INSTITUTE FOR NETWORKED SOLUTIONS

OST Ostschweizer Fachhochschule

Bug Chaser

Cloud Native Bug Bounty Plattform

Bachelorarbeit

Studiengang Informatik
Themengebiet Sicherheit
OST - Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Frühjahrssemster 2022

Autoren:	Janis Wolf, Marius Zindel
Version:	16. Juni 2022
Ausgabe:	E-Prints
Betreuerin:	Prof. Dr. Nathalie Weiler
Experte:	Christian Birchler
Gegenleserin:	Prof. Dr. Mitra Purandare

Abstract

Im Softwareentwicklungsprozess können Applikationen auf verschiedenste Weisen getestet werden. Eine mögliche Variante ist es, Software mittels Bug Bounty Programmen auf Schwachstellen und Fehler zu testen. Unternehmen und Organisationen erhalten so die Möglichkeit, selbst entwickelte Software von externen IT-Sicherheitsexperten zu prüfen. Auch die Fachhochschule OST, sowie deren Institute, setzen verschiedene interne Applikationen ein, welche von den Studierenden sowie den Mitarbeitenden täglich genutzt werden.

Ziel dieser Arbeit war es, eine Bug Bounty Plattform für die Fachhochschule zu entwickeln. Diese soll es einerseits ermöglichen, in Zukunft eigene, interne Applikationen auf Schwachstellen zu testen, andererseits soll die Plattform aber auch für die Ausbildung eingesetzt werden können. Im Modul Secure Software am Campus Rapperswil-Jona wird speziell unterrichtet, wie Applikationen sicher entwickelt werden. Ein Teil dieser Ausbildung setzt ebenfalls auf Bug Bounty Programme, welcher durch die entwickelte Plattform unterstützt werden soll.

Auf Basis von Recherchen zu bestehenden Lösungen und Technologien wurden Anforderungen an die Plattform aufgenommen. Des Weiteren sollte die Applikation kostengünstig im Betrieb sein. Zum Ende der Construction Phase wurde die während der Arbeit umgesetzten Plattform sowie das dahinterstehende Konzept mit Studierenden an der OST anhand einem Bug Bounty Programm mit einer fehlerbehafteten Software validiert.

Als Resultat der Arbeit ist die Bug Chaser Plattform entstanden. Eine Cloud Native Webapplikation, welche vollständig in der Amazon Web Services Cloud betrieben wird. Interne wie auch externe Organisationen erhalten die Möglichkeit, private oder öffentliche Bug Bounty Programme auf der Plattform zu betreiben. Hunter können nach Schwachstellen suchen und diese auf der Plattform rapportieren. Im Gegenzug können Belohnungen ausbezahlt werden. Während der Validierung zeigte es sich, dass die Applikation bereits eingesetzt werden kann. Weitere Funktionalitäten können in einer Folgearbeit erweitert werden.

Inhaltsverzeichnis

KAPITEL 1

Management Summary

1.1. Ausgangslage

1.1.1. Einführung

Professionelle Software-Entwicklungsunternehmen setzen auf öffentliche wie auch private Bug Bounty Programme, um ihre Applikationen zu testen. Auch Regierungsorganisationen nutzen solche Programme vermehrt. Dabei untersuchen externe IT-Sicherheitsexperten gezielt nach sicherheitskritischen Schwachstellen in Software. Ziel ist es, dass ethische Hacker eine Schwachstelle melden, bevor diese ausgenutzt werden kann und ein grösserer Schaden für das Unternehmen entsteht. Bei erfolgreichem Melden von Schwachstellen in der Software wird üblicherweise als Gegenleistung eine Kompensation ausbezahlt.

1.1.2. Problemstellung

Auch die Fachhochschule OST setzt auf eigenentwickelte Software, welche von den Studierenden sowie von den internen Mitarbeitern verwendet werden. Zum bisherigen Stand setzt die Fachhochschule noch nicht auf Bug Bounty Programme. Externe Bug Bounty Plattformen, auf welchen solche Programme betrieben werden können, sind sehr kostspielig.
Des Weiteren wird in die Ausbildung im Cyber-Security Bereich stark am Campus Rapperswil-Jona investiert. Im Modul Secure Software soll in Zukunft auch das Thema Bug Bounty stärker geschult werden.

1.1.3. Ziel der Arbeit

Als Ziel dieser Arbeit soll eine Bug Bounty Plattform entwickelt werden. Einerseits soll die Plattform für die Ausbildung im Bereich Bug Bounty beitragen und die Studierenden beim Kennenlernen mit dem Konzept hinter Bug Bounty unterstützen, andererseits soll die Plattform aber auch für interne Software eingesetzt werden, um das Vulnerability Management zu unterstützen und Schwachstellen aufzudecken.

1.2. Vorgehen

1.2.1. Ansatz

Im Vorfeld zur Implementation arbeitete sich das Projektteam in die Aufgabe ein und führte Recherchen zum Thema Bug Bounty durch. Dies beinhaltete eine Anforderungsanalyse an eine Bug Bounty Plattform mit deren funktionalen- wie auch nichtfunktionalen Anforderungen. Darauffolgend wurde vertieft ausgearbeitet, wie und in welchem Rahmen die Lösung ausgearbeitet werden kann. Zudem wurde ein Prototyp erstellt, um die Abläufe und Prozesse der Applikation besser zu visualisieren. In einer späteren Phase wurde der zuvor entworfene Prototyp erfolgreich als Applikation umgesetzt. In einer letzten Phase wurde eine Validierung der Arbeit durchgeführt. Während der Validierung wurde die Applikation mit einer Gruppe von Studenten getestet. Das Feedback der Studierenden und den Mitarbeitenden des Institute for Networked Solutions wurde eingeholt und in der Arbeit ausgewertet.

1.2.2. Technologien

Die Plattform wurde als Cloud Native Applikation entwickelt. Das bedeutet, dass keine klassische On-Premise Frontend-Backend Applikation vorliegt, sondern das Frontend in der Cloud bereitgestellt wird und die Backend-Logik über Serverless Lambda Functions ausgeführt wird. Dies hat zur Folge, dass die Applikation besonders kostengünstig in der Cloud betrieben werden kann.

Amazon Web Services wurde als Cloudlösung ausgewählt, da in Kombination mit AWS Amplify das Entwicklen von Software in der Cloud erleichtert wird. Als Frontend Framework wurde Next JS in Kombination mit Mantine UI und TypeScript eingesetzt. Die Webapplikation kommuniziert über eine GraphQL Schnittstelle mit der Datenbank sowie mit den bereitgestellten Serverless Functions.

1.3. Ergebnisse

Zum Ende dieser Arbeit liegt eine Bug Bounty Plattform vor, welche die gewünschten Ziele erfüllt. Die Plattform ist öffentlich über die Webseite www.bugchaser.ch und kann verwendet werden.

Unternehmen können sich registrieren und eine Organisation auf der Plattform anmelden. Nach einer Prüfung durch den Administrator der Plattform können neue Bug Bounty Programme gestartet werden. Ethische Hacker können nach ihrer Registrierung an öffentlichen Programmen teilnehmen. Zudem wurde für den Ausbildungsbetrieb auf der Plattform eine Kategorie Education entworfen, welche im Vergleich zum klassischen Bug Bounty Programm nur Punkte auszahlen kann.
In einer späteren Validierung der Arbeit konnte aufgezeigt werden, dass die Plattform bei den Studierenden gut angenommen wurde. Ebenfalls wurde erkannt, dass das Interesse an internen Bug Bounty Programmen sehr hoch ist.

1.4. Ausblick

1.4.1. Erweiterungen

In der für die Bachelorarbeit vorgegebenen kurzen Zeit wurde der Fokus auf die Kernfunktionalitäten gesetzt. Die Applikation kann in weiteren Folgearbeiten weiter ausgebaut werden. Im Kapitel Erweiterung werden mögliche Funktionalitäten aufgelistet, welche bisher noch nicht implementiert werden konnten.

1.4.2. Vision

Die Bug Chaser Plattform soll in Zukunft für die Ausbildung der Studierenden im Modul Secure Software eingesetzt werden. Zudem erhält die Fachhochschule eine Plattform, um eigene Software von Studierenden, Mitarbeiter oder auch externen IT-Sicherheitsexperten testen zu lassen. Ziel wäre es, dass die OST diese auch einsetzt. So entsteht ein Mehrwert für die OST, wie auch für die Personen, welche an Bug Bounty Programmen teilnehmen. Zudem profitieren alle von der verbesserten Sicherheit und Stabilität der internen Software.

Teil I.

Technischer Bericht

KAPITEL 2

Einführung

2.1. Übersicht

In diesem Kapitel wird eine Einführung in die Welt von Bug Bounty dokumentiert, sowie bestehende Lösungen analysiert und die Zielsetzung mit Lösungsansatz dieser Arbeit besprochen.

2.2. Einleitung

Schwachstellen in Applikationen zu finden ist nicht immer einfach. Für Entwicklungsteams ist es oft nicht möglich, alle Fehler zu finden. Applikationen können heutzutage selbstverständlich mit vielen verschiedenen Tools mehr oder weniger automatisiert nach Sicherheitslücken untersucht werden. Jedoch ist alles zu testen für eine Organisation meistens unmöglich. Genau dort setzt Bug Bounty an. Das Ziel ist es, eine Community von IT-Sicherheitsexperten selbstständig in einem eingeschränkten Bereich der Software nach Schwachstellen suchen zu lassen. Oft wird dabei auch von White-Hat- oder ethischen Hackern gesprochen, welche gefundene Schwachstellen an denjenigen Betreiber melden.

Bei Bug Bounty Programmen wird bei gefundenen Sicherheitslücken im Regelfall als Gegenleistung eine entsprechende Kompensation ausbezahlt. In welcher Höhe diese Kompensation ausfällt, wird jeweils oft in Form von Stufen vom Betreiber des Programms festgelegt. Die Stufen beziehen sich jeweils auf die Art des Fehlers. Ein Cross Site Scripting Angriff wird zum Beispiel oft niedriger gewertet, als eine Remote Code Execution auf einem Server des Betreibers. Dies kann jedoch je nach Betreiber des Programms unterschieden werden. Um eine Kompensation für einen Fehler zu erhalten, muss ein ethischer Hacker dafür ein Write-up einreichen. Dieses muss es dem Betreiber ermöglichen, den Fehler selbstständig zu replizieren.

Wer setzt nun Bug Bounty Programme ein? Grundsätzlich Unternehmen oder Organisationen, welche Software entwickeln. Dies reicht von sehr grossen IT-Unternehmen wie IBM, Twitter und Facebook über kleine Unternehmen bis zu Regierungsorganisationen wie das US Defense Department oder die Bundesverwaltung der Schweiz. Die Programme unterscheiden sich jedoch dabei stark, wer teilnehmen darf. Nicht alle Programme sind immer für jeden zugänglich, sondern nur für einen ausgewählten Kreis an Hackern.

Doch wer profitiert? Schlussendlich alle. Der Entwickler erhält je nach Bug Bounty Plattform eine vielzahl von IT-Sicherheitsexperten, welche seine Applikation nach Schwachstellen testen und bei einem Fund benachrichtigen. Die White-Hat-Hacker erhalten die Möglichkeit, innerhalb eines zuvor definierten Scope legal Applikationen nach Schwachstellen zu untersuchen. Zusätzlich können sie dabei echtes Geld verdienen, falls etwas gefunden wurde. Und zum Schluss profitieren natürlich auch die Endnutzer der Software, da sie so nach der Behebung des Fehlers durch den Anbieter eine sichere Applikation nutzen können.

2.3. Bestehende Lösungen

Für ein Bug Bounty Programm wird grundsätzlich nicht viel benötigt. Unternehmen wie Apple oder Swisscom führen auf Ihrer Webseite Informationen, wie bei einem Fund einer Schwachstelle Kontakt aufgenommen werden kann. Je nach Unternehmen, werden die Höhe der Kompensationen auch direkt auf der Webseite angegeben. Eine Bug Bounty Plattform unterscheidet sich hingegen dazu, dass mehrere Unternehmen und Organisationen dieselbe Plattform nutzen. Zudem werden die Berichte, welche bei einem Fund eingereicht werden, direkt über die Plattform abgewickelt und nicht per E-Mail. Öffentliche Bug Bounty Plattformen existieren bereits. Zwei sehr bekannte Plattformen sind HackerOne und Bugcrowd. Mit Bug Bounty Switzerland wurde im Jahr 2020 die erste Bug Bounty Plattform der Schweiz zum Leben erweckt.

Die folgende Abbildung ist ein Screenshot aus der Bugcrowd Platform. Die Programme sind nach verschiedenen Merkmalen kategorisierbar. Es kann nach Technologien, Kompensationsbeträgen und weiteren Eigenschaften gefiltert werden. Gewisse dieser Programme können ohne Vorwissen beigetreten werden, andere haben Anforderungen, welche ein ethischer Hacker erfüllt haben muss, um am Programm teilzunehmen.

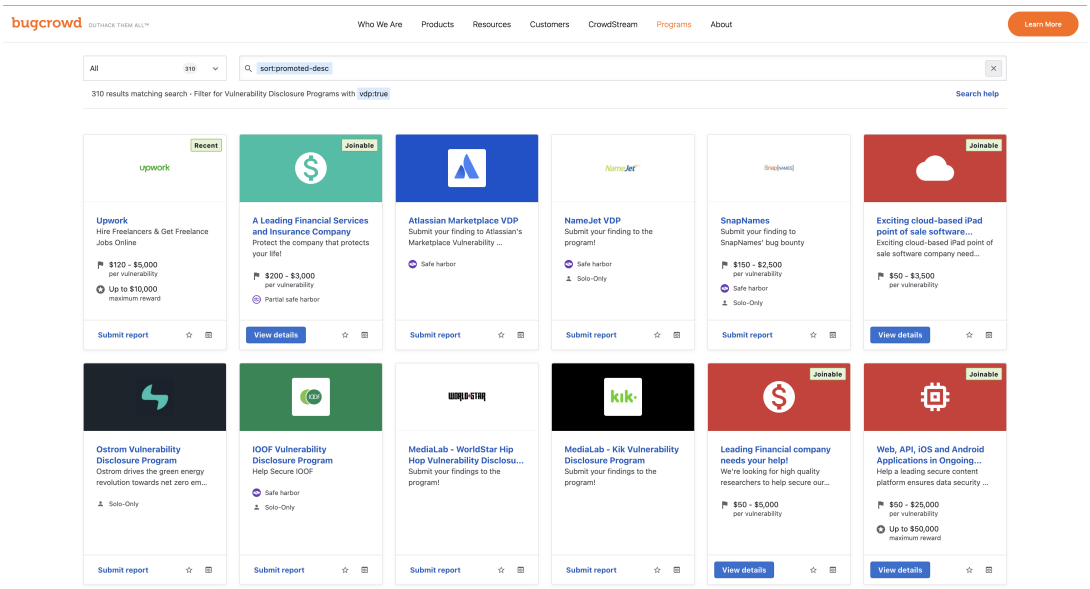


Abbildung 2.1.: Bugcrowd Plattform

Der folgende Screenshot zeigt die Plattform HackerOne. Auch hier sind diverse Programme zu sehen. Wie bei Bugcrowd kann natürlich auch nach diversen Kriterien gefiltert und sortiert werden. So lassen sich nur die Programme anzeigen, welche auch wirklich relevant sind. Der Bereich, von welchem eine Kompensation ausgezahlt wird, ist hier ebenfalls ersichtlich.

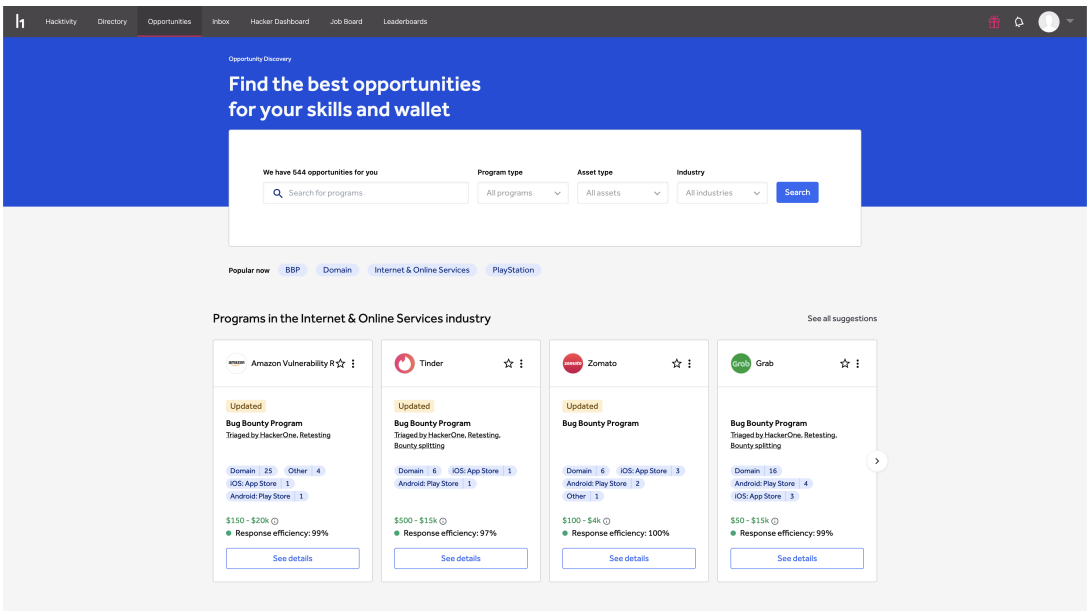
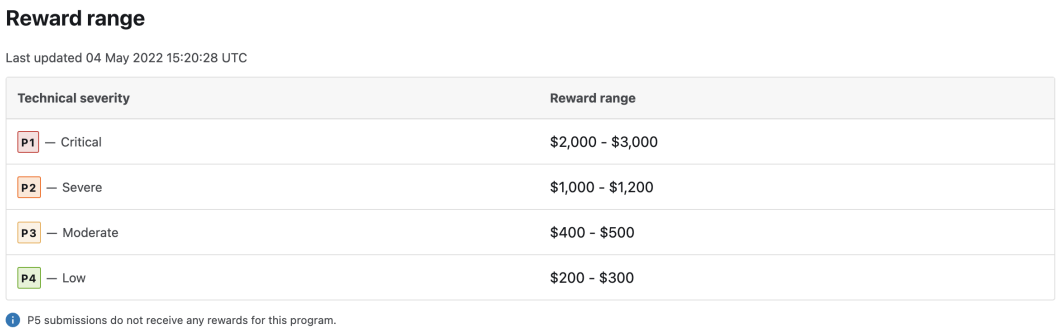


Abbildung 2.2.: HackerOne Plattform

Was beinhaltet ein Bug Bounty Programm? Normalerweise enthält ein Programm diverse Informationen über die Applikation selbst, eine kurze Beschreibung und Statistiken über die vergangenen Bug Reports. Ein Programm enthält auch eine Bug Bounty Table. Dort wird definiert, in welcher Höhe die Kompensation pro Stufe liegt. Ein Beispiel einer solchen Tabelle folgt.



