)

Wie in Abbildung 3.10 zusehen existiert zwischen der Tabelle ***groups*** und ***group_members*** eine 1 zu n Beziehung. Dies bedeutete, 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 Classless Internet Domain Routing Konvention (CIDR) abgelegt. Des Weiteren besitzt die ***groups*** Tabelle die Spalte *token*. In dieser wird der individuelle Token, welcher zur Flaggenerierung benötigt wird, abgespeichert. Dies ist nötig, damit der Server die selben Flags wie der GameClient generieren kann und die Betreuer die Richtigkeit verifizieren können.

Gruppenmitglieder

Die Tabelle ***group_members*** beinhaltet alle IP-Adressen, welche 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

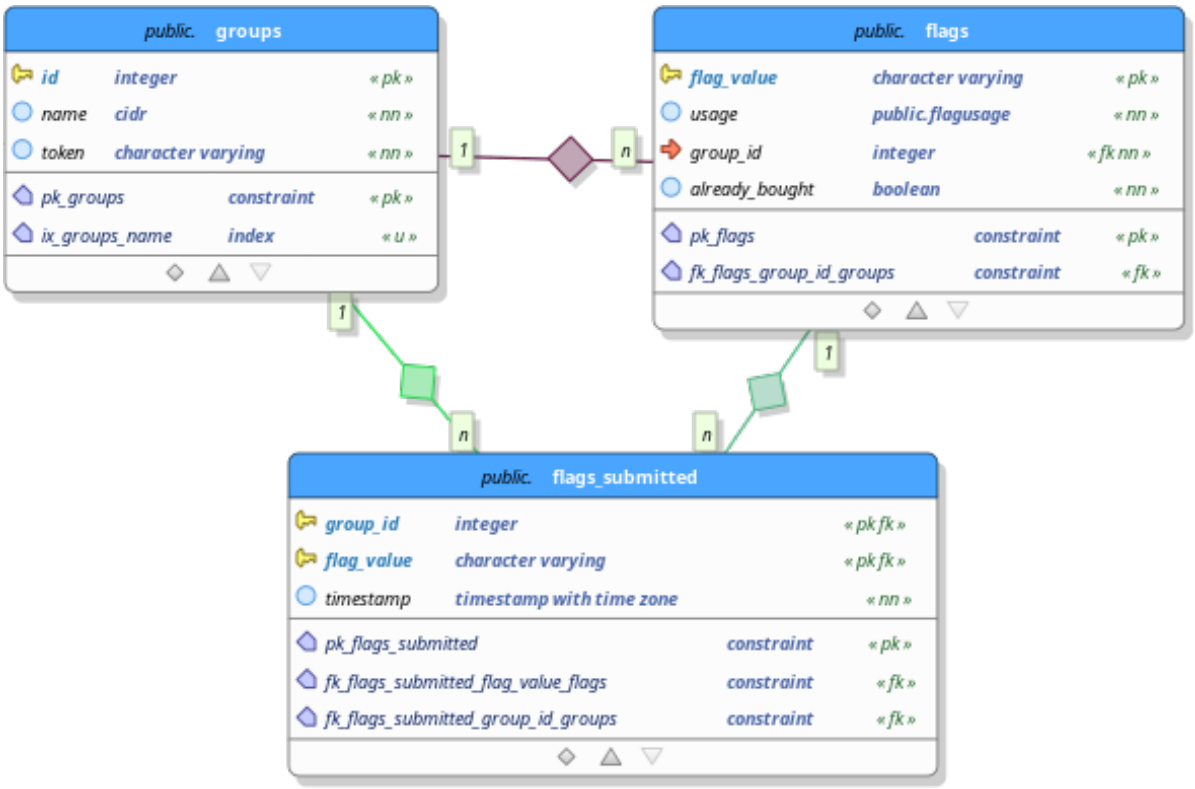

Abbildung 3.11: Ansicht der Flags-Tabellen (ER-Diagramm)

In der Abbildung 3.11 erkennbar ist die 1 zu n Beziehung zwischen der Tabelle ***groups*** und ***flags***. 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 zu m Beziehung zwischen ***groups*** und ***flags*** abgebildet. Diese wird in eine Verknüpfungstabelle namens ***flags_submitted*** mit zwei 1 zu n Beziehung zerlegt. Diese Tabelle enthält die abgegeben Flags.

Flags

Die während eines Spiels generierten und benötigten Flags werden in der Tabelle ***flags*** abgelegt. Hierzu wird der Wert der Flag als string in der Spalte *flag_value* festgehalten. Neben diesem wird Nutzungsart unter Zuhilfenahme eines Enums und die Gruppenzugehörigkeit als Referenz gespeichert. Neben diesem wird in der Spalte *already_bought* als bool vermerkt, ob diese Flag bereits im Flagshop erworben wurden ist. Diese Spalte besitzt nur Relevanz für die dementsprechenden Flagshop Flags.

Abgegebene Flags

Die durch die Studierenden abgegeben Flags werden in der Tabelle ***flags_submitted*** eingetragen. Auch wird der Zeitpunkt der Abgabe mitgespeichert, um eventuelle Betrüge aufzudecken. Die Spalten *group_id* und *flag_value* bilden zusammen den Primärschlüssel. Dieses ermöglicht, dass jede Gruppe jede Flagge und das eine Gruppe die selbe Flagge nicht mehrfach abgeben kann