Reward range

Last updated 04 May 2022 15:20:28 UTC

Technical severity	Reward range
P1 — Critical	$2,000 - $3,000
P2 — Severe	$1,000 - $1,200
P3 — Moderate	$400 - $500
P4 — Low	$200 - $300

P5 submissions do not receive any rewards for this program.

Abbildung 2.3.: Bugcrowd Program Bounty Table

Zusätzlich können pro Programm sogenannte Scopes (Targets) festgelegt werden. Diese Ziele definieren, welche Bereiche der Applikation legal untersucht bzw. angegriffen werden dürfen. Es ist auch festgelegt, unter welchen Bedingungen diese Untersuchungen und Angriffe vorgenommen werden dürfen. Meistens können auch Out of scope Targets angegeben werden. Diese dürfen unter keinen Umständen angegriffen werden, da diese meistens produktive Umgebungen enthalten.

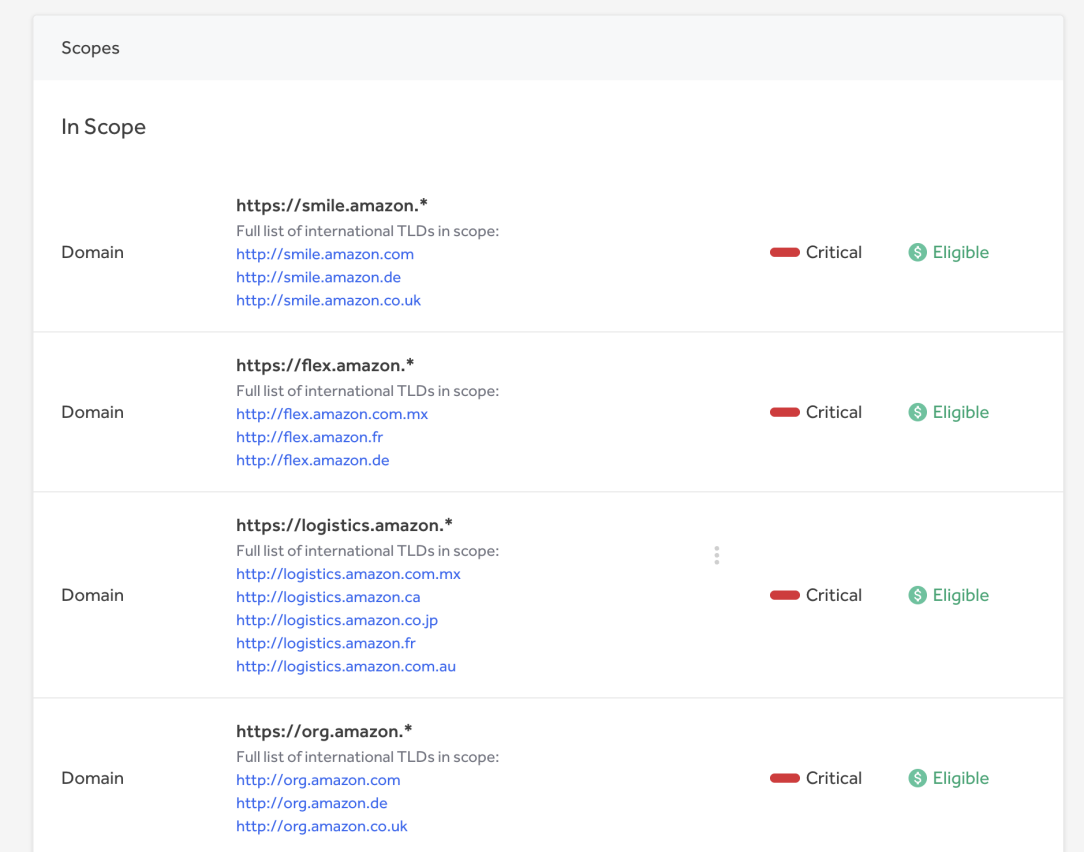


Abbildung 2.4.: Bugcrowd Program Scope

Diese bereits existierenden Lösungen sind sehr ausgereift und bieten viele Funktionen für Unternehmen, Organisationen und White-Hat-Hacker an. Im Gegenzug möchten auch die Anbieter der Bug Bounty Plattformen etwas dafür erhalten. Betreiber von Programmen bezahlen dafür Lizenzkosten an die Betreiber der Plattform. Aufgrund der hohen Kosten sind diese Plattformen daher für die Ausbildung in Hochschulen und Universitäten eher ungeeignet.

2.4. Zielsetzung

Ziel dieser Arbeit ist es, eine Bug Bounty Plattform für die Fachhochschule OST zu entwickeln. Diese soll für die Ausbildung im Modul Secure Software eingesetzt werden. Zudem sollte es in Zukunft der Fachhochschule ermöglicht werden, eigene interne Applikationen auf der Plattform als Bug Bounty Programm anzubieten.

Aufgrund der beschränkten Zeit des Projektes werden dabei vor allem auf die Hauptfunktionen, welche für eine Bug Bounty Plattform benötigt werden umgesetzt. Ziel ist es, dass in einer weiterführenden Arbeit zusätzliche Funktionalitäten hinzugefügt werden. Eine Übersicht an Funktionen, welche bis zum Ende der Arbeit nicht umgesetzt wurden ist im Kapitel Erweiterung zu finden.

2.5. Lösungsansatz

Die Arbeit wurde in vier verschiedene Phasen aufgeteilt: Inception, Elaboration, Construction und Transition. Diese Phasen halfen, dem Projekt einen Rahmen und zeitliche Beschränkung zu schaffen. Die Dauer des Projektes erstreckte sich über knapp 16 Wochen.

In der Inception Phase wurde die Aufgabenstellung analysiert und sich in das Thema eingelesen.

In der Elaboration Phase wurden dann die Anforderungen an die Arbeit und das Produkt festgelegt. Ein erster Prototyp mit Wireframes wurde zu diesem Zeitpunkt ebenfalls erstellt. Dieser half um die Applikation und Abläufe zu visualisieren und teilweise die Logik testen.

Die Entwicklung startete dann in der Construction Phase. Die initiale Konfiguration in der AWS Cloud in Kombination mit AWS Amplify dauerte etwas länger als geplant, da die Dokumentation teilweise unzufriedenstellend war. Nach den Startschwierigkeiten konnten dann aber in einem guten Tempo und nach Plan die Funktionen der Applikation implementiert werden.

In der letzten Phase, Transition, konnte die Applikation mithilfe von einer Gruppe Studenten aus dem Modul Secure Software validiert werden. Die Funktionalität und Intuitivität der Arbeit wurde dabei getestet. In dieser Phase wurde auch die Dokumentation abgeschlossen.

Es wird in einem Team von zwei Personen gearbeitet.

2.5.1. Validierung

Zum Ende der Arbeit soll mittels einer Validierung herausgefunden werden, ob die zuvor definierten Anforderungen und und das Konzept der Applikation standhalten. Einerseits sollen dabei Rückmeldungen den ethischen Hackern sowie den Betreibern eines Programms eingeholt werden. Das Projektteam entschied sich dazu, in einer Übungslektion im Modul Secure Software, in welcher die Applikation später auch verwendet werden soll, diese Validierung auszuführen.

Als ethische Hacker standen über 80 Studierende zur verfügung, welche die Möglichkeit erhielten, eine Applikation in einem Bug Bounty Programm zu testen. In diesem Szenario werden die eingereichten Bugs dann von der Übungsbetreuung kontrolliert. Die Übungsbetreuer sind so in der Position eines Unternehmens, welche das Programm betreiben. Während der Übung soll einerseits von den Studierenden eine Rückmeldung eingeholt werden sowie die Übungsbetreuer befragt werden. Die daraus entstandenen Erkenntnisse werden im Abschnitt Resultate der Validierung dokumentiert.

KAPITEL 3

Resultate

3.1. Ergebnisse

In den knapp 16 Wochen, in welchem diese Arbeit stattgefunden hat, wurde wie geplant eine Bug Bounty Plattform entwickelt. Diese heisst **Bug Chaser** und ist einerseits für Schulzwecke wie auch anderseits für Organisationen einsetzbar. Die Web Applikation wird als Cloud Native System in der AWS Cloud betrieben.

3.1.1. Resultate der Validierung

Die Validierung wurde mit einer Gruppe von Studenten aus dem Secure Software Modul durchgeführt. Die Aufgabe der Studenten wurde von den Übungsbetreuenden auf der Bug Chaser Plattform konfiguriert. Die Studenten hatten die Aufgabe, im hochgeladenen Programm, in welchem explizit Bugs eingebaut wurden, Schwachstellen zu suchen und diese im Tool zu rapportieren. Die Übungsbetreuung konnte dann die Bug Reports bewerten und die Studenten mit Punkten belohnen.

Um ein Feedback der Studenten zu erhalten wurden diese mit einem Microsoft Forms Formular nach Problemen, Erfolgen und Verbesserungsvorschlägen zur Applikation befragt. Die Auswertung dieser Befragung fiel sehr positiv aus. Die Plattform kam sehr gut an. Die meisten Studenten haben sich in der Applikation gut und intuitiv zurechtgefunden und konnten die Aufgabe lösen. Auch die Idee, dass die Fachhochschule OST in Zukunft, eigene Software in Bug Bounty Programmen testen kann wurde mit der Umfrage bestätigt. Es besteht bei den Studierenden grosses Interessen, dass die OST dies in naher Zeit einführt. Als Kompensation dafür wurde von den Studierenden zum Beispiel vorgeschlagen, dass Digitec, Brack, Mensa oder ähnliche Gutscheine verteilt werden. Einige waren auch der Meinung, dass sie alleine mit verbesserter Software zufrieden sind, also ohne Auszahlung von Geldbeträgen oder Gutscheinen.

Verbesserungsvorschläge beinhalteten hauptsächlich weitere Funktionalitäten der Applikation. So wurde zum Beispiel ein Leaderboard gewünscht, welches im Backend bereits umgesetzt wurde, im Frontend aufgrund der eingeschränkten Zeit jedoch nicht mehr implementiert werden konnte. Zudem wurde gewünscht, dass die FAQ-Seite mit Erklärungen weiter ausgebaut wird, sodass auch Personen, welche bisher noch keine Erfahrungen mit Bug Bounty sammeln konnte, den Einstieg erleichtert wird.

Zudem wurden die Übungsbetreuer nach ihren Erlebnissen mit der Plattform und dessen ersten Einsatz befragt. Dabei ist derselbe Trend zu erkennen, welcher auch schon bei den Übungsteilnehmer erkannt wurde. Grundsätzlich ist die Applikation gemäss Anforderungen einsetzbar. Auch von den Betreuern wurden zusätzliche Funktionen gewünscht, welche zum aktuellen Zeitpunkt noch nicht umgesetzt werden konnten. Diese konnten jedoch aufgenommen werden und sind im Abschnitt Erweiterung dokumentiert.

3.2. Schlussfolgerungen

Die erarbeitete Bug Chaser Plattform konnte wie geplant umgesetzt werden. Wie in einer Validierung gezeigt werden konnte, erfüllt diese die geforderten Anforderungen.
Dies bedeutet, dass die Lehre an der OST nun zusätzlich unterstützt werden kann. Weiter können zusätzliche Funktionen in die Anforderungen aufgenommen werden, welche später in die Plattform implementiert werden können.

Teil II.

Vorstudie

KAPITEL 4

Anforderungsspezifikation

4.1. Akteure

Für eine Bug Bounty Applikation wurden die folgenden Akteure definiert und beschrieben.

Akteur	Beschreibung
Hunter	Der Benutzer der Applikation, welcher nach Bugs sucht, wird Hunter genannt. Ein Hunter fungiert als Anwender der Applikation.
Organisation	Als Organisation wird ein Unternehmen bezeichnet, welches die Möglichkeit erhält, für seine Applikationen ein Bug Bounty Programm zu starten. Eine Organisation wird in mehrere Benutzer unterteilt, welche die Organisation selbst sowie ihre Bug Bounty Programme verwaltet. Diese werden wie folgt unterteilt:
Org-Administrator	Dieser verwaltet die Organisation selbst. Er ist zuständig für das Erstellen sowie das Bearbeiten der Informationen des Unternehmens. Er kann zusätzlich weitere Programm-Administratoren einladen.
Programm-Administrator	Das Bug Bounty Programm einer Applikation wird durch einen Programm-Administrator verwaltet.
Programm-Responder	Ein Responder kann auf Reports zu einem Programm antworten. Er handelt aus Sicht einer Organisation.
Administrator	Die Bug Bounty Applikation wird von einem Administrator verwaltet.

Tabelle 4.1.: Akteure

4.2. Epics

Die gewünschten Funktionen der Bug Bounty Applikation werden auf hoher Abstraktionsebene als Epics definiert. Diese werden anschliessend als Use Cases genauer spezifiziert und detaillierter beschrieben. Pro Epic ergeben sich immer mehrere Use Cases.

Epic ID	Titel	Beschreibung
E01	CRUD Benutzer	Ein Benutzer kann sich auf der Plattform registrieren. Persönliche Informationen können angegeben werden. Die E-Mail Adresse wird während der Registration verifiziert. Nach erfolgreicher Registrierung kann er seine eigenen Daten einsehen. Einige Daten können geändert werden, andere nicht. Das eigene Nutzerkonto kann gelöscht werden.
E02	CRUD Organisation	Eine Organisation kann sich auf der Plattform registrieren. Bei der Registrierung müssen Informationen zur Organisation angegeben werden. Diese werden für die Verifikation benötigt. Pro Organisation kann es nur einen Organisationsadministrator geben.
E03	CRUD Bug Bounty Programm	Ein Organisation-Administrator erhält die Möglichkeit, neue Bug Bounty Programme zur Organisation hinzufügen. Diese Programme können mit Informationen zur Applikation, welche getestet werden soll, ergänzt werden. Der Programm-Administrator kann später diese Informationen editieren. Ebenfalls kann er dieses auch wieder entfernen. Programme können öffentlich oder privat sein.
E04	Teilnehmen an einem Bug Bounty Programm	Ein Hunter kann sich auf einem Bug Bounty Programm teilnehmen. Ein Programm-Administrator oder Programm-Responder kann mit dem Hunter kommunizieren und sich so über den eingereichten Report austauschen. Entspricht der Report den Anforderungen der Organisation, so kann der Programm-Administrator eine Entlohnung auszahlen.
E05	Öffentliches Profil	Jeder Hunter sowie jede Organisation haben ein öffentliches Profil. Das Profil kann von allen anderen Benutzern eingesehen werden. Der Hunter respektive die Organisation können das Profil mit persönlichen Informationen ergänzen. Erhaltene Trophäen werden ebenfalls hier angezeigt.
E06	Öffentliches Leaderboard	Über die ganze Applikation wird eine Bestenliste in verschiedenen Kategorien geführt. Dieses ist öffentlich für alle angemeldeten Benutzer einsehbar. Statistiken werden automatisch im Leaderboard aktualisiert.
E07	Infoseite	Um den Einstieg in die Bug Bounty Plattform zu vereinfachen, werden verschiedene Seiten dem Nutzer erklären, welche Vorteile durch die Plattform für ihn geschaffen werden. Hauptsächlich dienen diese zu Marketing-Zwecken.

E08	Versenden von Nachrichten	Es sollen in verschiedenen Nachrichten an einen Benutzer versendet werden. Nachrichten können ein- oder auch ausgeschaltet werden. Es können verschiedene Plattformen zur Übertragung von Nachrichten verwendet werden.
E09	Administration der Plattform	Die Plattform wird von einem Administrator verwaltet. Der Administrator kann grundlegende Aktionen durchführen. Beispiel dafür ist das Aktivieren von Organisationen oder das Löschen von Benutzerkonten.

Tabelle 4.2.: Epics

4.3. Use Case Diagramm

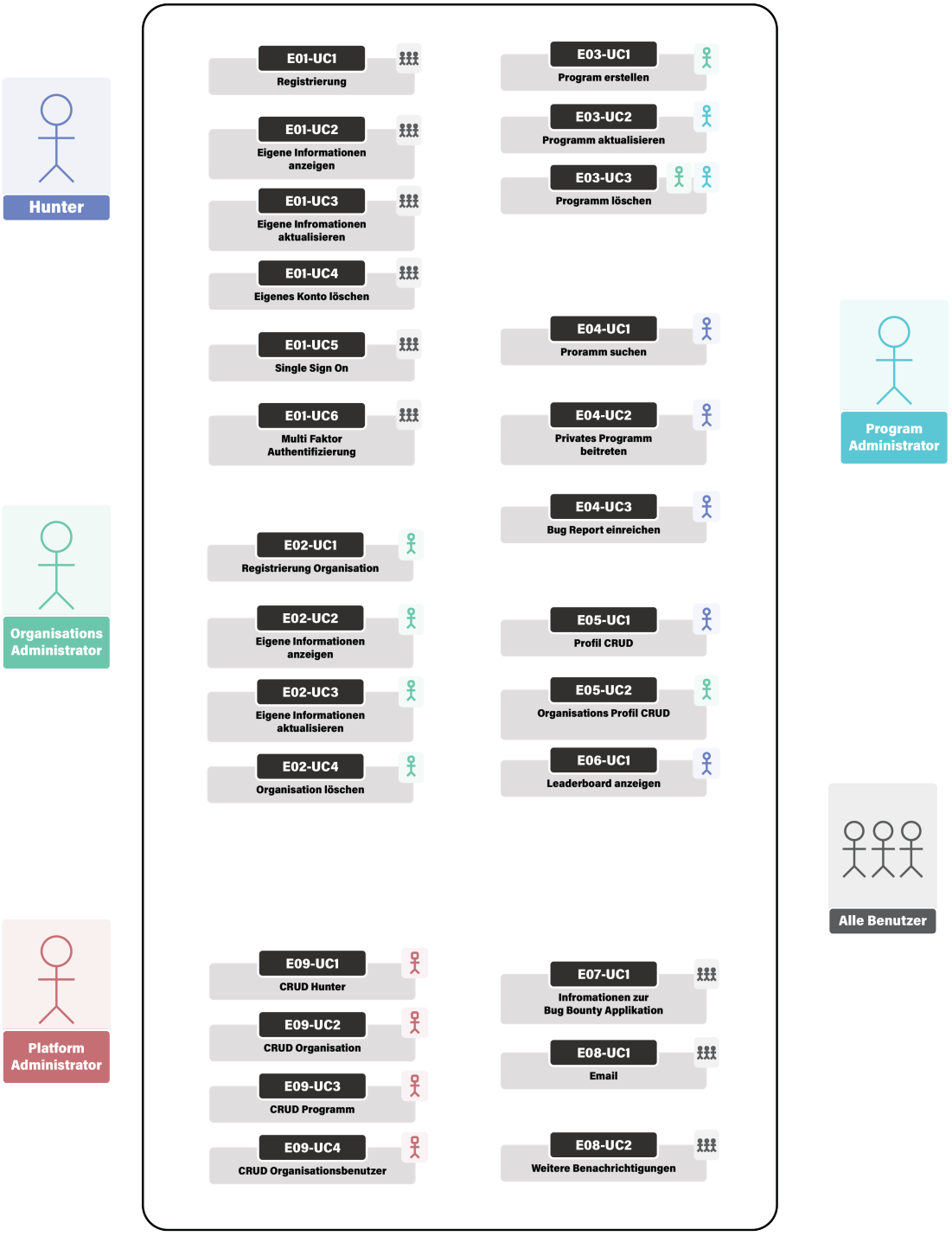


Abbildung 4.1.: Use Case Diagramm

4.4. Use Cases

In diesem Abschnitt werden alle Use Cases detailliert beschrieben. Die Use Cases sind immer unter ihrem dazugehörigen Epic aufgelistet. Zur Identifizierung erhalten alle eine eindeutige ID. Diese besteht aus dem Format `EpicID-UseCaseID` sowie einem passenden Titel.

4.4.1. Epic 01 - CRUD Benutzer

E01-UC01 Registrierung Hunter

Beschreibung	Ein Benutzer möchte sich als Hunter oder Organisationsbenutzer registrieren, um die Applikation verwenden zu können.
Akteur	Org-Administrator, Programm-Administrator, Programm-Responder, Hunter
Akzeptanzkriterien	☐ Der Benutzer kann mit sich mit einer E-Mail Adresse, Benutzernamen und Passwort registrieren. ☐ Ist der Benutzername bereits registriert, so wird dieser nicht akzeptiert und eine Fehlermeldung wird angezeigt. ☐ Pro E-Mail Adresse können mehrere Benutzerkonten registriert werden. ☐ Die E-Mail Adresse wird über eine Nachricht mit einem personalisiertem Link verifiziert. ☐ Das Konto ist erst nach erfolgreicher Verifizierung aktiv.
Priorität	Sehr hoch

E01-UC02 Eigene Informationen anzeigen

Beschreibung	Ein Benutzer, welcher sich erfolgreich registriert hat, möchte die von ihm gespeicherten Informationen anzeigen.
Akteur	Org-Administrator, Programm-Administrator, Programm-Responder, Hunter
Akzeptanzkriterien	☐ Benutzer kann nach Login eigene Informationen anzeigen
Priorität	Hoch bis sehr hoch

E01-UC03 Eigene Informationen aktualisieren

Beschreibung	Ein Benutzer möchte seine gespeicherten Informationen bearbeiten. Dazu gehört zum Beispiel sein Passwort.
Akteur	Org-Administrator, Programm-Administrator, Programm-Responder, Hunter
Akzeptanzkriterien	☐ Benutzer kann nach Login sein Passwort aktualisieren ☐ Hunter kann ein Profilbild hinzufügen, ändern
Priorität	Hoch bis sehr hoch

E01-UC04 Konto löschen

Beschreibung	Ein Benutzer entscheidet sich, dass er sein eigenes Profil löschen möchte.
Akteur	Org-Administrator, Programm-Administrator, Programm-Responder, Hunter
Akzeptanzkriterien	☐ Der Benutzer kann sein Konto löschen. ☐ Der Benutzer muss das Löschen seines Kontos bestätigen. ☐ Die Daten werden komplett gelöscht und sind nicht wiederherstellbar.
Priorität	Mittel

E01-UC05 Single Sign On

Beschreibung	Ein Benutzer möchte sich anstatt per E-Mail Adresse mit seinem Social Media Konto authentifizieren.
Akteur	Org-Administrator, Programm-Administrator, Programm-Responder, Hunter
Akzeptanzkriterien	☐ Der Nutzer kann sich über Single Sign On (SSO) registrieren. ☐ Der Nutzer kann sich über SSO anmelden. ☐ Ein späterer Wechsel von E-Mail zu SSO Authentication oder umgekehrt wird nicht unterstützt.
Priorität	Niedrig

E01-UC06 Multi-Faktor-Authentifizierung

Beschreibung	Ein Benutzer möchte zusätzlich zu seinem Passwort eine Multi-Faktor-Authentifizierung (MFA) für sein Konto hinzufügen.
Akteur	Org-Administrator, Programm-Administrator, Programm-Responder, Hunter
Akzeptanzkriterien	☐ MFA kann in den Kontoeinstellungen eingerichtet werden. ☐ Der Nutzer muss beim Einrichten des MFA diesen zuerst bestätigen, bevor dieser aktiv im Nutzerkonto eingesetzt wird. ☐ Bei Verlust eines Faktors, kann das Login über die E-Mail Adresse wiederhergestellt werden.
Priorität	Niedrig bis mittel

4.4.2. Epic 02 - CRUD Organisation

E02-UC01 Registrierung Organisation

Beschreibung	Ein Unternehmen möchte sich als Organisation registrieren, um die Applikation verwenden zu können.
Akteur	Org-Administrator
Akzeptanzkriterien	☐ Organisationsbenutzer kann neue Organisationen erstellen ☐ Die Organisation muss vom Plattform-Administrator aktiviert werden ☐ Die Organisation kann vom Plattform-Administrator abgelehnt werden
Priorität	Sehr hoch

E02-UC02 Eigene Informationen anzeigen

Beschreibung	Der Organisations Administrator möchte die gespeicherten Informationen zur Organisation anzeigen.
Akteur	Org-Administrator
Akzeptanzkriterien	☐ Organisations-Administrator kann Details über Organisation anzeigen.
Priorität	Hoch

E02-UC03 Eigene Informationen aktualisieren

Beschreibung	Der Organisations Administrator möchte die gespeicherten Informationen zur Organisation bearbeiten.
Akteur	Org-Administrator
Akzeptanzkriterien	☐ Der Organisations-Administrator kann Details über Organisation aktualisieren. ☐ Der Organisation kann deaktiviert und aktiviert werden.
Priorität	Mittel

E02-UC04 Organisation löschen

Beschreibung	Die Organisation entscheidet sich, in Zukunft kein Bug Bounty Programm mehr zu verwenden. Dazu wird die Organisation in der Applikation gelöscht
Akteur	Org-Administrator
Akzeptanzkriterien	☐ Organisationsadministrator kann gesamte Organisation löschen ☐ Alle Programme der Organisation werden ebenfalls gelöscht
Priorität	Mittel bis hoch

4.4.3. Epic 03 - CRUD Bug Bounty Program

E03-UC01 Programm erstellen

Beschreibung	Ein Organisations-Administrator möchte ein neues Bug Bounty Programm für eine Applikation erstellen.
Akteur	Organisations-Administrator
Akzeptanzkriterien	☐ Organisations-Administrator kann Programme erstellen ☐ Organisations-Administrator wird automatisch als Programm-Administrator im Programm hinzugefügt ☐ Name und allgemeine Informationen über die Applikation können hinterlegt werden ☐ verwendete Technologien können angegeben werden ☐ Bounty Tabelle kann definiert werden ☐ Out of Scope und In Scope Targets können definiert werden ☐ allgemeine Spielregeln können festgelegt werden ☐ privat oder öffentlicher Modus kann definiert werden ☐ es kann zwischen EDU, VDP und BBP unterschieden werden
Priorität	Sehr hoch

E03-UC02 Programm aktualisieren

Beschreibung	Ein Programm-Administrator möchte ein bestehendes Bug Bounty Programm für eine Applikation aktualisieren.
Akteur	Programm-Administrator
Akzeptanzkriterien	☐ Programmadministrator kann eigene erstellte Programme anpassen
Priorität	Hoch

E03-UC03 Programm löschen

Beschreibung	Ein Programm-Administrator möchte ein bestehendes Bug Bounty Programm für eine Applikation löschen.
Akteur	Programm-Administrator oder Organisations-Administrator
Akzeptanzkriterien	☐ Programmadministrator kann eigene erstellte Programme löschen
Priorität	Hoch

4.4.4. Epic 04 - Participate in Bug Bounty Program

E04-UC01 Programm anzeigen

Beschreibung	Ein Benutzer möchte an einem Programm teilnehmen. Er kann durch eine Übersicht sich alle Programme anzeigen lassen.
Akteur	Hunter
Akzeptanzkriterien	☐ Hunter kann Programme anzeigen.
Priorität	Sehr hoch

E04-UC02 Privates Programm beitreten

Beschreibung	Ein Hunter will an einem Programm teilnehmen welches privat ist.
Akteur	Hunter
Akzeptanzkriterien	☐ Hunter kann Programm Übersicht anschauen. ☐ Über ein Token kann der Hunter einem privaten Programm beitreten.
Priorität	Sehr hoch

E04-UC03 Bug Report einreichen

Beschreibung	Nachdem ein Bug gefunden wurde, möchte der Hunter einen neuen Report einreichen.
Akteur	Hunter
Akzeptanzkriterien	☐ Hunter kann auf eingeschrieben Programme Bug Reports einreichen ☐ Hunter kann Report in Markdown ähnlicher Syntax geschrieben werden ☐ Hunter und Programmadministrator können kommunizieren (Chat Funktion) ☐ Programmadministrator kann Rewards an Hunter übertragen
Priorität	Sehr hoch

4.4.5. Epic 05 - Hunter / Organisation Profile

E05-UC01 Profil CRUD

Beschreibung	Ein Hunter kann sein eigenes Profil und Details anpassen und anzeigen lassen.
Akteur	Hunter
Akzeptanzkriterien	☐ Hunter kann eigenes Profil einsehen ☐ Hunter kann eigenes Profil editieren – Biographie – Last Activity – Statistics – Social Media Links ☐ Hunter kann andere Profile einsehen (Hunter und Organisationen)
Priorität	Mittel

E05-UC02 Organisations Profil CRUD

Beschreibung	Ein Organisation-Administrator kann sein das Profil und Details anpassen und anzeigen lassen.
Akteur	Organisation-Administrator
Akzeptanzkriterien	☐ Organisation-Administrator und Hunter kann das Organisations Profil einsehen ☐ Organisation-Administrator kann das Organisations Profil editieren – Beschreibung der Organisation – Kontaktinformationen
Priorität	Mittel

4.4.6. Epic 06 - Leaderboard Dashboard

E06-UC01 Leaderboard anzeigen

Beschreibung	Jeder Benutzer kann sich die Bestenliste, aller Hunter anzeigen lassen. Dabei wird eine Statistik in verschiedenen Kategorien angezeigt.
Akteur	Hunter
Akzeptanzkriterien	☐ Leaderboard kann öffentlich zugegriffen werden
	☐ Leaderboard stellt die Hunter mit den meisten Punkten dar
Priorität	Niedrig bis mittel

4.4.7. Epic 07 - Infoseite

E07-UC01 Informationen zur Bug Bounty Applikation

Beschreibung	Eine Informationsseite gibt Auskunft über die Bug Bounty Applikation.
Akteur	Alle Benutzer
Akzeptanzkriterien	☐ Infoseite ist öffentlich zugänglich
	☐ informiert über Bug Bounty allgemein
	☐ informiert über die Plattform selbst
	☐ informiert über die Ersteller der Applikation
	☐ ist mobile friendly
Priorität	Sehr hoch

4.4.8. Epic 08 - Benachrichtigungen

E08-UC01 E-Mail

Beschreibung	Die Applikation kann E-Mails versenden. Dies wird für zum Beispiel für die Registrierung oder Passwort zurücksetzen benötigt.
Akteur	Alle Benutzer
Akzeptanzkriterien	☐ Für die Registrierung werden Benachrichtigungen versendet ☐ Für den Passwort Reset werden Benachrichtigungen versendet
Priorität	Sehr hoch

E08-UC02 Weitere Benachrichtigungen

Beschreibung	Ein Nutzer erhält bei Aktivität auf dem Konto eine Benachrichtigung per E-Mail.
Akteur	Alle Benutzer
Akzeptanzkriterien	☐ Benachrichtigungen werden automatisch per E-Mail versendet bei Aktivität auf dem Konto
Priorität	Niedrig

4.4.9. Epic 09 - Administration der Plattform

E09-UC01 CRUD Benutzer

Beschreibung	Der Administrator der Applikation kann die Benutzer verwalten.
Akteur	Administrator
Akzeptanzkriterien	☐ Administrator kann Hunter und Organisationsbenutzer löschen
Priorität	Mittel bis hoch

E09-UC02 CRUD Organisation

Beschreibung	Der Administrator der Applikation kann die Organisationen verwalten.
Akteur	Administrator
Akzeptanzkriterien	☐ Administrator kann Organisationen löschen ☐ Administrator kann Organisationen aktivieren ☐ Administrator kann Organisationen deaktivieren
Priorität	Mittel bis hoch

4.5. Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen werden nach der internationale Norm ISO/IEC 25010:2011 [1] definiert. Diese Norm deckt die Qualitätsanforderungen ab, welche vom Produkt gefordert werden, jedoch noch nicht als funktionale Anforderung aufgenommen wurde. Merkmale, welche zwar in der Norm enthalten sind, für das Projekt jedoch irrelevant sind, werden nicht weiter spezifiziert.

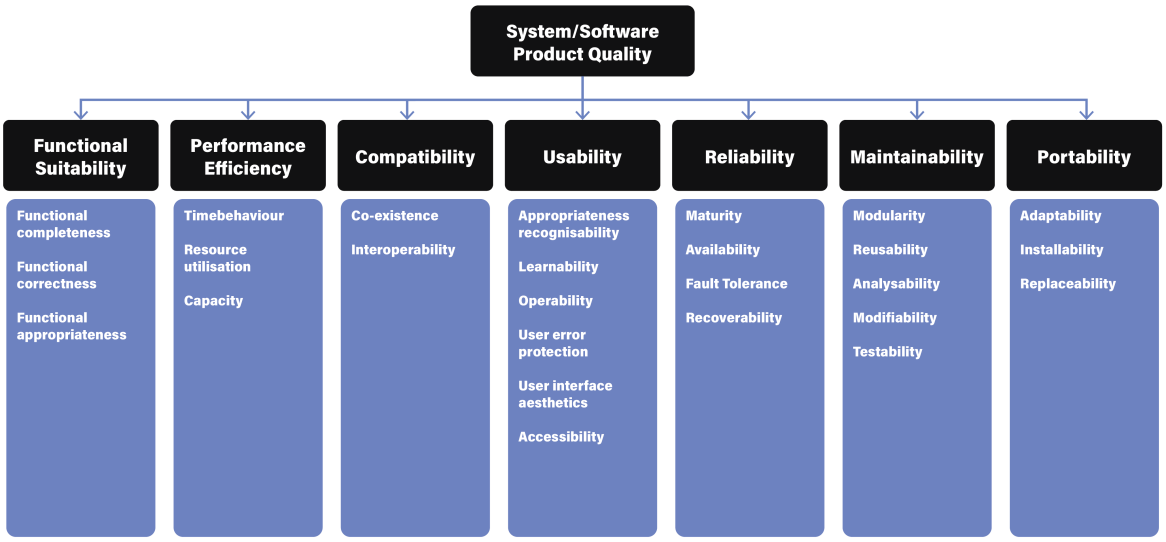


Abbildung 4.2.: Übersicht ISO/IEC 25010:2011

4.5.1. Functional Suitability

Charakteristik	Merkmal	Akzeptanzkriterium
Functional completeness	Die in der Anforderungsspezifikation definierten funktionalen Anforderungen sind vorhanden und funktionieren vollständig.	Die definierten Akzeptanzkriterien der Use Cases wurden erfüllt.
Functional correctness	Vom Benutzer eingegebene Daten werden korrekt von System verarbeitet und korrekt ausgegeben.	Eine Input Validierung muss sowohl auf dem Frontend sowie im Backend, respektive in der Datenbank implementiert sein.

Tabelle 4.3.: Anforderungen zu Functional Suitability

4.5.2. Performance Efficiency

Charakteristik	Merkmal	Akzeptanzkriterium
Time behavior	Die Applikation sendet die Antworten an den Benutzer zeitnah. Bei längeren Wartezeiten wird der Nutzer darüber informiert.	Die Interaktionszeit der Applikation soll bei einer stabilen Verbindung maximal eine Reaktionszeit von einer Sekunde haben. Eine Animation wird bei Wartezeiten angezeigt.
Resource utilization	Die Applikation soll mit Ressourcen sparsam umgehen.	Die Applikation wird in Teile aufgetrennt. Das Rendering der Anzeige wird auf den Client ausgelagert. Das Backend wird mittels Serverless Functions umgesetzt.