3.6.4 Service-Tabellen

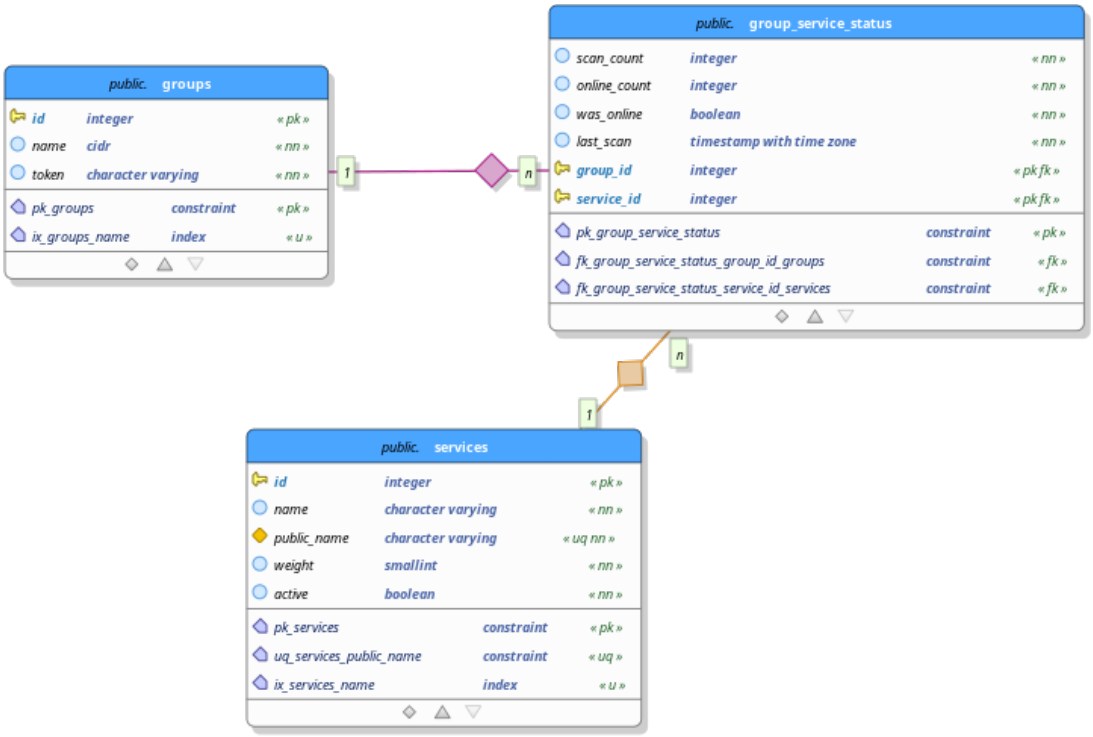

Abbildung 3.12: Ansicht der Service-Tabellen (ER-Diagramm)

In der Abbildung 3.12 ist die ***services*** Tabelle sowie die n zu m Beziehung zwischen ***groups*** und ***service*** zu sehen. Diese wird als Referenztabelle (***group_service_status***) mit zwei 1 zu n Beziehungen dargestellt.

Services

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

Servicestatus der Gruppen

In der Referenztabelle ***group_service_status*** wird das Ergebnis der Überprüfung der Schwachstellen 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 überwacht Schwachstelle ausgebessert und den überwachten Dienst online gehalten haben.

3.6.5 Flagshop-Tabellen

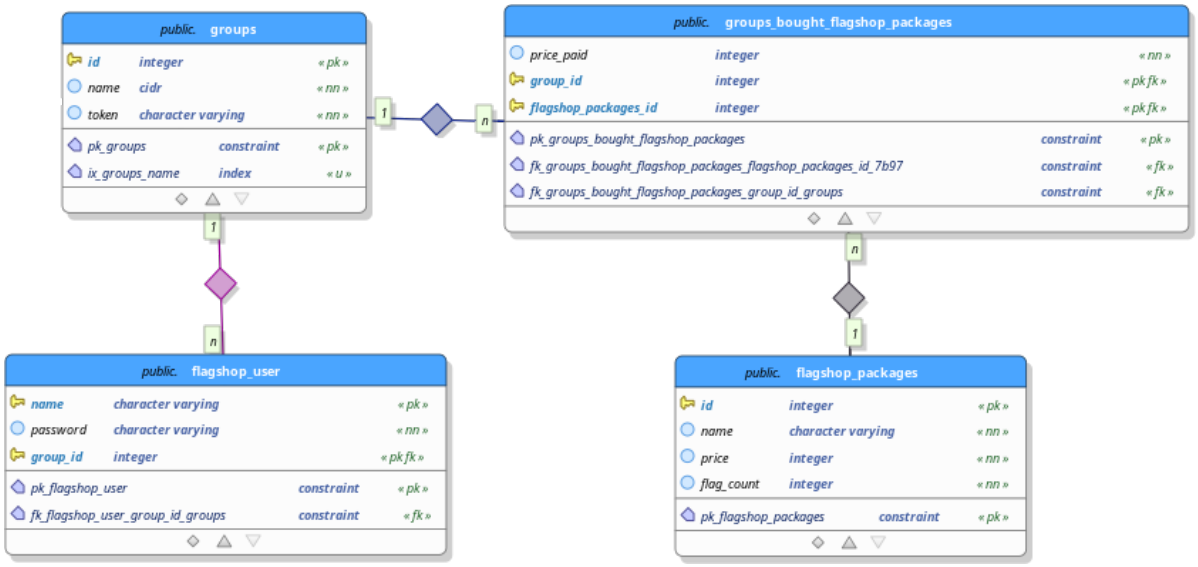

Abbildung 3.13: Ansicht der Flagshop-Tabellen (ER-Diagramm)

Die Abbildung 3.13 stellt die 1 zu n Beziehung zwischen den Tabellen ***groups*** und ***flagshop_user*** dar. Das heißt, dass jeder Gruppe mehrere Flagshop Benutzer zugehörig sein können, jeder Benutzer aber genau einer Gruppe angehört. Des Weiteren wird die n zu m Beziehung zwischen ***groups*** und ***flagshop_packages*** als Referenztabelle (***groups_bought_flagshop_packages***) mit zwei 1 zu n Beziehungen dargestellt. Diese beinhaltet die gekauften Flagshop Pakete pro Team.