Tabelle 4.4.: Anforderungen zu Performance Efficiency

4.5.3. Compatibility

Charakteristik	Merkmal	Akzeptanzkriterium
Co-existence	Die Applikation stellt eine API zur Verfügung. Der Benutzer soll diese nur über das Frontend verwenden.	Die API kann nur vom Frontend genutzt werden. Eine Schnittstelle für Dritte ist nicht vorgesehen.

Tabelle 4.5.: Anforderungen zu Compatibility

4.5.4. Usability

Charakteristik	Merkmal	Akzeptanzkriterium
Appropriateness recognizably	Die Applikation soll nur Funktionen unterstützen, für welche das Produkt ursprünglich auch angedacht war.	Es wurden Erweiterungen als Use Cases für die Applikation definiert. Andere Funktionen sind nicht vorgesehen und wurden nicht umgesetzt.
Learnability	Ein Benutzer findet sich ohne Anleitung in der Applikation zurecht und kann selbstständig, ohne externe Hilfestellung das Produkt verwenden.	Mittels Usabilitytests wurde niedergelegt, dass ein Benutzer ohne Hilfe die Applikation nutzen kann.
Operability	Die Applikation kann sowohl auf Desktop- wie Mobile-Plattformen genutzt werden.	Es werden Usabilitytests auf Desktop- wie Mobile-Plattformen durchgeführt.
User error protection	Vom Benutzer verursachte Fehler führen nicht zu Verlust oder Inkonsistenz der Daten.	Input Validation ist sowohl auf dem Frontend sowie im Backend, respektive in der Datenbank implementiert. Der Nutzer wird bei Eingabefehler darauf hingewiesen.
User interface aesthetics	Das Design soll den Nutzer ansprechen, die Applikation zukünftig respektive weiterhin zu verwenden.	Usabilitytests prüfen auch das Design der Benutzeroberfläche.
Accessibility	Benutzer mit einem eingeschränkten Sehvermögen können die Applikation ebenfalls verwenden.	Die Benutzeroberfläche weist ein Kontrastverhältnis nach Double A [2] Standard auf.

Tabelle 4.6.: Anforderungen zu Usability

4.5.5. Reliability

Charakteristik	Merkmal	Akzeptanzkriterium
Maturity	Die Applikation sollte durchgängig verfügbar sein. Bei Wartungsarbeiten muss der Benutzer vorgängig informiert werden.	Die Applikation wird in der Cloud betrieben.
Availability	Die Applikation ist nur online verfügbar. Eine Offline-Version wird nicht angeboten.	Die Applikation wird in der Cloud betrieben.
Fault tolerance	Bei Hardwarefehler soll die Applikation ununterbrochen auf anderer Hardware ausgeführt werden können.	Ein Deployment in der Cloud eliminiert die Hardwareabhängigkeiten.
Recoverability	Vom Benutzer gelöschte Daten sind gelöscht und können nicht zurückgeholt werden.	Es wird kein Papierkorb geführt.

Tabelle 4.7.: Anforderungen zu Reliability

4.5.6. Security

Charakteristik	Merkmal	Akzeptanzkriterium
Confidentially	Benutzer der Applikation erhalten nach erfolgreichem Login nur Zugriff auf Daten, für welche sie berechtigt sind.	Die Autorisierung wird mit einem IAM System umgesetzt.
Integrity	Die gespeicherten Daten sind persistiert. Weder Fehler oder Stromausfälle des Host-Servers führen zum Datenverlust. Benutzer können nur Daten verändern, für welche sie auch berechtigt sind.	Die Autorisierung wird mit einem IAM System umgesetzt. Ein Deployment in der Cloud eliminiert sämtliche Risiken, welche durch Energieengpässe entstehen für den Betreiber.
Authenticity	Die Applikation soll die Echtheit gegenüber dem Benutzer preisgeben.	Es wird ein gültiges SSL/TLS Zertifikat bei einer HTTPS Verbindung ausgeliefert. HTTP Anfragen werden automatisch zu HTTPS umgeleitet.

Tabelle 4.8.: Anforderungen zu Security

4.5.7. Maintainability

Charakteristik	Merkmal	Akzeptanzkriterium
Modularity	Die Applikation sollte in Teile unterteilt sein, welche austauschbar sind.	Das Backend wird mittels Serverless Functions umgesetzt. Diese Funktionen sind voneinander unabhängig.
Reusability	Der Code soll möglichst so umgesetzt werden, sodass grosse Teile wiederverwendet werden können.	Die Qualität des Codes wird bei jedem Merge nach dem Style Guide geprüft. Zusätzlich prüft SonarQube die Qualität.
Analyzability	Der Source Code soll verständlich sein.	Jeder Entwickler sollte in der Lage sein, den Code zu verstehen. Anderenfalls sind komplexe Methoden dokumentiert.
Modifiability	Die Architektur soll so umgesetzt werden, sodass auch in Zukunft Änderungen an der Funktionalität möglich sind.	Die Applikation wird vollständig dokumentiert.
Testability	Die Applikation sollte automatisiert getestet werden können.	Für die Applikation werden Tests entwickelt, welche halbautomatisch ausgeführt werden können.

Tabelle 4.9.: Anforderungen zu Maintainability

4.5.8. Portability

Charakteristik	Merkmal	Akzeptanzkriterium
Adaptability	Die Applikation soll unabhängig von der Hardware oder vorinstallierter Software ausgeführt werden können.	Das Produkt wird als Deployment in der Cloud ausgeliefert. Alle Abhängigkeiten sind im Repository hinzugefügt.
Installability	Die Applikation soll nur über den Webbrowser verfügbar sein.	Eine Installation respektive Deinstallation ist nicht nötig.

Tabelle 4.10.: Anforderungen zu Portability

KAPITEL 5

Domain Analyse

5.1. Übersicht

In diesem Kapitel wird die Domain Analyse mit einem Domain Model der Datenbank beschrieben. Diese werden während der Construction Phase laufend noch angepasst. Die finale Version des Datenbankschemas ist Architektur Diagramm zu finden.

5.2. Domain Model

Das folgende Domain Model wurde während der Elaboration Phase erstellt. Dieses wurde während der Construction Phase aber noch laufend angepasst.

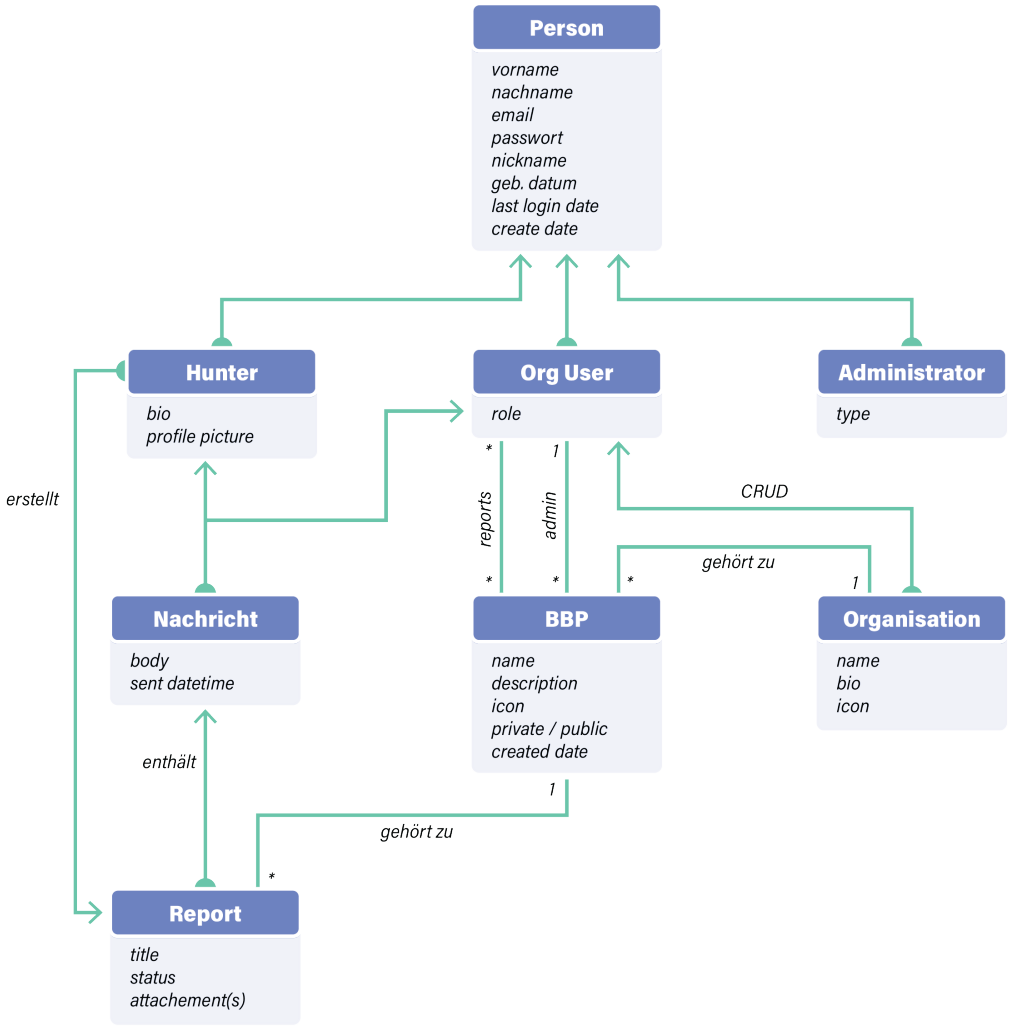


Abbildung 5.1.: Domänenmodell

Die im Domain Model verwendeten Klassen werden folgend genauer erläutert.

5.2.1. Person

Diese Klasse wird als Basis für die Klasse Hunter, Org User und Administrator verwendet. Sie beinhaltet grundlegende Informationen über einen registrierten Benutzer im System.

5.2.2. Hunter

Abgeleitet von der Basisklasse Person enthält die Hunter Klasse noch zusätzliche Informationen, welche nur für den Hunter relevant sind.

5.2.3. Org User

Der Organisations User enthält eine Rolle, welche in der Organisation für die Berechtigungen relevant ist.

5.2.4. Administrator

Der Administrator ist ein spezieller Benutzer mit grundlegenden Rechten über das gesamte System. Er kann globale Änderungen und Aktionen tätigen.

5.2.5. Organisation

Die Organisation beinhaltet Informationen zur Firma, welche das Bug Bounty Programm veröffentlicht.

5.2.6. Message

Diese Klasse wird im Report verwendet, wenn Hunter und Org User miteinander kommunizieren.

5.2.7. BBP (Bug Bounty Programm)

Das Programm wird von einer Organisation veröffentlicht, und lässt es zu, dass Hunter darauf Bugs einreichen.

5.2.8. Report

Der Report selbst wird von einem Hunter eingereicht und gehört zu einem spezifischen Programm.

KAPITEL 6

Technologie-Evaluation

6.1. Übersicht

In dieser Arbeit wird eine Applikation entwickelt, welche auf verschiedenen Plattformen ausführbar sein muss. In erster Linie wird das Produkt hauptsächlich von Softwareentwickler und Personen aus dem Cyber Security Bereich verwendet. Eine Anforderung ist es daher, dass die Anwendung einerseits auf Windows und MacOS, sowie auch auf Linux benutzt werden kann.

In diesem Abschnitt wird dokumentiert, welche Überlegungen bei der Auswahl der Architektur sowie der verwendeten Technologien getroffen wurde. Ziel ist es, Technologien einzusetzen, welche aktuell in der Softwareentwicklung sehr verbreitet sind. Trotzdem sollten diese auch in Zukunft sicher noch einen hohen Stellenwert im Softwareentwicklungsprozess haben.

6.2. Einschränkungen

In dieser Vorstudie werden nicht alle möglichen Varianten und Technologien, welche zum Zeitpunkt des Projektes auf dem Markt sind, beschrieben. Hauptsächlich wurde bei der Auswahl auf bewährte Modelle gesetzt und diese miteinander verglichen. Zusätzlich müssen diese eine Möglichkeit bieten, die definierten Anforderungen in der Anforderungsanalyse umzusetzen.

6.3. Varianten

Die Bug Bounty Applikation kann in verschiedenen Deployment-Modellen umgesetzt werden. Dazu wurden mehrere Varianten entwickelt, wie das Produkt umgesetzt werden kann. Die jeweiligen Varianten wurden einzeln und unabhängig von persönlichen Meinungen voneinander analysiert und objektiv dokumentiert. Die Vor- und Nachteile wurden aufgenommen, welche zu einem anschliessenden Entscheid führten.

Variante 1: 3-Tier Webapplikation

Idee: Die Applikation wird als Webanwendung auf einem Server betrieben. Einerseits gibt es ein Backend mit einer Datenbank, welches REST Anfragen entgegennimmt. Andererseits wird ein Frontend zur Verfügung gestellt. Das Rendering der Darstellung soll auf dem Client ausgeführt werden. Die Hochschule stellt einen Server zur Verfügung, welcher genügend Ressourcen bereitstellt.

Umsetzung: Diese Variante könnte mit verschiedenen Technologien umgesetzt werden. Als Frontend eignet sich React oder Angular. Als Backend Framework könnte Express oder FastAPI verwendet werden. Eine SQL-Datenbank kann für das Persistieren der Daten hinzugefügt werden. Alle Teile der Applikation werden als 3-Tier App umgesetzt. Die Auslieferung als Docker Container ist möglich, um Hardwareunabhängig zu bleiben.
Die Applikation kann über ein einfaches Docker-Compose Deployment betrieben werden. Falls das Produkt skaliert werden soll, ist eine Umsetzung mit Kubernetes oder Docker Swarm möglich. Dazu muss jedoch ein Loadbalancer hinzugefügt werden.

Vorteile:

- Einfache Architektur
- Fast keine Abhängigkeiten von Dritten
- Datenhoheit beim Betreiber der Applikation

Nachteile:

- Ressourcen müssen ständig zur Verfügung stehen
- Infrastruktur muss verwaltet werden
- Erhöhter Aufwand für die Nutzerverwaltung

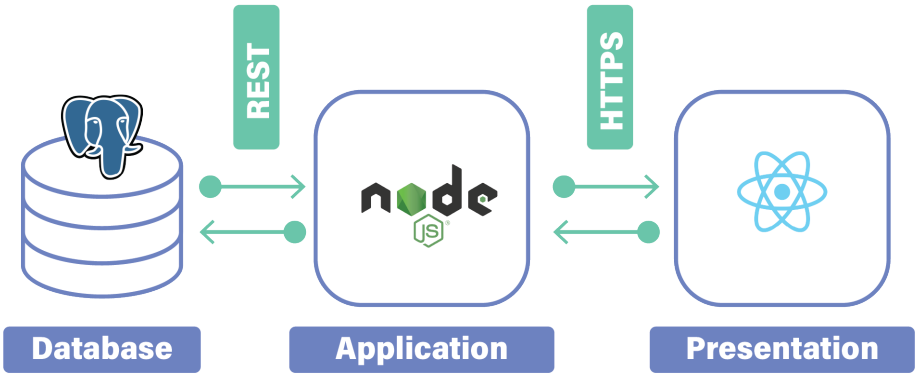


Abbildung 6.1.: Deploymentdiagramm Variante 1

Variante 2: Native App

Idee: Eine Idee ist es, eine Native Desktop Applikation zu entwickeln, welche der Benutzer der Applikation auf seinem Gerät installiert. Die Desktop Applikation kommuniziert mit einer API und tauscht Informationen aus. Besteht keine Verbindung zur API (Offline-Modus), so können die getätigten Arbeitsschritte lokal gespeichert werden, bis wieder in den Online-Modus gewechselt wird.

Umsetzung: Eine Möglichkeit wäre es, die Desktop Applikation mit .NET umzusetzen. Einerseits könnten so Windows wie auch MacOS und Linux Geräte abgedeckt werden. Das Backend könnte wie in Variante 1 mit einem Backend Framework wie Express oder FastAPI umgesetzt werden. Für das Persistieren der Daten kann ebenfalls eine SQL-Datenbank genutzt werden. Das Deployment besteht aus der Sicht des Betreibers nur aus dem Backend. Jedoch muss zusätzlich eine Webanwendung den Download der Desktop Applikation zur Verfügung stellen.

Vorteile:

- Offline-Funktionalität kann umgesetzt werden

Nachteile:

- Applikation muss installiert werden
- Zusätzliche Webseite für Download der Nativen App
- Benutzer muss Updates selber herunterladen / installieren
- Zusätzliche Implementationen für mobile Geräte benötigt

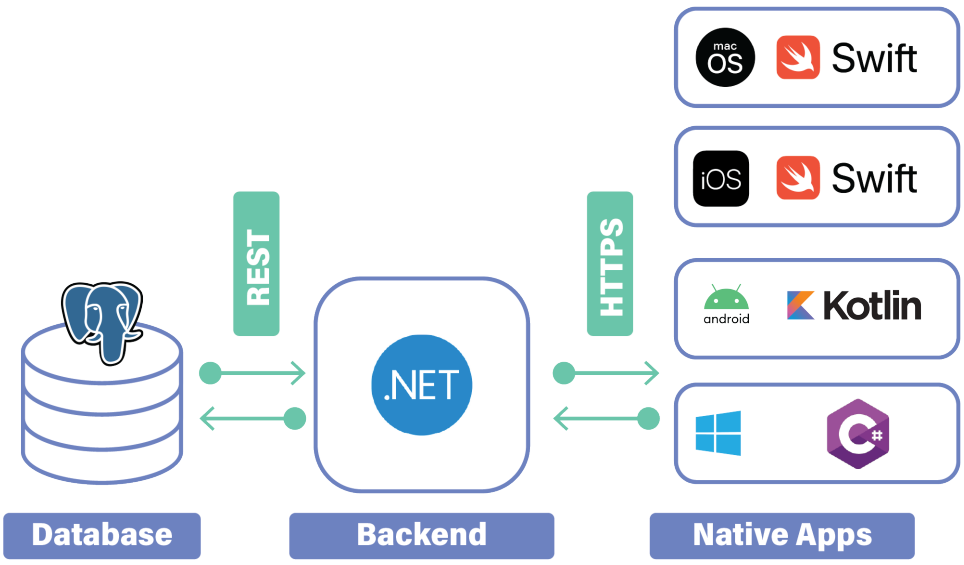


Abbildung 6.2.: Deploymentdiagramm Variante 2

Variante 3: Webapplikation in der Cloud

Idee: Hingegen zu den ersten beiden Varianten wird hier die Applikation in der Cloud betrieben. Die Applikation wird als Webanwendung ausgeliefert und läuft im Browser des Benutzers. Diese kommuniziert mit einer API, welche in einer Cloud betrieben wird. Die API kann entweder direkt auf die Datenbank schreiben oder führt mittels Serverless Functions den Code direkt aus.

Umsetzung: Mit einer dritten Variante könnte das Frontend wie in Variante 1 mit einem Framework wie React oder Angular umgesetzt werden. Im Unterschied wird das Backend mittels Serverless Functions nur bei Statusänderungen ausgeführt. Eine solche Lösung wird von verschiedenen Cloud Anbietern zur Verfügung gestellt. Zum Beispiel werden Serverless Functions in der AWS Cloud Lambda genannt, in der Microsoft Azure Cloud heissen sie Azure Functions.
Auch für das Deployment der Applikation wird von verschiedenen Cloudanbietern fertige Lösungen angeboten. Eine mögliche Umsetzung wäre in der Google Cloud mit Firebase oder in der Amazone Cloud mittels AWS Amplify. Weitere Dienste wie zum Beispiel die Nutzerverwaltung sind in diesen Lösungen bereits inkludiert und können in der Entwicklung verwendet werden.

Vorteile:

- Verfügbare Ressourcen
- Pay-as-you-go
- Verschiedene Dienste werden direkt zur Verfügung gestellt

Nachteile:

- Abhängigkeit vom Cloudanbieter
- Datenhoheit beim Cloudanbieter

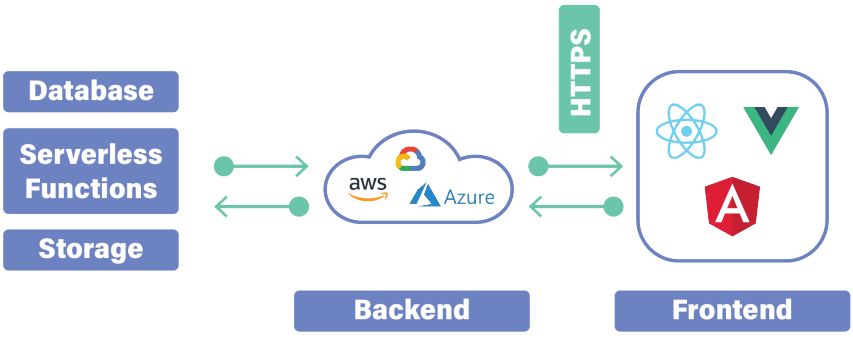


Abbildung 6.3.: Deploymentdiagramm Variante 3

Entscheid

Nach Auswertung der verschiedenen Varianten, entschied sich das Team für Variante 3 (Webapplikation in der Cloud). Die Vorteile überwiegen im Vergleich zu den andern beiden. Grund dafür war hauptsächlich die unbegrenzten Möglichkeiten, welche in der Cloud umgesetzt werden können.

Ein weiterer Vorteil dieser Variante ist auch die schnelle Entwicklung durch das Framework React, da hier bereits Erfahrung besteht.

6.4. Technologien

In den folgenden Abschnitten werden die verschiedenen Technologien welche als Frontend und Serverless Backend eingesetzt werden könnten erklärt und verglichen.

6.4.1. Frontend

Als Frontend wollen wir ein Javascript Framework verwenden. Es gibt diverse Optionen. Wir führen die für uns interessanten Optionen folgend auf und geben einen kurzen Überblick.

React JS

React JS erlaubt es auf Basis von Komponenten in Javascript oder Typescript eine Single Page Applikation zu entwickeln. Wir haben aus besuchten Modulen und persönlichen Projekten bereits gute Erfahrungen damit gemacht [3].

Next JS

Next JS basiert auf React JS und nennt sich *The React Framework for Production*. Es bietet viele Erweiterungen, welche Probleme von React lösen sollen. Next unterstützt zum Beispiel *Pre-rendering* was für SEO extrem wichtig ist. *File-system Routing* wird auch unterstützt, was das Arbeiten mit verschiedenen Routen auch vereinfacht, ohne noch eine Router Komponente zu implementieren [4].

Angular JS

Angular JS bietet ähnliche Funktionen wie React JS und basiert auf dem Konzept von wiederverwendbaren kleinen Komponenten. Allerdings ist Angular eher für grosse Applikationen ausgelegt und braucht viel Boilerplate Code. Mit Angular haben wir auch bereits Erfahrung [5].

6.4.2. Cloud Backend

Ein Backend in der Cloud hat viele Vorteile. Meistens bereitet die Verfügbarkeit und Skalierung in einer On-Premise Lösung grossen Aufwand. Durch die Auslagerung dieser Verantwortungen an einen Cloud Provider können die Ressourcen eher auf das Frontend fokussiert werden. Unter der grossen Auswahl an Cloud Anbieter evaluieren wir Google Cloud und AWS genauer. Diese Provider bieten je ein Serverless Backend an, welches Authentifizierung, Datenablage und Datenbankzugriffe vereinfachen sollen. Diese Services heissen Firebase (Google Cloud) und AWS Amplify und werden folgend genauer erläutert.

Google Firebase

Firebase bietet ein Serverless Backend mit vielen Funktionen, welche bis zu gewissen Limits sogar kostenfrei verwendet werden können. Die relevanten Funktionen und Services werden folgend erklärt.

- **Authentication**: Provider mit verschiedenen Möglichkeiten der Authentifizierung (Google, Facebook, Apple, Github, Microsoft etc.)
- **Firestore Database**: NoSQL-Datenbank mit Möglichkeit für Realtime Updates

- **Storage**: Speicherpool für Dateien, welche über eine API zugegriffen werden
- **Functions**: Serverless Funktionen, welche in Javascript verfasst werden

Grundsätzlich bietet Firebase eine breite Auswahl an Services, welche für eine Full Stack Applikation benötigt werden. Die Konfigurationsmöglichkeiten sind teilweise aber etwas eingeschränkt und nicht vollständig anpassbar.

AWS Amplify

AWS bietet eine sehr umfangreiche Cloud Umgebung für alle möglichen Typen von Applikationen. Der grosse Umfang macht AWS jedoch auch sehr komplex. Hier kommt AWS Amplify (ein Service von AWS) ins Spiel. AWS Amplify bietet die Funktionalität von eines Serverless Backend's an, ohne die gesamte Komplexität, aber vielen Funktionen der AWS Cloud.
Es werden folgende Funktionen angeboten:

- **Authentication**: Provider mit verschiedenen Möglichkeiten der Authentifizierung (Google, Facebook, Apple, Github, Microsoft etc.)
- **DataStore**: NoSQL-Datenbank mit Möglichkeit für Realtime Updates
- **Storage**: Speicherpool (AWS S3 Bucket) für Dateien via Amplify-SDK zugegriffen werden kann
- **Functions**: Lambda Functions können serverless eingesetzt werden und lassen auch die Kommunikation mit anderen AWS Services zu

AWS Amplify bietet also alle Funktionen, die man sich von einem Backend wünschen könnte. Die Erweiterbarkeit durch Lambda Functions, und somit die Erweiterbarkeit auf weitere AWS Services, ist ein wichtiger positiver Punkt.

6.4.3. Entscheid

Schlussendlich haben wir und für das Frontend Framework Next JS und AWS Amplify als Serverless Backend entschieden. Next JS erlaubt uns schnell und zielgerichtet unsere Applikation umzusetzen. AWS Amplify nimmt uns viel Arbeit ab, da viele Funktionen schon vorbereitet sind. Next wird von AWS Amplify auch unterstützt.

Teil III.

Software Dokumentation

KAPITEL 7

Architektur

7.1. Übersicht

In diesem Kapitel wird die gewählte und umgesetzte Architektur der gesamten Applikation dokumentiert.

7.2. Gesamtarchitektur Diagramm

Im folgenden Diagramm ist die Architektur grafisch dargestellt. Die gesamte Applikation wird in der AWS Cloud bereitgestellt. Über ein Content Deliviery Network (CDN) werden statische Dateien wie das Frontend dem Web-Client bereitgestellt.

Sobald der Client das Frontend vom CDN geladen hat, kann der Web-Client auf die Backend Ressourcen in der Cloud zugreifen.

Durch AWS Amplify wurden zwei komplett getrennte Umgebungen eingerichtet, **main** und **dev**. Diese werden für die Produktiv- und Entwicklungsumgebung verwendet. Die Ressourcen der beiden Umgebungen können so separat konfiguriert werden.

Der Web-Client kann nun über eine GraphQL Schnittstelle, bereitgestellt über die AWS AppSync API, auf Daten in der Datenbank zugreifen. Datenmutationen können entweder direkt über AWS AppSync auf die Datenbank geschrieben werden oder über eine Lambda Funktion ausgelöst werden. Über Lambda Funktionen können mehrere und komplexere Operationen durchgeführt werden. Für das Erstellen einer Organisation beispielsweise sind mehrere Schritte nötig. Diese können nur durch eine Lambda Funktion erledigt werden, da ein Organisationbenutzer standardmässig nicht auf alle nötigen Ressourcen die nötigen Rechte hat.

Dateien und Medien werden im S3 Bucket abgelegt. Dieser ist zuständig für die Speicherung von Daten. Über AWS Amplify können diese vom Client verwaltet werden.

Damit die Benutzer auch die nötigten Rechte erhalten, werden sie durch den Cognito Dienst berechtigt und authentifiziert. Dieser ist auch zuständig für die Verifizierung der E-Mail, Registrierung der Nutzer

und Verwaltung von Passwörtern und Benutzerdaten. Da dieser Service aber beispielsweise Daten wie ein Profilbildlink nicht abspeichern kann, wird bei der Registrierung durch eine Lambda Funktion in der Datenbank in der Personen Tabelle ein neuer Eintrag mit den zusätzlich nötigen Daten erstellt.

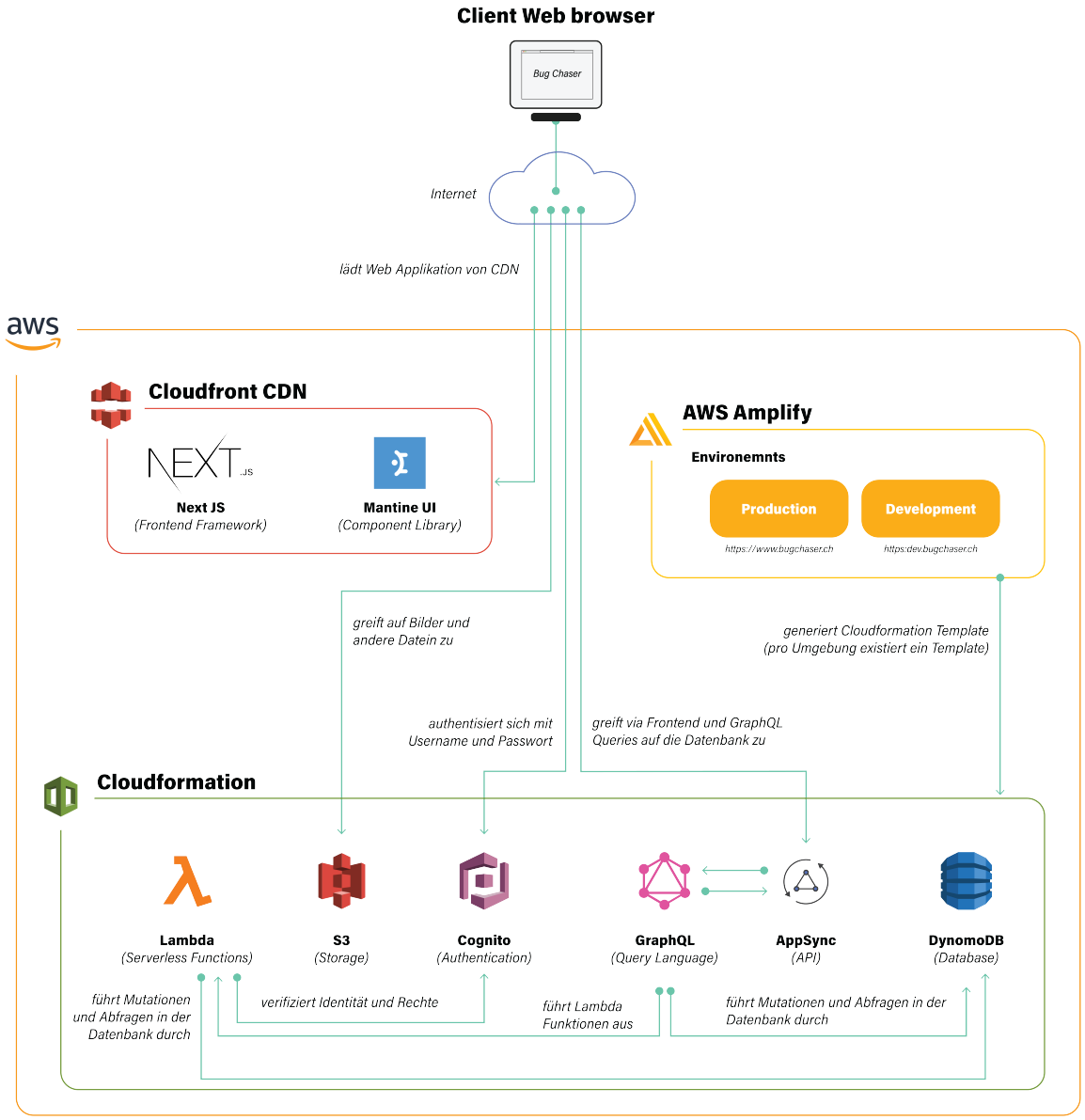


Abbildung 7.1.: Architektur Diagramm

7.3. Datenbank Schema

Die folgende Grafik zeigt den Aufbau der Datenbank als Visualisierung der GraphQL Schemas. Die dunkelblauen Objekte werden durch AWS Amplify in DynomoDB Tabellen umgewandelt und durch AWS AppSync ansprechbar. Grün markiert sind die Datentypen der Felder. Ist ein Feld als nicht optional definiert, wird dies mit einem roten Ausrufezeichen markiert. Die Syntax in der Visualisierung entspricht der GraphQL Syntax in AWS Amplify.

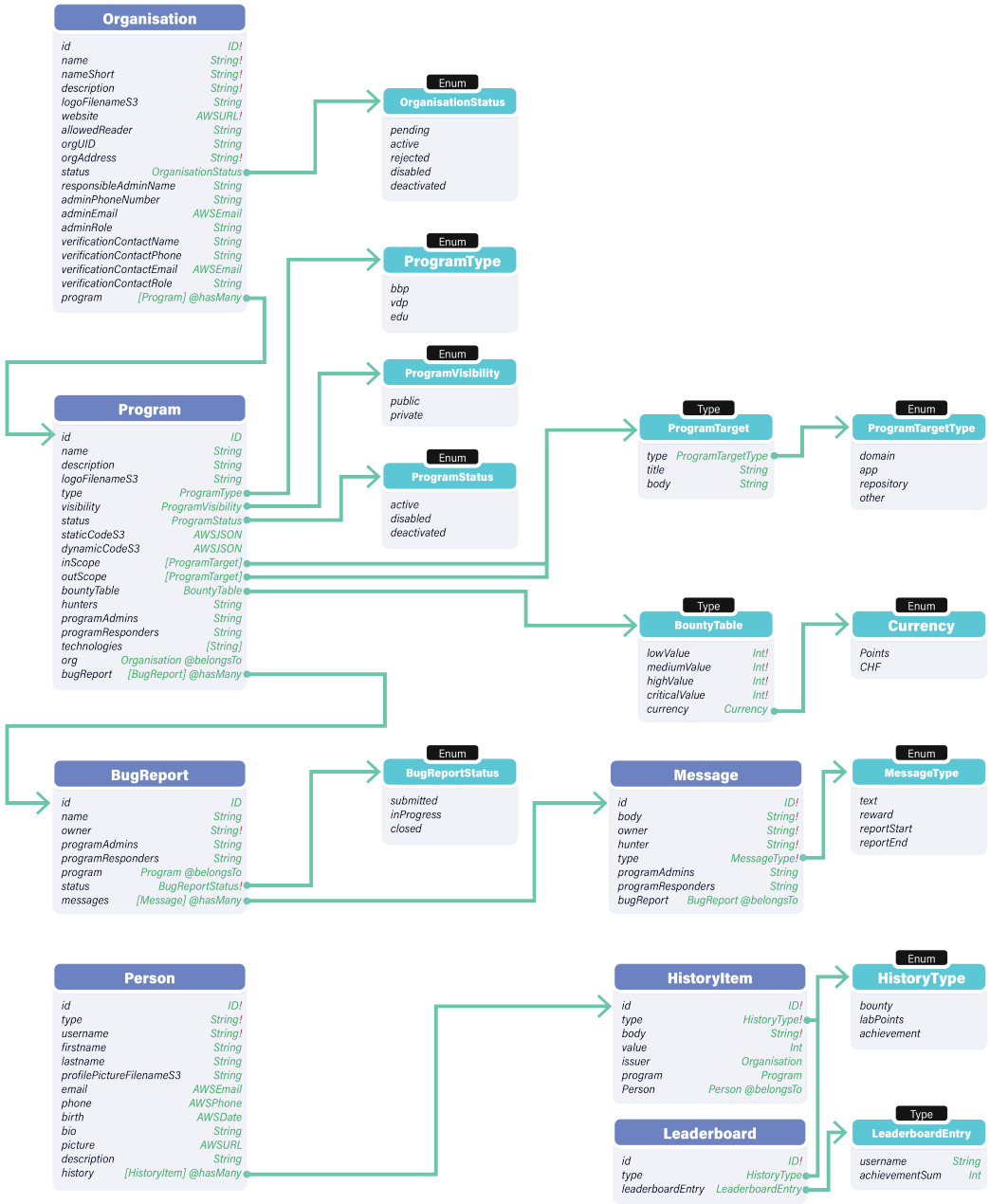


Abbildung 7.2.: GraphQL Schema Visualisierung

KAPITEL 8

Frontend

8.1. Übersicht

In diesem Kapitel wird auf das Frontend der Applikation eingegangen. Die getroffenen Entscheidungen werden dokumentiert und begründet.
Als Frontend Framework wird Next JS mit Mantine UI als Komponenten Bibliothek und Redux Toolkit als State Management System verwendet. Die werden folgend genauer beschrieben.