Flagshop Nutzer

Für die Nutzung des Flagshops müssen die Gruppen einen oder mehrere Flagshop Nutzer anlegen. Diese werden in der Tabelle ***flagshop_user*** abgelegt. Hierfür wird 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 Nutzeraccounts mit demselben Namen anlegen.

Flagshop Pakete

Die Betreuer und Administratoren können Flagshop Pakete anlegen, welche durch Studierende kaufbar sind. Diese werden in der Tabelle ***flagshop_packages*** abgelegt und erhalten Informationen zum Preis (*price*) und der Anzahl der erhaltenen Flags (*flag_count*). Auch kann ein Name für das Paket über das Feld *name* angegeben werden.

Gekaufte Flagshop Pakete

Die durch die Gruppen gekauften Flagshop Pakete werden in der Tabelle ***group_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*) zusammen setzt.

3.6.6 Challenge-Tabellen

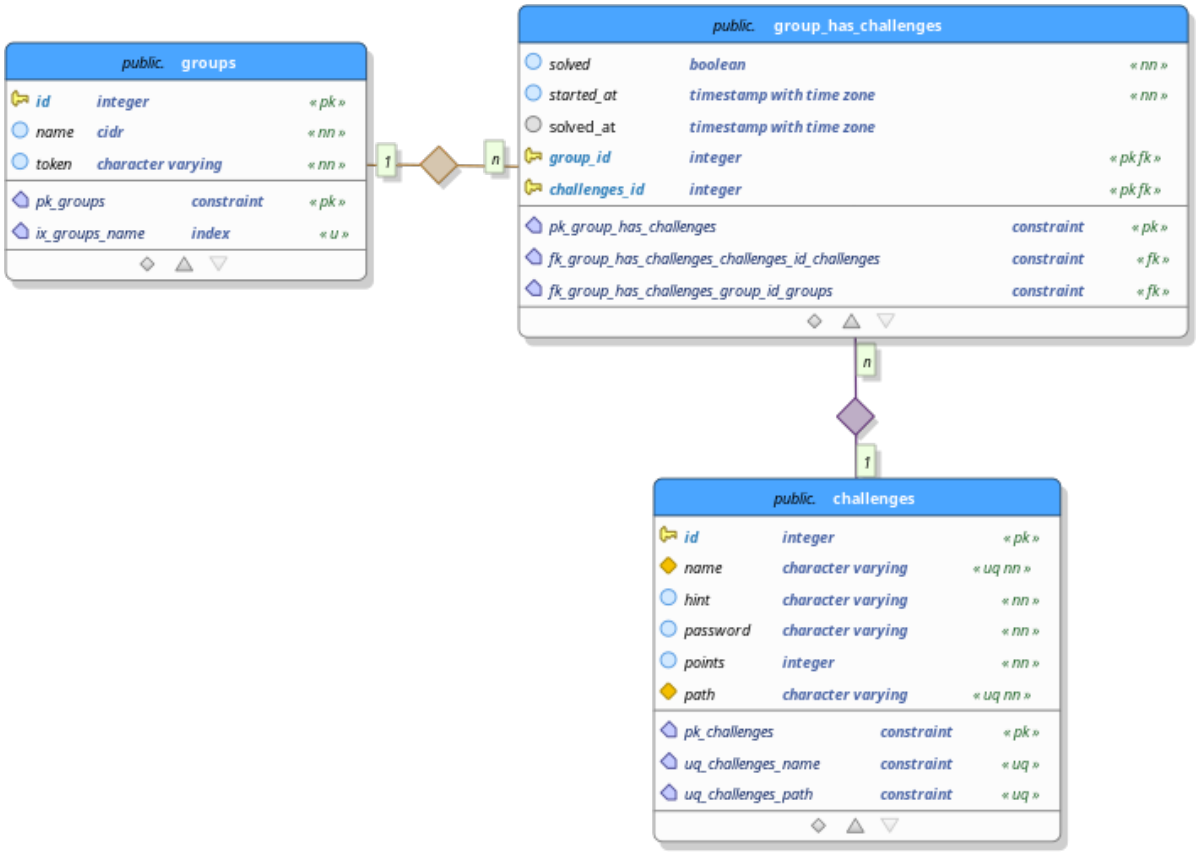

Abbildung 3.14: Ansicht der Service-Tabellen (ER-Diagramm)

In der Abbildung 3.14 ist die ***Challenge*** Tabelle und die n zum m Beziehung zwischen den Tabelle ***groups*** und ***challenges*** zusehen. Diese Beziehung ist als Verknüpfungstabelle (***group_has_challenges***) mit zwei 1 zu n Beziehung realisiert.

Challenges

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

Gestartete / Abgeschlossene Challenges

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

3.6.7 Weitere Tabellen

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

Wie in Abbildung 3.15 erkennbar besitzen die Tabellen keine Abhängigkeiten und Verbindungen zu anderen. 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 noch 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 Betreuern und Administratoren die Möglichkeit geboten werden Notizen und Hinweise zum Rahmen und Durchführung des Versuches zu geben. 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 *created_at* abgespeichert. Die Informationen zu einem Spiel sind im JSON Format in der Spalte *data* abgelegt.

3.7 Webclient

3.7.1 SPA vs MPA

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

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

Der Vorteil von MPAs ist die Optimierbarkeit für Suchmaschinen, das sogenannte SEO (Search Engine Optimization). Ein gutes SEO Rating sorgt dafür, dass die Webseite bei Suchmaschinen weit oben zu finden ist. Dies ist besonders wichtig bei Webseiten und Shops, welche um Kunden konkurrieren. Anzuführen sind hier diverse Webshops und Zeitungen.

[1] Beinhalten Regeln für die Darstellung von unter anderem Webseiten

Single Page Applications Die Single Page Applications, kurz SPA, stellt das genaue Gegenteil von MPA dar. Bei SPA 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 wird. Durch dieses Verfahren wird sicher gestellt, 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 selber 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) zurück zugreifen.

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

Die SEO Optimierung gestaltet sich schwieriger, da es sich um eine dynamische Anwendung handelt. Zur Nutzung von SPA muss im Browser JavaScript verfügbar und aktiviert sein.

Zusammenfassung Vor- und Nachteile

Für die Entwicklung der Anwendung entscheide ich mich für die Verwendung einer SPA. Dieses geschieht unter den Gesichtspunkten der Entkopplung zwischen Front- und Backend, der Performance der Anwendung und der Zukunftssicherheit, welche meiner Meinung nach für SPA besteht. Die Nachteile vom SPA betreffen meine Anwendung gering. 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. Einzig die Gefahr von XSS Attacken besteht, diese hoffe ich durch eine geeignete Wahl der Frontend Technologie zu reduzieren.[Mel20]

3.7.2 Mockups

(todo: Mockups einfügen)

3.8 Game Client

Der Gameclient bleibt durch diese Bachelorarbeit unberührt. Es ist jedoch eine Änderung an diesem vorzunehmen, damit er mit dem *Game Information System* kompatibel ist. Am Spiel können nur noch Gruppen teilnehmen, die ihre Teilnahme gegenüber dem GIS bekunden.

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

	SPA	MPA
Vorteile	• Sehr schnell, dank dynamischen nachladen • Entkoppelung zwischen Front- und Backend • Effizientes cachen von Daten	• MPA Architektur ist ausgereift • MPAs sind Entwickler freundlich, da ein kleiner Technologiestack benötigt wird • Ältere Browser werden unterstützt • SEO ist einfacher zu implementieren
Nachteile	• JavaScript muss im Browser verfügbar sein • Alte Browser werden nur teilweise unterstützt • Herausfordernde SEO Implementierung • Gefahr von XSS Attacken	• Anwendung sind weniger performant als MPAs • Front- und Backend haben eine starke Kopplung

Tabelle 3.3: Vor- und Nachteile SPA/MPA

Bei dieser Anmeldung erwartet das GIS einen Token, welcher bei der Generierung der Flags verwendet wird.

In derzeitige implementierte 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 verwendeten Funktion angepasst werden, damit die generierten Flags vergleichbar bleiben.

4 Technologien

4.1 Backend

Bei der Entwicklung des Webservers wird auf ein Backend-Webframework zurückgegriffen. Dieses erleichtert die Entwicklung und Wartung der Webanwendung in dem Werkzeuge und Bibliotheken zur Verfügung gestellt werden, welche allgemeine Aufgabe übernehmen. Zu diesen allgemeinen Aufgaben gehört das Verarbeiten von HTTP Anfragen, das Erstellen von HTTP Antworten und das Weiterleiten von Anfragen an die entsprechende Funktion. Bei der Implementierung der Logik müssen alle diese allgemeinen Aufgaben nicht mehr beachtet werden.

Um die Weiterentwicklung durch Studierende des Bachelor Informatik der Hochschule Niederrhein zu gewährleisten, werden nur Webframeworks, welche 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), welche 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 mehrere 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, so dass ein Nutzer keine oder wenige separate Pakete herunterladen muss.[Kuc]

Django liefert unter anderem Modulen für Caching, Logging, Versenden von E-Mails, RSS-Feeds, Pagination, Security und der automatischen Generierung von Administrationsseiten. [Dja] Durch den *Batteries Included* Ansatzes funktionieren die Komponenten reibungslos untereinander und sind mit dem Kern kompatibel. Des Weiteren sind die Dokumentationen der verschiedenen Module zentrale an einem Ort verfügbar.

Django wurde von einem Team entwickelt, welches ursprünglich Zeitungs-Websites erstellt und verwaltet hat. Dieses Team hat die gemeinsame Codebasis abstrahiert und in ein Framework überführt. Daher ist Django besonders, 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 zu 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 (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 ein solide Grundlage für alle Anwendung 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. Des Weiteren werden viele Module von Django nicht benötigt. Diese 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, welche alle einen ähnlichen Funktionsumfang mitbringen.

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

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

4.2.1 MySQL und MariaDB

MySQL ist das beliebteste Open-Source SQL Datenbankmanagementsystem, welches im Jahr 1995 veröffentliche 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. Des Weiteren ist es auf diversen Plattformen einsetzbar und für große Datenmengen optimiert. [Ora20b]

MariaDB ist eine Abspaltung von MySQL. Diese 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 ohne Lizenzierung nutzbar. Es wurde als universitäres Projekt an der University of California at Berkeley Computer Science Department gestartet. Es sind neben den SQL92 und SQL99 Standard 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, welche nicht aufgesetzt oder verwaltet werden muss. SQLite speichert die Daten in einer gewöhnlichen Datei, welche 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 von PostgreSQL im Vergleich zu SQLite kann durch die Nutzung eines Dockerimages reduziert werden. Des Weiteren haben sich die Studierenden mit der PostgreSQL Datenbank in der Veranstaltung *Datenbanksysteme* beschäftigt und im EZS-Labor besitzen bereits Mitarbeiter Kenntnisse von PostgreSQL.