8.2. Verwendetet Technologien

8.2.1. Framework

Wie in der Evaluation beschrieben wird das Javascript Framework Next JS eingesetzt. Durch einen modularen Aufbau mit Komponenten lassen sich komplexe Web-Applikationen entwickeln.

8.2.2. Component library

Eine Komponenten Bibliothek ermöglicht es schnell und mit viel Flexibilität eine Web UI zusammenzustellen. Wir haben uns für **Mantine UI** entschieden. https://mantine.dev/
Die Library ist unter einer MIT Lizenz veröffentlicht und basiert auf TypeScript.
Die Bibliothek bietet diverse Komponenten wie Buttons, Listen, Inputfelder, Banner, Benachrichtigungen und mehr. Alle diese Komponenten können natürlich einzeln und aber auch kombiniert benutzt werden. Durch eine einheitliche Konvention von Mantine UI lassen sich diese auch nach Bedürfnis konfigurieren. Grösse, Farbe, Variante und viele andere Eigenschaften können angepasst werden.
Die Library ist unterteilt in verschiedene Pakete.

- **@mantine/hooks** → Hooks for state and UI management
- **@mantine/core** → Core Komponenten wie Buttons, Formularfelder etc.
- **@mantine/form** → Form management library
- **@mantine/dates** → Date inputs, calendars
- **@mantine/notifications** → Notifications system

- **@mantine/prism** → Code highlight with your theme colors and styles
- **@mantine/rte** → Rich text editor based on Quill.js
- **@mantine/dropzone** → Capture files with drag and drop
- **@mantine/modals** → Centralized modals manager
- **@mantine/spotlight** → Overlay command center

Die von uns verwendeten Package sind **Core**, **dropzone**, **form**, **hook**, **next** und **notifications**.

Die Dokumentation dieser Library ist sehr gut und ausführlich. Sie ist hier zu finden.

8.2.3. Icon Library

An diversen Orten der Applikation werden Icons eingesetzt. Damit diese einfach und gratis in eingesetzt werden, verwenden wir tabler icon (https://tabler-icons-react.vercel.app/). Auch diese sind unter einer MIT Lizenz verwendbar.

8.2.4. TypeScript

Javascript unterstützt nativ keine typsichere Entwicklung. TypeScript basiert auf Javascript, kann jedoch mit Typen verwendet werden. Next JS unterstützt TypeScript Out-of-the-Box.
Wir verwendet TypeScript um Fehler bezüglich Typen frühzeitig zu erkennen. Intelisense wird durch die Typen auch viel mächtiger und kann bessere Verschläge anbieten.

8.3. Konfiguration

8.3.1. Routen

Da die Applikation grundsätzlich in 3 Teile aufgeteilt wurde, widerspiegelt sich diese auch meistens in der Struktur der Pfade.

- **/admin/***
  Seiten für Plattform Admins: Verwalten von Organisationen, Programmen und Benutzern.
- **/hunter/***
  Seiten für Hunter: Alle Programme anzeigen, eigene Reports anschauen, Reports einreichen etc..
- **/orguser/***
  Seiten für Orguser: Eigene Organisationen anzeigen und verwalten, Programme anzeigen und verwalten, Reports verwalten etc..
- **/organisations/***
  Organisationen anzeigen
- **/programs/***
  Programme anzeigen

8.3.2. Linting

Um den Code in einer einheitlichen Formatierung und Syntax zu halten wird ESLint in Kombination mit Prettier verwendet. Beide Tools können durch eine Konfiguration im Root Verzeichnis angepasst werden. Die Konfigurationsdateien wurden leicht angepasst.

ESLint verwendet verschiedene Richtlinien um den Code so effizient und fehlerfrei wie möglich zu gestalten. Es weist zum Beispiel auf Bereiche im Code hin, welche gar nie erreicht werden.

```
{
    "env": {
        "browser": true,
        "es2021": true
    },
    "extends": ["plugin:react/recommended", "next", "google", "prettier"],
    "parser": "@typescript-eslint/parser",
    "parserOptions": {
        "ecmaFeatures": {
        "jsx": true
        },
        "ecmaVersion": 12,
        "sourceType": "module"
    },
    "plugins": ["react", "@typescript-eslint"],
    "rules": {
        "require-jsdoc": "off",
        "react/react-in-jsx-scope": "off"
    }
}
```

Listing 8.1: ESLint Konfiguration (.eslintrc.json)

Prettier ist hauptsächlich zuständig für die Formatierung des Codes. Die Zeilenlänge wird zum Beispiel auf 120 Zeichen beschränkt.

```
{
    "endOfLine": "auto",
    "printWidth": 120,
    "tabWidth": 2,
    "trailingComma": "es5"
}
```

Listing 8.2: Prettier Konfiguration (.prettierrc)

8.3.3. Dependencies

Je grösser das Projekt wächst, desto mehr Pakete werden für verschiedene Funktionen benötigt. Folgend wird kurz erklärt welche Pakete für welche Zwecke verwendet werden.

```
"@mantine/core": "^4.2.2",
"@mantine/dropzone": "^4.2.2",
"@mantine/form": "^4.2.2",
"@mantine/hooks": "^4.2.2",
"@mantine/next": "^4.2.2",
"@mantine/notifications": "^4.2.2",
"@reduxjs/toolkit": "^1.8.1",
"@tiptap/extension-character-count": "^2.0.0-beta.26",
"@tiptap/extension-placeholder": "^2.0.0-beta.48",
"@tiptap/react": "^2.0.0-beta.109",
"@tiptap/starter-kit": "^2.0.0-beta.184",
"@types/react-redux": "^7.1.24",
"aws-amplify": "^4.3.21",
"cookies-next": "^2.0.4",
"cypress": "^9.6.1",
"formik": "^2.2.9",
"next": "12.1.6",
"react": "18.1.0",
"react-dom": "18.1.0",
"react-redux": "^8.0.1",
"tabler-icons-react": "^1.47.0"
```

Listing 8.3: package.json (Ausschnitt)

```
Mantine UI
File Upload
Formulare
Mantike Hooks
Next JS + Mantine UI
Benachrichtigungen
Redux RTK
Text Editor Extension
Text Editor Extension
Text Editor
Text Editor Basic
typescript Redux
AWS Amplify
Cookies Next JS
Cypress (testing)
Formulare
Next JS Basic
React JS
React DOM
React + Redux
Icons
```

Listing 8.4: Erklärung package.json

8.3.4. State Management

Um Daten zwischen Komponenten zu teilen benötigt es eine State Management Lösung. Eine simple Implementation bietet React Context. In diesem Store können Daten abgelegt, modifiziert und wieder gelesen werden.

Eine weitere sehr verbreitete State Management Lösung ist Redux. Sie bietet mehr Möglichkeiten was die Konfiguration und Ausbaufähigkeit angeht. Allerdings wird viel Boilerplate Code benötigt, was die Lösung in den meisten Fällen unnötig komplex machen kann.

Redux Toolkit (RTK) verbindet die Modularität von Redux und die Simplizität von React Context. Durch sogenannte Slices kann der State von verschiedenen Teilen organisiert abgelegt und geteilt werden. Aktuell werden vier Slices verwendet (organisationSlice, profileSlice, programSlice, reportSlice). Diese kommen zum Zug, wenn ein Benutzer eine Organisation, Userprofil, Programm oder Report bearbeitet. So müssen die Daten nur einmal von der Datenbank geladen werden und werden während dem Wechseln von Tabs nicht erneut angefordert.

Für die Authentisierung wird React Context verwendet. Es bietet alle Möglichkeiten, die benötigt werden.

8.4. Security

8.4.1. Authentisierung

Die Authentisierung wird hauptsächlich durch AWS Amplify erledigt. Daten für die Authentisierung werden via JWT Token im LocalStorage des Browsers abgelegt. Im Frontend der Applikation können durch den AWS Amplify Import und Konfiguration Authentifizierungsfunktionen aufgerufen werden. Das Login beispielsweise könnte so aussehen.

```
1 import { Auth } from "aws-amplify";
2 await Auth.signIn(username, password);
```

Listing 8.5: Login Authentifizierung Beispiel

8.5. Verbindung zum Backend

Damit das Frontend weiss, welche Ressourcen im AWS Amplify Backend zur Verfügung stehen wird eine Konfiguration in der **_app.tsx** Datei mitgegeben. Für die lokale Entwicklung wird die Datei **aws-exports.js** generiert. Diese enthält die Schlüssel und nötigen Informationen, damit das Frontend auf die Backend Ressourcen zugreifen kann.

```
1 import Amplify from "aws-amplify";
2 import awsconfig from "../aws-exports";
3 Amplify.configure(awsconfig);
```

Listing 8.6: Amplify Konfiguration (app.tsx)

```
4 const awsmobile = {
5     "aws_project_region": "eu-central-1",
6     "aws_cognito_identity_pool_id": "eu-central-1:*****",
7     "aws_cognito_region": "eu-central-1",
8     "aws_user_pools_id": "eu-central-1_*****",
9     "aws_user_pools_web_client_id": "*****",
10    "oauth": {},
11    "aws_cognito_username_attributes": [],
12    "aws_cognito_social_providers": [],
13    "aws_cognito_signup_attributes": [
14        "EMAIL"
15    ],
16    "aws_cognito_mfa_configuration": "OFF",
17    "aws_cognito_mfa_types": [
18        "SMS"
19    ],
20    "aws_cognito_password_protection_settings": {
21        "passwordPolicyMinLength": 8,
22        "passwordPolicyCharacters": []
23    },
24    "aws_cognito_verification_mechanisms": [
25        "EMAIL"
26    ],
27    "aws_appsync_graphqlEndpoint":
          "https://*****.appsync-api.eu-central-1.amazonaws.com/graphql",
28    "aws_appsync_region": "eu-central-1",
29    "aws_appsync_authenticationType": "AMAZON_COGNITO_USER_POOLS",
30    "aws_appsync_apiKey": "*****",
31    "aws_user_files_s3_bucket": "bugchaserbucket*****-dev",
32    "aws_user_files_s3_bucket_region": "eu-central-1"
33 };
34
```

```
35 export default awsmobile;
```

Listing 8.7: AWS Exports (aws-exports.js)

8.6. Wichtige Komponenten und Funktionen

- `BasicPageFrame.tsx`
  Definiert das Layout der Seite und lässt einen Titel und Subtitel zu.
- `showCustomNotification(type: NotificationType, message?: ReactNode, title?: ReactNode)`
  Um dem user eine Benachrichtigung als Info, Warnung oder Fehler anzuzeigen kann diese Funktion verwendet werden. Sie erscheint im rechten unteren Ecken und kann einen Titel, eine Nachricht und einen Typ beinhalten. Der Typ kann folgende Werte haben: `success`, `error`, `warning`, `info`. Die Farbe der Benachrichtigung hängt vom verwendeten Typ ab.
- Konversionen von AWS Zeitstempel
  Da AWS ein eigenes Zeitstempelformat verwendet, müssen diese zuerst umgewandelt werden, damit diese leserlich dargestellt werden können. Da dies häufig vorkommt, finden sich im Dokument datetimeConversions.tsx mehrere Funktionen, welche ein leserliches Format zurückgeben.

8.6.1. Text Editor

Damit Nutzer der Plattform effizient einen Bug Report einreichen können, wurde eine Text Editor Komponente verwendet. Dieses Paket heisst tiptap und beschreibt sich selbst in den folgenden Worten:
The headless editor framework for web artisans.

Der Editor lässt sich einfach importieren und eine grosse Auswahl an Konfigurationsmöglichkeiten. Er lässt eine Markdown ähnliche Syntax zu. So lassen sich einfach und intuitiv Titel, Paragraphen und Codefragmente einbauen. Durch Module lässt sich die Funktionalität sogar noch mehr erweitern. Die Dokumentation ist sehr ausführlich und ist hier zu finden.

Abbildung 8.1.: TipTap Editor Beispiel

8.7. Funktionalität

8.7.1. Dark Mode

Die Applikation bietet einen Dark Mode an. Da alle Komponenten von Mantine UI eine helle und dunkle Version haben passen sich die diese perfekt an. Der benötigte Provider dafür heisst **ColorSchemeProvider** und ist in **@mantine/core** enthalten. Mantine legt den Theme im Cookie ab, damit nach einem Refresh der Seite das Theme immer noch stimmt.

8.7.2. Image Caching

Um die Dateien wie Profilbilder nicht immer wieder vom S3 Bucket anzufragen müssen, werden die Adressen der Bilder im LocalStorage abgelegt. Anfragen an eine Datei sollen daher über Funktionen in der Datei **S3actions.tsx** abgewickelt werden. Der Link, welcher vom S3 Bucket zur Verfügung gestellt wird, läuft nach einer gewissen Zeit ab. Nach dieser Zeitdauer wird automatisch ein neuer Link angefordert und im LocalStorage gespeichert.
Der Benutzer bemerkt von diesem Caching nichts. Optimaler wäre das Ablagen der gesamten Dateien im Browser Cache, so würde man noch weniger Ressourcen auf dem S3 verwenden. Mehr dazu aber im Kapitel Erweiterung.

KAPITEL 9

Backend

9.1. Übersicht

In diesem Kapitel werden die Funktionalitäten des Backends dokumentiert. Dazu werden die einzelnen Komponenten im jeweiligen Abschnitt beschrieben. Das Backend ist für die Speicherung und Bereitstellung von Daten zuständig.

9.2. AWS Amplify

Das Backend wird in der Cloud betrieben. Wie dem Kapitel Technologie-Evaluation entnommen werden kann, wurde die Amazon Web Services Cloud dafür ausgewählt. Die Amazon Web Services Cloud bietet über 200 verschiedene Dienste an und ist Führer im Cloud Infrastructure & Platform Services Bereich [6,7]. Um die Verwendung der Cloud zu vereinfachen wurde zusätzlich AWS Amplify verwendet. AWS Amplify ist ein Dienst, welcher das Erstellen von full-Stack Web sowie Mobile Applikationen vereinfacht. Dies ermöglicht einem Team, die Entwicklung sowie die Bereitstellung einer Applikation um einiges zu beschleunigen. AWS Amplify stellt eine CLI bereit, über welche direkt mit der Cloud interagiert werden kann. Zusätzlich kann AWS Amplify Studio genutzt werden, falls lieber mit einem GUI gearbeitet wird.

AWS Amplify bietet einiges an Funktionalität. Schlussendlich wird im Hintergrund ein Cloudformation Template generiert, welches wiederum die AWS Clouddienste enthält und bereitgestellt. Eine vereinfachte Übersicht kann der Abbildung Übersicht AWS Amplify entnommen werden.

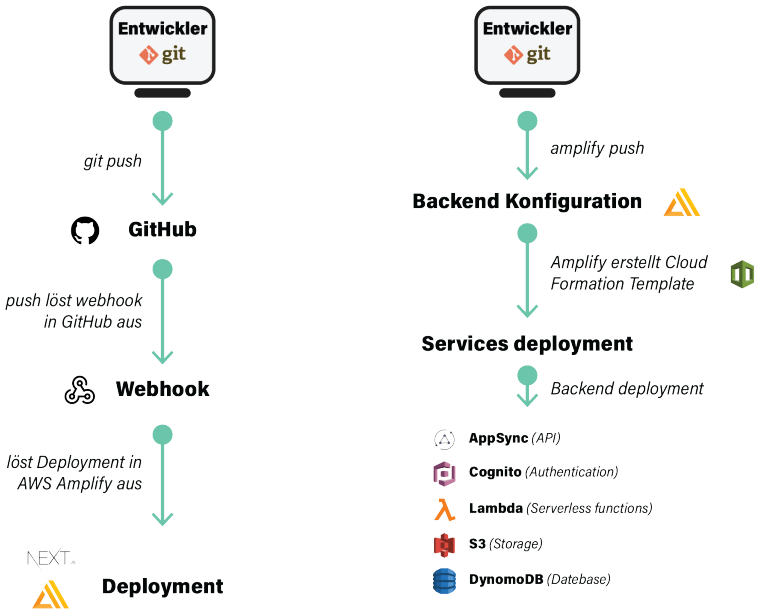


Abbildung 9.1.: Übersicht AWS Amplify

9.3. Schnittstellen

Ein Nutzer der Applikation möchte über das Frontend mit dem Backend interagieren. Dafür stellt AWS Amplify zwei verschiedene Möglichkeiten bereit, wie eine solche Schnittstelle implementiert werden kann. In den folgenden beiden Abschnitten werden diese genauer beschrieben und ihre Vor- und Nachteile aufgezeigt. AWS Amplify setzt jedoch die Limitierung, dass nur eine der Beiden verwendet werden kann. Zu Beginn eines Projektes muss daher entschieden werden, welche API verwendet werden soll. Ebenfalls werden nicht mehrere Schnittstellen vom gleichen Typ unterstützt. Dies wäre zum Beispiel für Schnittstellen zu Drittanbieter wünschenswert.

9.3.1. REST Schnittstelle

Eine der bekanntesten Schnittstellen, welche im heutigen Web verwendet wird, ist REST.