4.3 Frontend

REACT VS ANGULAR VS VUE

aus andere ba quellen nehmen eigene schlussfolgerung

5 Realisierung

6 Zusammenfassung & Aussicht

Anhang

```
-- Database generated with pgModeler (PostgreSQL Database Modeler).
-- pgModeler  version: 0.9.2
-- PostgreSQL version: 12.0
-- Project Site: pgmodeler.io
-- Model Author: ---


-- Database creation must be done outside a multicommand file.
-- These commands were put in this file only as a convenience.
-- -- object: postgres | type: DATABASE --
-- -- DROP DATABASE IF EXISTS postgres;
-- CREATE DATABASE postgres
--      ENCODING = 'UTF8'
--      LC_COLLATE = 'en_US.utf8'
--      LC_CTYPE = 'en_US.utf8'
--      TABLESPACE = pg_default
--      OWNER = postgres;
-- -- ddl-end --
-- COMMENT ON DATABASE postgres IS E'default administrative connection datab
-- -- ddl-end --
--

-- object: public.alembic_version | type: TABLE --
-- DROP TABLE IF EXISTS public.alembic_version CASCADE;
CREATE TABLE public.alembic_version (
        version_num character varying(32) NOT NULL,
        CONSTRAINT alembic_version_pkc PRIMARY KEY (version_num)

);
-- ddl-end --
-- ALTER TABLE public.alembic_version OWNER TO postgres;
-- ddl-end --

-- object: public.backups_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.backups_id_seq CASCADE;
CREATE SEQUENCE public.backups_id_seq
```

```
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.backups_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.backups | type: TABLE --
-- DROP TABLE IF EXISTS public.backups CASCADE;
CREATE TABLE public.backups (
        id integer NOT NULL DEFAULT nextval('public.backups_id_seq'::
        created_at timestamp with time zone NOT NULL DEFAULT now(),
        data json NOT NULL,
        CONSTRAINT pk_backups PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.backups OWNER TO postgres;
-- ddl-end --

-- object: public.flagshop_packages_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.flagshop_packages_id_seq CASCADE;
CREATE SEQUENCE public.flagshop_packages_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.flagshop_packages_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.flagshop_packages | type: TABLE --
-- DROP TABLE IF EXISTS public.flagshop_packages CASCADE;
CREATE TABLE public.flagshop_packages (
        id integer NOT NULL DEFAULT nextval('public.flagshop_packages_
        name character varying NOT NULL,
        price integer NOT NULL,
```

```
        flag_count integer NOT NULL,
        CONSTRAINT pk_flagshop_packages PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.flagshop_packages OWNER TO postgres;
-- ddl-end --

-- object: public.groups_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.groups_id_seq CASCADE;
CREATE SEQUENCE public.groups_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.groups_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.groups | type: TABLE --
-- DROP TABLE IF EXISTS public.groups CASCADE;
CREATE TABLE public.groups (
        id integer NOT NULL DEFAULT nextval('public.groups_id_seq'::regclass
        name cidr NOT NULL,
        token character varying NOT NULL,
        CONSTRAINT pk_groups PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.groups OWNER TO postgres;
-- ddl-end --

-- object: ix_groups_name | type: INDEX --
-- DROP INDEX IF EXISTS public.ix_groups_name CASCADE;
CREATE UNIQUE INDEX ix_groups_name ON public.groups
        USING btree
        (
          name
        )
        WITH (FILLFACTOR = 90);
-- ddl-end --
```

```
-- object: public.services_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.services_id_seq CASCADE;
CREATE SEQUENCE public.services_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.services_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.services | type: TABLE --
-- DROP TABLE IF EXISTS public.services CASCADE;
CREATE TABLE public.services (
        id integer NOT NULL DEFAULT nextval('public.services_id_seq':
        name character varying NOT NULL,
        public_name character varying NOT NULL,
        weight smallint NOT NULL,
        active boolean NOT NULL,
        CONSTRAINT pk_services PRIMARY KEY (id),
        CONSTRAINT uq_services_public_name UNIQUE (public_name)

);
-- ddl-end --
-- ALTER TABLE public.services OWNER TO postgres;
-- ddl-end --

-- object: ix_services_name | type: INDEX --
-- DROP INDEX IF EXISTS public.ix_services_name CASCADE;
CREATE UNIQUE INDEX ix_services_name ON public.services
        USING btree
        (
          name
        )
        WITH (FILLFACTOR = 90);
-- ddl-end --

-- object: public.userrole | type: TYPE --
-- DROP TYPE IF EXISTS public.userrole CASCADE;
CREATE TYPE public.userrole AS
```

```
 ENUM ('admin','supervisor','player');
-- ddl-end --
-- ALTER TYPE public.userrole OWNER TO postgres;
-- ddl-end --

-- object: public.users_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.users_id_seq CASCADE;
CREATE SEQUENCE public.users_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.users_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.users | type: TABLE --
-- DROP TABLE IF EXISTS public.users CASCADE;
CREATE TABLE public.users (
        id integer NOT NULL DEFAULT nextval('public.users_id_seq'::regclass)
        name character varying NOT NULL,
        password character varying NOT NULL,
        role public.userrole NOT NULL,
        CONSTRAINT pk_users PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.users OWNER TO postgres;
-- ddl-end --

-- object: ix_users_name | type: INDEX --
-- DROP INDEX IF EXISTS public.ix_users_name CASCADE;
CREATE UNIQUE INDEX ix_users_name ON public.users
        USING btree
        (
          name
        )
        WITH (FILLFACTOR = 90);
-- ddl-end --

-- object: public.flagusage | type: TYPE --
```

```
-- DROP TYPE IF EXISTS public.flagusage CASCADE;
CREATE TYPE public.flagusage AS
 ENUM ('on_client','flagshop_registration','flagshop_buy');
-- ddl-end --
-- ALTER TYPE public.flagusage OWNER TO postgres;
-- ddl-end --

-- object: public.flags | type: TABLE --
-- DROP TABLE IF EXISTS public.flags CASCADE;
CREATE TABLE public.flags (
        flag_value character varying NOT NULL,
        usage public.flagusage NOT NULL,
        group_id integer NOT NULL,
        already_bought boolean NOT NULL DEFAULT false,
        CONSTRAINT pk_flags PRIMARY KEY (flag_value)

);
-- ddl-end --
-- ALTER TABLE public.flags OWNER TO postgres;
-- ddl-end --

-- object: public.flagshop_user | type: TABLE --
-- DROP TABLE IF EXISTS public.flagshop_user CASCADE;
CREATE TABLE public.flagshop_user (
        name character varying NOT NULL,
        password character varying NOT NULL,
        group_id integer NOT NULL,
        CONSTRAINT pk_flagshop_user PRIMARY KEY (name,group_id)

);
-- ddl-end --
-- ALTER TABLE public.flagshop_user OWNER TO postgres;
-- ddl-end --

-- object: public.group_members | type: TABLE --
-- DROP TABLE IF EXISTS public.group_members CASCADE;
CREATE TABLE public.group_members (
        alias_ip cidr NOT NULL,
        group_id integer NOT NULL,
        CONSTRAINT pk_group_members PRIMARY KEY (alias_ip)

);
-- ddl-end --
-- ALTER TABLE public.group_members OWNER TO postgres;
```

```
-- ddl-end --

-- object: public.flags_submitted | type: TABLE --
-- DROP TABLE IF EXISTS public.flags_submitted CASCADE;
CREATE TABLE public.flags_submitted (
        group_id integer NOT NULL,
        flag_value character varying NOT NULL,
        "timestamp" timestamp with time zone NOT NULL DEFAULT now(),
        CONSTRAINT pk_flags_submitted PRIMARY KEY (group_id,flag_value)

);
-- ddl-end --
-- ALTER TABLE public.flags_submitted OWNER TO postgres;
-- ddl-end --

-- object: public.discover_points | type: VIEW --
-- DROP VIEW IF EXISTS public.discover_points CASCADE;
CREATE VIEW public.discover_points
AS

SELECT flags_submitted.group_id,
    count(flags_submitted.flag_value) AS points
   FROM (flags_submitted
     JOIN flags ON (((flags_submitted.flag_value)::text = (flags.flag_value)
  WHERE (flags_submitted.group_id = flags.group_id)
  GROUP BY flags_submitted.group_id;
-- ddl-end --
-- ALTER VIEW public.discover_points OWNER TO postgres;
-- ddl-end --

-- object: public.offence_points | type: VIEW --
-- DROP VIEW IF EXISTS public.offence_points CASCADE;
CREATE VIEW public.offence_points
AS

SELECT flags_submitted.group_id,
    count(flags_submitted.flag_value) AS points
   FROM (flags_submitted
     JOIN flags ON (((flags_submitted.flag_value)::text = (flags.flag_value)
  WHERE (flags_submitted.group_id <> flags.group_id)
  GROUP BY flags_submitted.group_id;
-- ddl-end --
-- ALTER VIEW public.offence_points OWNER TO postgres;
-- ddl-end --
```

```
-- object: public.defence_points | type: VIEW --
-- DROP VIEW IF EXISTS public.defence_points CASCADE;
CREATE VIEW public.defence_points
AS

SELECT flags.group_id,
    count(DISTINCT flags.flag_value) AS points
   FROM (flags
     JOIN flags_submitted ON (((flags.flag_value)::text = (flags_subm
  WHERE (flags.group_id <> flags_submitted.group_id)
  GROUP BY flags.group_id;
-- ddl-end --
-- ALTER VIEW public.defence_points OWNER TO postgres;
-- ddl-end --

-- object: public.penalties_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.penalties_id_seq CASCADE;
CREATE SEQUENCE public.penalties_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.penalties_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.penalties | type: TABLE --
-- DROP TABLE IF EXISTS public.penalties CASCADE;
CREATE TABLE public.penalties (
        id integer NOT NULL DEFAULT nextval('public.penalties_id_seq'
        reason character varying NOT NULL,
        points integer NOT NULL,
        group_id integer NOT NULL,
        user_id integer,
        "timestamp" timestamp with time zone NOT NULL DEFAULT now(),
        CONSTRAINT pk_penalties PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.penalties OWNER TO postgres;
```

```
-- ddl-end --

-- object: public.groups_bought_flagshop_packages | type: TABLE --
-- DROP TABLE IF EXISTS public.groups_bought_flagshop_packages CASCADE;
CREATE TABLE public.groups_bought_flagshop_packages (
        price_paid integer NOT NULL,
        group_id integer NOT NULL,
        flagshop_packages_id integer NOT NULL,
        CONSTRAINT pk_groups_bought_flagshop_packages PRIMARY KEY (group_id,

);
-- ddl-end --
-- ALTER TABLE public.groups_bought_flagshop_packages OWNER TO postgres;
-- ddl-end --

-- object: public.group_service_status | type: TABLE --
-- DROP TABLE IF EXISTS public.group_service_status CASCADE;
CREATE TABLE public.group_service_status (
        scan_count integer NOT NULL,
        online_count integer NOT NULL,
        was_online boolean NOT NULL,
        last_scan timestamp with time zone NOT NULL DEFAULT now(),
        group_id integer NOT NULL,
        service_id integer NOT NULL,
        CONSTRAINT pk_group_service_status PRIMARY KEY (group_id,service_id)

);
-- ddl-end --
-- ALTER TABLE public.group_service_status OWNER TO postgres;
-- ddl-end --

-- object: public.challenges_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.challenges_id_seq CASCADE;
CREATE SEQUENCE public.challenges_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.challenges_id_seq OWNER TO postgres;
-- ddl-end --
```

```
-- object: public.challenges | type: TABLE --
-- DROP TABLE IF EXISTS public.challenges CASCADE;
CREATE TABLE public.challenges (
        id integer NOT NULL DEFAULT nextval('public.challenges_id_seq
        name character varying NOT NULL,