In AWS Amplify kann über die CLI eine neue REST API hinzugefügt werden. Im Hintergrund verwendet AWS Amplify den Amazon API Gateway. Dabei fallen dieselben Probleme an, welche bei jeder REST API auftreten, unabhängig vom deployment:

Under-fetching

Das größte Problem bei REST Schnittstellen ist die Tatsache, dass es mehrere Endpunkte gibt. Üblicherweise stellt jeder Endpunkt eine Ressource dar. Dies erfordert, dass die Benutzer mehrere Anfragen machen müssen, um alle gewünschten Daten zu erhalten. Ein Beispiel dafür ist folgende Abbildung Beispiel Under-fetching. Ein Nutzer möchte alle Bug Reports einer Organisation anzeigen. Mit einer REST Schnittstelle ist dies nur in drei Schritten möglich:

1. GET: Anfrage der Organisation über ihre ID
   Resultat: Ganzes Organisationsobjekt mit allen Identitäten aller Programmen

2. GET: Anfragen für jedes Programm über die jeweilige ID
   Resultat: Ganzes Programmobjekt mit allen Identitäten aller Bug Reports

3. GET: Anfragen für jeden Bug Report über die jeweilige ID
   Resultat: Ganzes Bug Report-Objekt

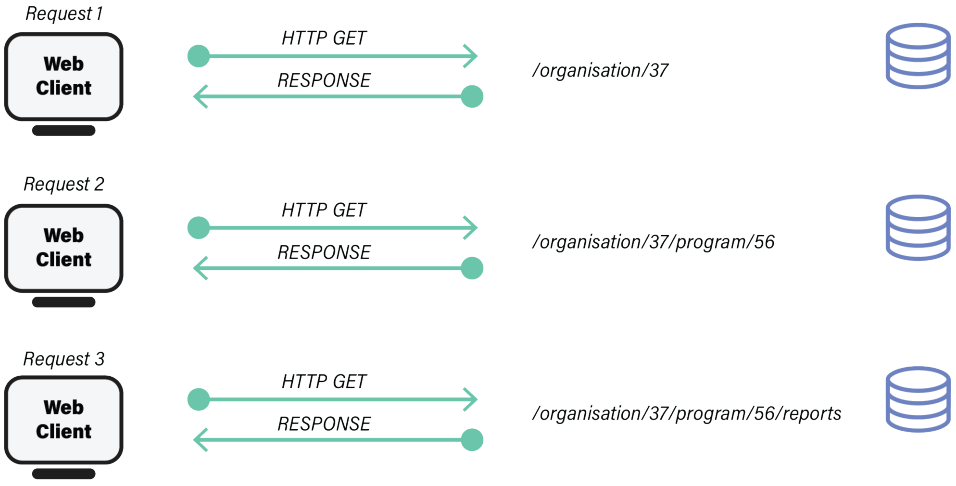


Abbildung 9.2.: Beispiel Under-fetching

Durch die unzähligen Anfragen vom Nutzer an den Server ist mit Leistungseinbussen zu rechnen, was die Applikation verlangsamt. Eine mögliche Lösung wäre es, direkt einen Endpunkt bereitzustellen, welcher alle Bug Reports einer Organisation zurückgibt. Dabei tauchen jedoch zwei weitere Probleme auf. Einerseits ist mit erhöhtem Aufwand zu rechnen, da alle möglichen „Abkürzungen" durch einen Entwickler implementiert werden müssen sowie andererseits das Over-fetching-Problem, welches im nächsten Abschnitt beschrieben wird.

Over-fetching

Bei einer REST Schnittstelle gibt es keine Möglichkeit, dem Server zu sagen, was genau benötigt wird. Eine Anfrage auf eine Ressource liefert so immer alle Daten zurück. Dies ist zwar nicht so problematisch wie das Under-fetching-Problem, kann jedoch ebenfalls zu längeren Ladezeiten der Daten führen.

Ein Beispiel dafür ist folgende Abbildung Beispiel Over-fetching. Um dem Under-fetching-Problem aus dem vorherigen Beispiel zu entgehen wurde eine zusätzliche Route erstellt, welche alle Reports einer Organisation mit allen Daten zurückgibt.

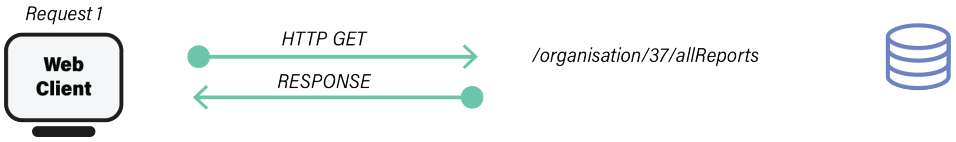


Abbildung 9.3.: Beispiel Over-fetching

Durch das Einführen einer neuen Route muss nur noch eine Anfrage gesendet werden, jedoch kann die Antwort des Servers sehr gross sein. Möchte nun der Benutzer nur den Titel des Reports anzeigen, so müsste das Resultat erneut gefiltert werden, und ein grosser Teil der erhaltenen Daten verworfen werden.

9.3.2. GraphQL Schnittstelle

GraphQL wurde im Jahr 2012 von Facebook entwickelt, ist seit 2015 Open Source und wird heute von der GraphQL Foundation verwaltet. [8] GraphQL wurde entwickelt, um APIs schnell, flexibel und entwicklerfreundlich zu machen. Dem Benutzer die Möglichkeit geboten, mit dem Server zu kommunizieren und abzusprechen, welche Daten benötigt werden. AWS Amplify unterstützt nebst einer REST Schnittstelle auch GraphQL. Im Hintergrund verwendet AWS Amplify den AWS AppSync Service.

Durch die Kommunikation zwischen Client und Server wird es ermöglicht, nur die Daten zu erhalten, welche auch gewünscht sind. Der Benutzer spricht immer eine `/graphql` Ressource an, kann jedoch direkt verschachtelte Attribute abfragen.

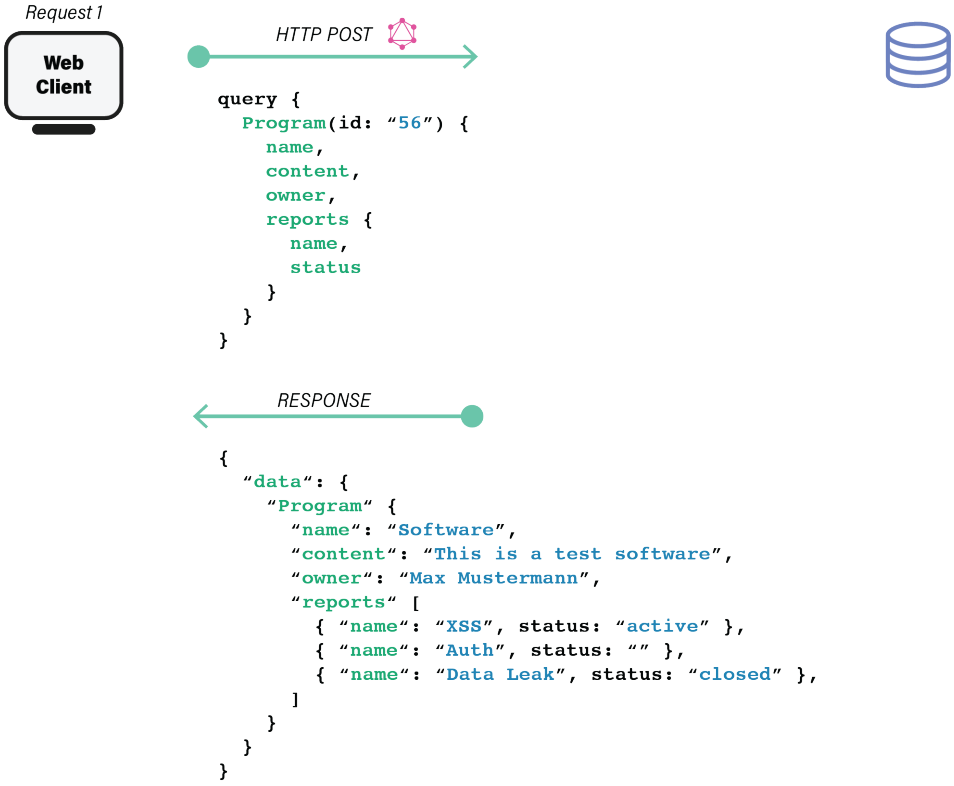


Abbildung 9.4.: Beispiel GraphQL Abfrage

Ebenfalls ist es möglich, direkt zu auf der Server Seite zu filtern, sodass der Benutzer nur das erhält, was er auch erhalten möchte. In diesem Beispiel nur die Bug Reports, welche neu eingereicht wurden.

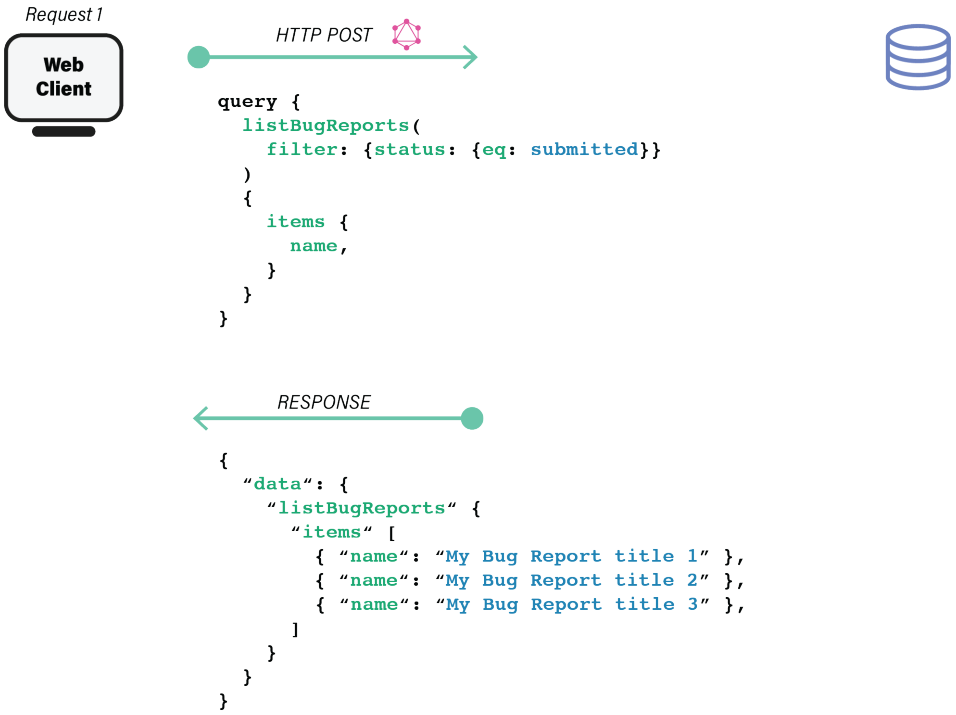


Abbildung 9.5.: Beispiel GraphQL Abfrage mit Filter

9.3.3. Definition der Schnittstelle

Aufgrund der Nachteile von REST wurde entschieden, GraphQL als Schnittstelle zum Backend zu verwenden. Die GraphQL Schnittstelle wird in einem Schema definiert, welches dann durch AWS AppSync bereitgestellt wird. Das Schema ist in der Datei `schema.graphql` definiert.

Ein vereinfachtes Beispiel eines Eintrages im Schema sieht wie folgt aus:

```
type Person {
  id: ID!
  username: String!
  age: Int
}
```

Listing 9.1: Beispiel GraphQL

AWS Amplify generiert daraus automatisch alle Abfragen, welche anschliessend im Frontend importiert und verwendet werden können. Im Frontend können jedoch auch eigene Abfragen geschrieben werden, um dem under- und over-fetching-Problem zu entgehen.

9.3.4. Verwendung der Schnittstelle

Die GraphQL Schnittstelle bietet drei unterschiedliche Möglichkeiten, mit ihr zu interagieren: Queries, Mutations und Subscriptions.

Query

Eine Query kann abgesetzt werden, um Daten zu lesen, ohne sie zu bearbeiten. In der Applikation werden Queries entweder direkt von AWS Amplify generiert oder im Frontend geschrieben.

Mutation

Mutationen werden verwendet, um Datensätze zu erstellen, bearbeiten oder löschen. Zudem können aktualisierte Daten oder Fehler direkt zurückgegeben werden. Die Serverless Functions werden ebenfalls über Mutationen ausgelöst.

Subscription

Ein Benutzer kann sich auf Änderungen an Objekten anmelden. Dies wird mit Subscriptions umgesetzt. Der Vorteil ist es, dass der Server automatisch den Client mit neuen Daten benachrichtigt, sobald es Änderungen gibt.

9.3.5. AppSync Client

Die GraphQL Schnittstelle kann nicht nur über die API angesprochen werden, welche AWS zur Verfügung stellt, sondern auch über einen Query-Builder. Dies ist sehr hilfreich, um das Backend manuell zu testen. Die gewünschten Abfragen können entweder zusammengeklickt oder über einen Editor selber geschrieben werden. Zudem werden die Abfragen im Scope des Benutzers ausgeführt, sodass auch die Authentisierung geprüft werden kann.

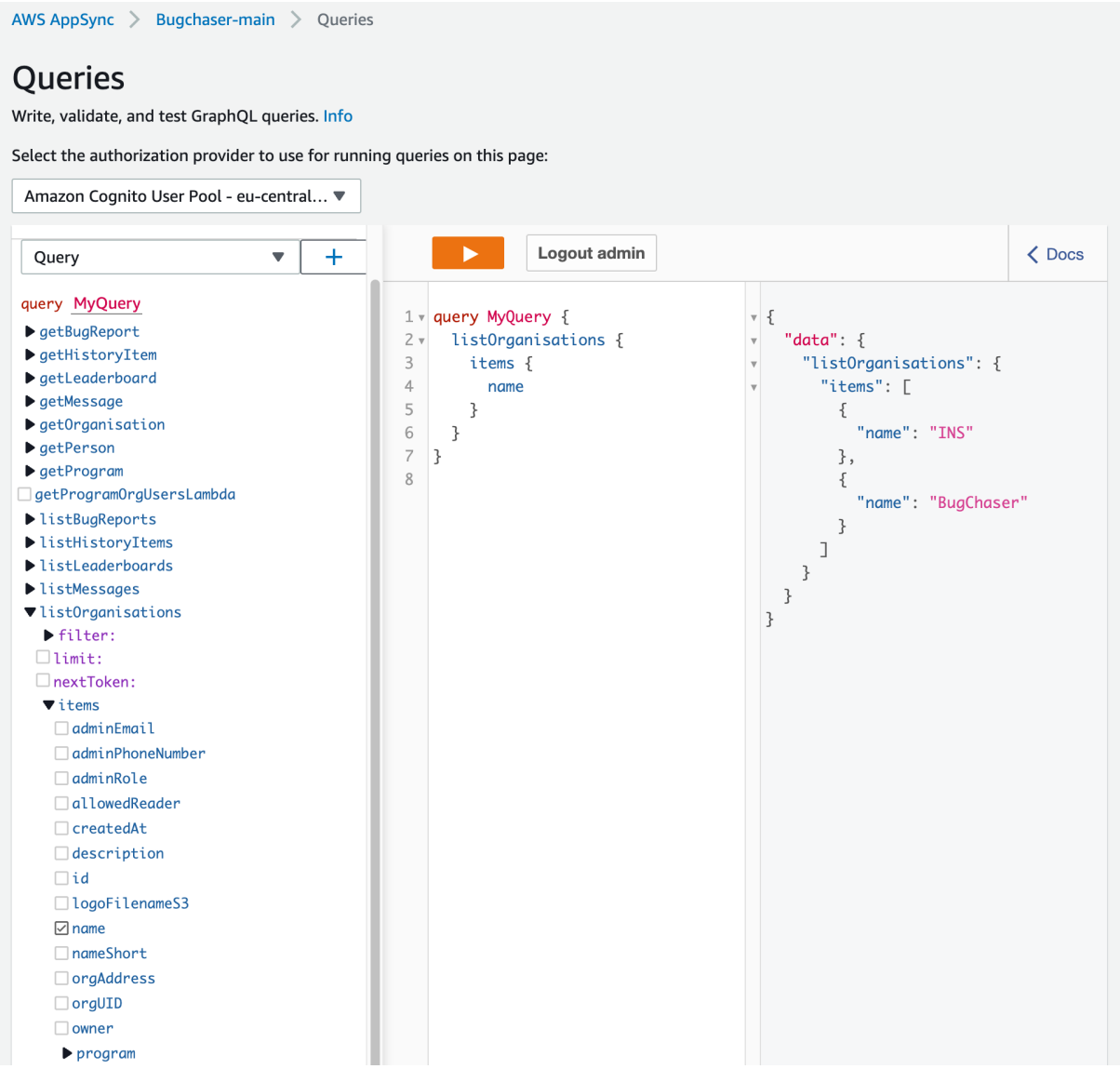


Abbildung 9.6.: AppSync Queries

9.4. Serverless functions

Im Abschnitt Nichtfunktionale Anforderungen wurde die Anforderung an die Applikation gesetzt, dass sie möglichst Ressourcen arm betrieben werden kann, um Kosten einzusparen. Wird eine Applikation in der Cloud betrieben, so erlaubt dies einem, das Backend nur dann auszuführen, wenn es benötigt wird. In der AWS Cloud wird dies mit dem Lambda Service möglich gemacht.

AWS Lambda ist ein serverless, ereignisgesteuerter Rechenservice, mit dem Code für praktisch jede Art von Anwendung ausgeführt werden kann, ohne einen Server zu verwalten. Abgerechnet wird dann nur die Zeit und Ressourcen, welche für die Funktion benötigt wurden. Dies ermöglicht es, das Backend sehr einfach zu skalieren. Mit AWS Lambda können wiederum die Dienste der AWS Cloud selbst verwendet werden, was zu unbegrenzten Möglichkeiten an Funktionalität führt.

In der Bug Chaser Applikation werden JavaScript Funktionen vom Lambda Service ausgeführt. Diese kommen hauptsächlich für die Datenmodellierung zum Einsatz. Ein Beispiel dafür ist das Erstellen eines neuen Bug Bounty Programms. Die Lambda Funktion erstellt einerseits neue Daten in der Datenbank, muss aber auch im Autorisierung-Dienst Cognito neue Gruppen für die Berechtigungen auf das Programm erstellen.