        hint character varying NOT NULL,
        password character varying NOT NULL,
        points integer NOT NULL,
        path character varying NOT NULL,
        CONSTRAINT pk_challenges PRIMARY KEY (id),
        CONSTRAINT uq_challenges_name UNIQUE (name),
        CONSTRAINT uq_challenges_path UNIQUE (path)

);
-- ddl-end --
-- ALTER TABLE public.challenges OWNER TO postgres;
-- ddl-end --

-- object: public.group_has_challenges | type: TABLE --
-- DROP TABLE IF EXISTS public.group_has_challenges CASCADE;
CREATE TABLE public.group_has_challenges (
        solved boolean NOT NULL DEFAULT false,
        started_at timestamp with time zone NOT NULL DEFAULT now(),
        solved_at timestamp with time zone,
        group_id integer NOT NULL,
        challenges_id integer NOT NULL,
        CONSTRAINT pk_group_has_challenges PRIMARY KEY (group_id,chal

);
-- ddl-end --
-- ALTER TABLE public.group_has_challenges OWNER TO postgres;
-- ddl-end --

-- object: public.logaction | type: TYPE --
-- DROP TYPE IF EXISTS public.logaction CASCADE;
CREATE TYPE public.logaction AS
 ENUM ('game_created','game_started','game_paused','game_resumed','ga
-- ddl-end --
-- ALTER TYPE public.logaction OWNER TO postgres;
-- ddl-end --

-- object: public.logs_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.logs_id_seq CASCADE;
```

```
CREATE SEQUENCE public.logs_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.logs_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.logs | type: TABLE --
-- DROP TABLE IF EXISTS public.logs CASCADE;
CREATE TABLE public.logs (
        id integer NOT NULL DEFAULT nextval('public.logs_id_seq'::regclass),
        action public.logaction NOT NULL,
        data json NOT NULL,
        created_at timestamp with time zone NOT NULL DEFAULT now(),
        CONSTRAINT pk_logs PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.logs OWNER TO postgres;
-- ddl-end --

-- object: public.logs_old_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.logs_old_id_seq CASCADE;
CREATE SEQUENCE public.logs_old_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.logs_old_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.logs_old | type: TABLE --
-- DROP TABLE IF EXISTS public.logs_old CASCADE;
CREATE TABLE public.logs_old (
        id integer NOT NULL DEFAULT nextval('public.logs_old_id_seq'::regcla
```

```
        action public.logaction NOT NULL,
        data json NOT NULL,
        created_at timestamp with time zone NOT NULL DEFAULT now(),
        CONSTRAINT pk_logs_old PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.logs_old OWNER TO postgres;
-- ddl-end --

-- object: public.notes_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS public.notes_id_seq CASCADE;
CREATE SEQUENCE public.notes_id_seq
        INCREMENT BY 1
        MINVALUE 1
        MAXVALUE 2147483647
        START WITH 1
        CACHE 1
        NO CYCLE
        OWNED BY NONE;
-- ddl-end --
-- ALTER SEQUENCE public.notes_id_seq OWNER TO postgres;
-- ddl-end --

-- object: public.notes | type: TABLE --
-- DROP TABLE IF EXISTS public.notes CASCADE;
CREATE TABLE public.notes (
        id integer NOT NULL DEFAULT nextval('public.notes_id_seq'::re
        title character varying NOT NULL,
        text character varying NOT NULL,
        CONSTRAINT pk_notes PRIMARY KEY (id)

);
-- ddl-end --
-- ALTER TABLE public.notes OWNER TO postgres;
-- ddl-end --

-- object: public.settings | type: TABLE --
-- DROP TABLE IF EXISTS public.settings CASCADE;
CREATE TABLE public.settings (
        key character varying NOT NULL,
        value character varying NOT NULL,
        CONSTRAINT pk_settings PRIMARY KEY (key)
```

```
);
-- ddl-end --
-- ALTER TABLE public.settings OWNER TO postgres;
-- ddl-end --

-- object: public.challenge_points | type: VIEW --
-- DROP VIEW IF EXISTS public.challenge_points CASCADE;
CREATE VIEW public.challenge_points
AS

SELECT group_has_challenges.group_id,
    sum(challenges.points) AS total_points
   FROM (group_has_challenges
     JOIN challenges ON ((group_has_challenges.challenges_id = challenges.id
  WHERE (group_has_challenges.solved = true)
  GROUP BY group_has_challenges.group_id;
-- ddl-end --
-- ALTER VIEW public.challenge_points OWNER TO postgres;
-- ddl-end --

-- object: public.flagshop_points | type: VIEW --
-- DROP VIEW IF EXISTS public.flagshop_points CASCADE;
CREATE VIEW public.flagshop_points
AS

SELECT groups_bought_flagshop_packages.group_id,
    sum(groups_bought_flagshop_packages.price_paid) AS points_spend
   FROM groups_bought_flagshop_packages
  GROUP BY groups_bought_flagshop_packages.group_id;
-- ddl-end --
-- ALTER VIEW public.flagshop_points OWNER TO postgres;
-- ddl-end --

-- object: public.penalty_points | type: VIEW --
-- DROP VIEW IF EXISTS public.penalty_points CASCADE;
CREATE VIEW public.penalty_points
AS

SELECT penalties.group_id,
    sum(penalties.points) AS total_penalty
   FROM penalties
  GROUP BY penalties.group_id;
-- ddl-end --
-- ALTER VIEW public.penalty_points OWNER TO postgres;
```

```
-- ddl-end --

-- object: public.group_service_points | type: VIEW --
-- DROP VIEW IF EXISTS public.group_service_points CASCADE;
CREATE VIEW public.group_service_points
AS

SELECT group_service_status.group_id,
    group_service_status.service_id,
    (abs(round((((group_service_status.online_count - group_service_s
    (round((COALESCE(((100 / group_service_status.scan_count) * group
   FROM (group_service_status
     JOIN services ON ((group_service_status.service_id = services.id
-- ddl-end --
-- ALTER VIEW public.group_service_points OWNER TO postgres;
-- ddl-end --

-- object: public.group_service_online_status | type: VIEW --
-- DROP VIEW IF EXISTS public.group_service_online_status CASCADE;
CREATE VIEW public.group_service_online_status
AS

SELECT group_service_status.group_id,
    group_service_status.service_id,
    group_service_status.was_online
   FROM group_service_status;
-- ddl-end --
-- ALTER VIEW public.group_service_online_status OWNER TO postgres;
-- ddl-end --

-- object: public.total_points | type: VIEW --
-- DROP VIEW IF EXISTS public.total_points CASCADE;
CREATE VIEW public.total_points
AS

SELECT grp.id,
    grp.name,
    (COALESCE(cha.total_points, ((0)::smallint)::bigint))::smallint A
    (COALESCE(dis.points, ((0)::smallint)::bigint))::smallint AS disc
    (COALESCE(ofe.points, ((0)::smallint)::bigint))::smallint AS offe
    (COALESCE(def.points, ((0)::smallint)::bigint))::smallint AS defe
    (COALESCE(tmp.service_points, ((0)::smallint)::bigint))::smallint
    (COALESCE(fla.points_spend, ((0)::smallint)::bigint))::smallint A
    (COALESCE(pen.total_penalty, ((0)::smallint)::bigint))::smallint
```

```
    ((((((COALESCE(cha.total_points, ((0)::smallint)::bigint))::smallint +
   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)));
-- ddl-end --
-- ALTER VIEW public.total_points OWNER TO postgres;
-- ddl-end --

-- object: fk_flags_group_id_groups | type: CONSTRAINT --
-- ALTER TABLE public.flags DROP CONSTRAINT IF EXISTS fk_flags_group_id_grou
ALTER TABLE public.flags ADD CONSTRAINT fk_flags_group_id_groups FOREIGN KEY
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_flagshop_user_group_id_groups | type: CONSTRAINT --
-- ALTER TABLE public.flagshop_user DROP CONSTRAINT IF EXISTS fk_flagshop_us
ALTER TABLE public.flagshop_user ADD CONSTRAINT fk_flagshop_user_group_id_gr
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_group_members_group_id_groups | type: CONSTRAINT --
-- ALTER TABLE public.group_members DROP CONSTRAINT IF EXISTS fk_group_membe
ALTER TABLE public.group_members ADD CONSTRAINT fk_group_members_group_id_gr
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_flags_submitted_flag_value_flags | type: CONSTRAINT --
-- ALTER TABLE public.flags_submitted DROP CONSTRAINT IF EXISTS fk_flags_sub
ALTER TABLE public.flags_submitted ADD CONSTRAINT fk_flags_submitted_flag_va
REFERENCES public.flags (flag_value) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --
```

```
-- object: fk_flags_submitted_group_id_groups | type: CONSTRAINT --
-- ALTER TABLE public.flags_submitted DROP CONSTRAINT IF EXISTS fk_fla
ALTER TABLE public.flags_submitted ADD CONSTRAINT fk_flags_submitted_g
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_penalties_group_id_groups | type: CONSTRAINT --
-- ALTER TABLE public.penalties DROP CONSTRAINT IF EXISTS fk_penalties
ALTER TABLE public.penalties ADD CONSTRAINT fk_penalties_group_id_grou
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE CASCADE ON UPDATE CASCADE;
-- ddl-end --

-- object: fk_penalties_user_id_users | type: CONSTRAINT --
-- ALTER TABLE public.penalties DROP CONSTRAINT IF EXISTS fk_penalties
ALTER TABLE public.penalties ADD CONSTRAINT fk_penalties_user_id_users
REFERENCES public.users (id) MATCH SIMPLE
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: fk_groups_bought_flagshop_packages_flagshop_packages_id_7b9
-- ALTER TABLE public.groups_bought_flagshop_packages DROP CONSTRAINT
ALTER TABLE public.groups_bought_flagshop_packages ADD CONSTRAINT fk_
REFERENCES public.flagshop_packages (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_groups_bought_flagshop_packages_group_id_groups | type:
-- ALTER TABLE public.groups_bought_flagshop_packages DROP CONSTRAINT
ALTER TABLE public.groups_bought_flagshop_packages ADD CONSTRAINT fk_
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_group_service_status_group_id_groups | type: CONSTRAINT
-- ALTER TABLE public.group_service_status DROP CONSTRAINT IF EXISTS
ALTER TABLE public.group_service_status ADD CONSTRAINT fk_group_servi
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_group_service_status_service_id_services | type: CONSTR
```

```
-- ALTER TABLE public.group_service_status DROP CONSTRAINT IF EXISTS fk_grou
ALTER TABLE public.group_service_status ADD CONSTRAINT fk_group_service_stat
REFERENCES public.services (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_group_has_challenges_challenges_id_challenges | type: CONSTRAI
-- ALTER TABLE public.group_has_challenges DROP CONSTRAINT IF EXISTS fk_grou
ALTER TABLE public.group_has_challenges ADD CONSTRAINT fk_group_has_challeng
REFERENCES public.challenges (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: fk_group_has_challenges_group_id_groups | type: CONSTRAINT --
-- ALTER TABLE public.group_has_challenges DROP CONSTRAINT IF EXISTS fk_grou
ALTER TABLE public.group_has_challenges ADD CONSTRAINT fk_group_has_challeng
REFERENCES public.groups (id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --
```

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

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.

[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.

[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.

[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).

[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).

[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).

[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_Kompetenzzentr 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).

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

[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).

[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).

[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).

[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).

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

[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.