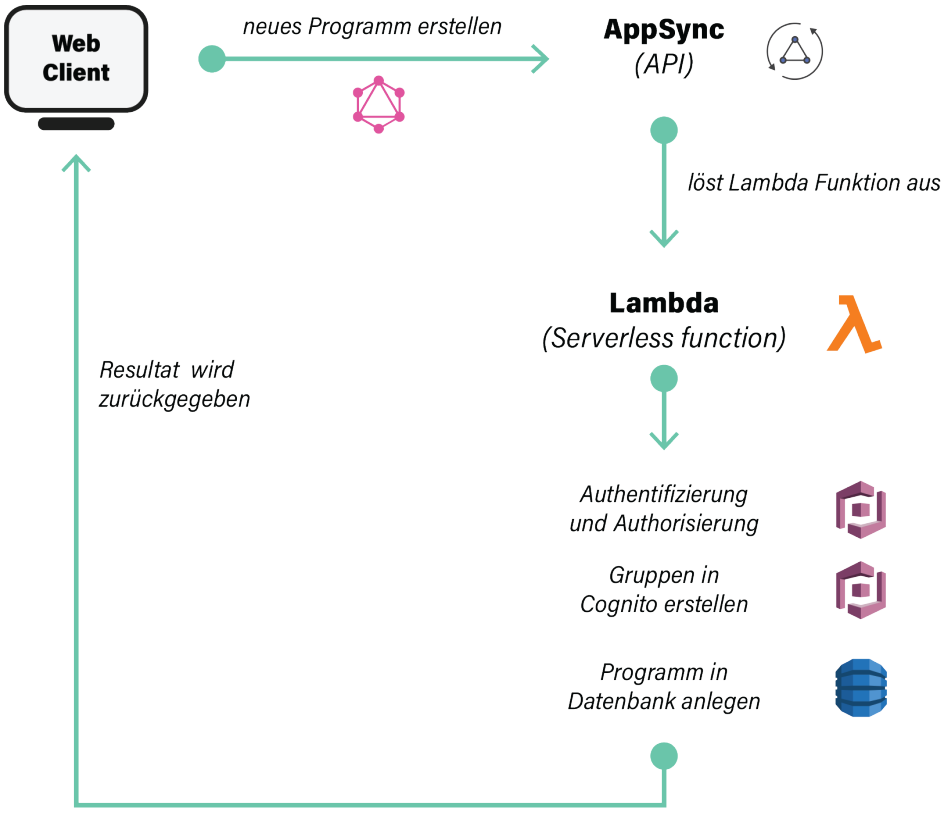


Abbildung 9.7.: AWS Lambda Ablauf

9.4.1. Auslösung einer Lambda Funktion

Wie der Abbildung entnommen werden kann, werden die Lambda Funktionen über die GraphQL Schnittstelle angesprochen. Die API prüft bei jedem Aufruf, ob alle benötigten Parameter mitgegeben werden. Ansonsten wird ein Fehler zurückgegeben. Lambda Funktionen können einerseits einzelne Parameter entgegennehmen oder ganze Objekte. Einfache Funktionen, wie zum Beispiel das Beitreten zu einem Program wird über einzelne Parameter gelöst. Weitere Information über den Nutzer können in der Lambda Funktion direkt aus dem Event-Objekt entnommen werden. Komplexere Funktionen wie das bereits genannte Erstellen eines Programms wird mit einem Objekt als Parameter ausgeführt.

Um eine Lambda Funktion mit einem Objekt als Input zu verwenden, muss das Objekt selbst zuerst als Input definiert werden.

```
347 input ProgramInput {
348   name: String!
349   description: String
350   visibility: ProgramVisibility!
351   # ...
```

Listing 9.2: GraphQL Schema: Ausschnitt der Definition eines Inputobjekt

Darauffolgend kann das Inputobjekt dann für eine Lambdafunktion verwendet werden.

```
296 createProgramLambda(program: ProgramInput): Int
297     @function(name: "createProgram-${env}", region: "eu-central-1")
298     @auth(rules: [{ allow: groups, groups: ["orgusers"] }])
```

Listing 9.3: GraphQL Schema: Definition Lambda Funktion

Mit ein paar Ausnahmen gibt die Lambda Funktion immer einen Status 0 vom Typ Int zurück, falls die Funktion erfolgreich ausgeführt werden konnte. Bei einem Fehler wird ein Fehlerobjekt zurückgegeben, welches beschreibt, wieso die Aktion nicht ausgeführt werden konnte. Zum Beispiel, falls der Nutzer keine Berechtigung hat, die Funktion auszuführen oder eine Ressource nicht zur Verfügung steht.

9.4.2. Erstellen von Lambda Funktionen

Lambda Funktionen können über die AWS Amplify CLI erstellt werden. Als Runtime wird in der Bug Chaser Applikation immer NodeJS verwendet.

```
mzindel@MFDINT-NBK001$ amplify add function

? Select which capability you want to add: Lambda function (serverless function)
? Provide an AWS Lambda function name: MyLambdaFunction
? Choose the runtime that you want to use: NodeJS
? Choose the function template that you want to use: Hello World

Available advanced settings:
- Resource access permissions
- Scheduled recurring invocation
- Lambda layers configuration
- Environment variables configuration
- Secret values configuration

? Do you want to configure advanced settings? Yes
? Do you want to access other resources in this project from your Lambda function? Yes
? Select the categories you want this function to have access to. storage
? Storage has 8 resources in this project. Select the one you would like your Lamb
da to access: Person:@model(appsync)
? Select the operations you want to permit on Person:@model(appsync) read

You can access the following resource attributes as environment variables from your
    Lambda function
        API_BUGCHASER_GRAPHQLAPIIDOUTPUT
        API_BUGCHASER_PERSONTABLE_ARN
        API_BUGCHASER_PERSONTABLE_NAME
        ENV
        REGION

? Do you want to invoke this function on a recurring schedule? No
? Do you want to enable Lambda layers for this function? No
? Do you want to configure environment variables for this function? No
? Do you want to configure secret values this function can access? No
? Do you want to edit the local lambda function now? No

Successfully added resource MyLambdaFunction locally.
```

Listing 9.4: Erstellen von Lambda Funktionen

Beim Erstellen einer Lambda Funktion können zudem Berechtigungen gesetzt werden, auf welche die Funktion Zugriff hat. AWS Amplify unterstützt Berechtigungen auf die Ressourcen der folgenden Typen:

- `function`
  Auslösen von anderen Lambda Funktionen (Mehrere Funktionen können ausgewählt werden)
- `auth`
  Zugriff auf einen Userpool Cognito (create, read, update, delete)
- `api`
  Abfragen an die API senden (Auswahl zwischen Queries, Mutations und Subscriptions)
- `storage`
  Datenmodellierung in der Datenbank oder auf einem S3 Bucket (Mehrere Datenbanktabellen oder Buckets mit jeweiligen create-, read-, update- und delete-Rechten)

In der Datei `<LambdaFunctionName>-cloudformation-template.json` werden die Berechtigungen für andere AWS Services gespeichert. Sollen weitere AWS Dienste angesprochen werden, welche nicht von AWS Amplify unterstützt werden, so können diese dort ergänzt werden. Für die Bug Chaser Plattform war dies jedoch nicht nötig.

9.5. Datenbank

Als Datenbank wird die Amazon eigene AWS Dynamo Database verwendet. DynamoDB ist ein NoSQL-Datenbankservice, welcher eine niedrige Latenzzeit und hohe Skalierbarkeit bietet. DynamoDB kann direkt mit AWS Amplify verwendet werden.

9.5.1. Datenbankschema

Amazon vereinfacht die Definition des Datenbankschemas für die Entwickler einer Applikation sehr. Das Datenbankschema wird direkt in GraphQL definiert. Durch das Hinzufügen eines `@model` Directives, wird der zuvor definierte GraphQL Type direkt auch als Tabelle in der DynamoDB angelegt.

Ein vereinfachtes Beispiel für einen Eintrag im Schema sieht wie folgt aus:

```
type Person @model {
  id: ID!
  username: String!
  age: Int
}
```

Listing 9.5: Beispiel Datenbankschema

Attribute haben einen von 14 Datentypen. Durch ein `!` müssen die Attribute immer definiert sein. Der Wert `null` kann so nicht zugewiesen werden. In AWS AppSync sowie in AWS DynamoDB sind folgende Scalar Types verfügbar:

- ID
  Ein eindeutiger Key für ein Objekt. Vergleichbar zu Primary-Key in SQL-Datenbanken.
- String
  UTF-8 character Sequenz.
- Int
  Eine Zahl zwischen $-(2^{31})$ und $2^{31} - 1$
- Float
  Ein IEEE 754 Gleitkommawert
- Boolean
  `true` oder `false`
- AWSDate
  Datum im Format `YYYY-MM-DD`
- AWSTime
  Zeit im Format`hh:mm:ss.sss`
- AWSDateTime
  Datum und Zeit im Format `YYYY-MM-DDThh:mm:ss.sssZ`
- AWSTimestamp
  Ein Int Wert, welcher die Anzahl Sekunden vor oder nach `1970-01-01-T00:00Z` angibt
- AWSEmail
  E-Mail Adresse im Format `example@bugchaser.ch`
- AWSJSON
  Generisches JSON Objekt
- AWSPhone
  Telefonnummer, wird jedoch als String gespeichert und nicht validiert
- AWSURL
  URL im Format `https://www.bugchaser.ch/path/` oder `mailto:example@bugchaser.ch`
- AWSIPAddress
  IPv4 oder IPv6 Adresse

9.5.2. One-to-Many Beziehungen

In der Bug Chaser Applikation werden einige one-to-many Beziehungen im Datenbankmodell abgebildet. Beziehungen werden mit den directives `@hasMany` und `@belongsTo` abgebildet.

```
52 type Organisation @model {
53   id: ID!
54   name: String!
55   program: [Program] @hasMany
56 }
```

Listing 9.6: Ausschnitt Datenbankschema

```
141 type Program @model {
142   id: ID!
143   name: String!
144   org: Organisation @belongsTo
145 }
```

Listing 9.7: Ausschnitt Datenbankschema

Die Einträge in den Tabellen werden über den Primärschlüssel referenziert. Dies ermöglicht verschachtelte Abfragen über die GraphQL Schnittstelle.

9.6. Datei-Speicherung

In AWS, die Datei-Speicherung erfolgt über den Service S3 Bucket. Dieser lässt es zu Dateien hoch- und herunter zu laden. Die Dateien sind in verschiedenen Buckets organisiert. Ein Bucket ist immer an einen geografischen Standort von AWS gebunden und kann eigene Richtlinien wie Verschlüsselung, Versionierung und Berechtigungen haben.

Aktuell können in verschiedenen Bereichen der Applikation Dateien hoch- und heruntergeladen werden. Pro Program können zwei Dateien hinterlegt werden (Statische und dynamische Code Dateien). Im Backend werden diese Dateien in einem S3 Bucket abgelegt.

9.6.1. Limitationen AWS Amplify

Durch AWS Amplify treten einige Limitationen in Kraft, welche die Verknüpfung und Benutzung zum S3 Bucket erschweren. AWS Amplify unterstützt nur einen S3 Bucket. Das bedeutet, dass alle Dateien in einem Bucket abgespeichert werden müssen. Dies ist grundsätzlich nicht unbedingt ein Nachteil, jedoch macht es teilweise Sinn, verschiedene Typen von Dateien in verschiedene Buckets zu speichern. Zum Beispiel ein Bucket für die Profilbilder, ein Bucket für Programm Dateien und ein Bucket für Bilder der Webseite. Eine weitere Limitation ist, dass durch die AWS Amplify API auf den S3 Bucket zugegriffen werden muss. So sind nicht alle Funktionen verfügbar, welche vom regulären S3 Bucket Service angeboten werden.

Da die Sicherheit der Daten potenziell sehr hoch ist, muss versichert werden, dass die Dateien nur von berechtigen Personen bzw. Gruppen zugegriffen werden können. Mit dem Zugriff auf den S3 Bucket lässt sich das über AWS Amplify aber so nicht lösen, denn es werden nur die folgenden drei Levels an Berechtigungen unterstützt.

- **public:** zugänglich für alle Benutzer der Applikation
- **protected:** Leserecht für alle Benutzer der Applikation, Schreibrecht nur für den Ersteller (Owner) der Datei
- **private:** Nur Lese- und Schreibrecht für den Ersteller (Owner) der Datei

Damit auf Dateien vom Level **protected** zugegriffen werden kann muss man entweder Ersteller (Owner) sein, oder den Dateinamen und die IdentityId (eindeutige Identifikations UID eines Users) des Erstellers kennen.

Wie sich in den oben erklärten Levels zu erkennen gibt, passt keines dieser Levels zu den Szenarien, da wir die Rechte aufgrund von Cognito Gruppen zuteilen müssten.

9.6.2. Limitationen AWS S3

Auch die Implementation des S3 Bucket lässt im Bereich von Berechtigungen etwas zu wünschen übrig. Die Berechtigungen über Cognito Gruppen zu verwalten, lässt sich auch hier nicht konfigurieren.

9.6.3. Lösung für die Dateiablage

Für die Dateiablage wurde eine funktionale, jedoch nicht optimale und nicht unbedingt sichere Variante ausgewählt.

Profilbilder und Logos von Organisationen und Programmen

Diese Dateien sind grundsätzlich für alle User sichtbar, daher passt das Level **protected** ziemlich gut. Als Beispiel das Profilbild des Benutzers. Es darf von allen Benutzern gelesen und vom Ersteller (Owner) zusätzlich noch bearbeitet werden. Allerdings kann ein Logo eines Programms von mehreren Benutzern angepasst werden. Mit dem Level protected ist das eigentlich nicht möglich. Nun legen wir die Daten aber trotzdem als protected ab (Logos, Profilbilder und Programm Code Dateien). In der Datenbank wir der Dateiname, IdentityId des Erstellers und Pfad im S3 Bucket abgelegt.

Das bedeutet folgendes: Ein Benutzer, welcher nicht in einem Programm eingeschrieben ist, könnte theoretisch auf Dateien eines Programms zugreifen, auf welches er keine Rechte hat, solang er den Dateinamen, IdentityId und Pfad im S3 Bucket irgendwie erlangt. Es bedeutet auch, dass wenn ein Programm zuerst aktiv ist, dann deaktiviert wird, die Dateien theoretisch immer noch von den Benutzern zugegriffen werden können. Diese Implementation und Konfiguration sollte in Hinsicht auf einen produktiven Einsatz mit sensitiven Daten überdacht und angepasst werden.

9.7. Zugriffsberechtigung

9.7.1. Authentisierung

Authentisierung bezeichnet das Nachweisen einer Identität. In der Bug Chaser Applikation wird dies mit Cognito umgesetzt. Cognito ist der Authentisierungsservice, welcher von AWS zur Verfügung gestellt wird. In einem Userpool befinden sich alle Nutzerobjekte, welche sich bereits registriert haben. Nach dem erfolgreichen Bestätigen der E-Mail Adresse wird ein Benutzer zusätzlich in eine Gruppe zugewiesen. Es gibt drei Gruppen:

- Orgusers
  Alle Nutzer, welche Programme oder Organisationen verwalten gehören zu dieser Gruppe. Die Nutzer werden automatisch zur Gruppe hinzugefügt.
- Hunters
  Benutzer, die auf der Plattform ihre gefundenen Bugs rapportieren möchten, gehören zur Gruppe Hunters. Die Nutzer werden ebenfalls automatisch zur Gruppe hinzugefügt.
- Admins
  Nutzergruppe für Administratoren. In diese Gruppe werden keine Benutzer automatisch zugewiesen. Dies ist nur manuell über die Cloud möglich.

Der Zugriff auf Programme oder Organisationen muss dynamisch entfernbar sein, falls zum Beispiel der Typ auf privat oder der Status auf deaktiviert gesetzt wird. Dazu werden pro Programm weitere Gruppen erstellt. Diese Gruppe beinhalten alle Hunters, für den Fall, dass das Programm privat ist. Ebenfalls gibt es jeweils eine Gruppe für Benutzer mit Responder oder Admin rechten pro Programm. Mit einer Lambda Funktion können Nutzer dann aus den Gruppen gelesen, hinzugefügt oder entfernt werden.

Die Authentisierung wird von Cognito über den Nutzernamen und des dazugehörigen Passworts geprüft. Eine E-Mail Adresse kann mehrmals verwendet werden. Dies wurde bewusst so umgesetzt, damit zum Beispiel ein Betreuer einerseits ein Hunter- sowie ein Organisations-Konto mit derselben E-Mail Adresse besitzen kann.

Eine Authentisierung über externe Anbieter wie Google, Apple oder Facebook wurde bisher noch nicht umgesetzt. Eine Multi-Faktor-Authentisierung wurde ebenfalls noch nicht implementiert, wird aber in Cognito unterstützt. Eine mögliche Erweiterung wäre es dies für die Benutzer anzubieten.

9.7.2. Autorisierung

Unter Autorisierung wird der Prozess verstanden, welcher während der Prüfung der Berechtigungen für den Zugriff auf eine Ressource durchgeführt wird. Amazon Cognito erstellt während der Authentisierung ein JWT für den Benutzer aus. Darin enthalten sind Informationen über den Benutzer sowie dessen Gruppen enthalten.

Bei einem Zugriff auf die GraphQL Schnittstelle wird geprüft, ob der Benutzer berechtigt ist, auf die von ihm angefragte Ressource zuzugreifen.

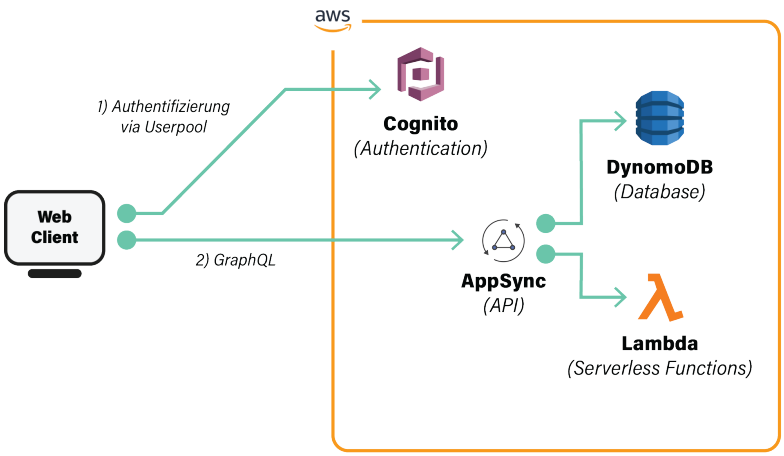


Abbildung 9.8.: AWS Coginito

Bei komplexeren Prüfungen, ob ein Nutzer berechtigt ist, eine Aktion durchzuführen werden ebenfalls AWS Lambda Funktionen zur Autorisierung herbeigezogen.

9.7.3. Definition der Berechtigungen im GraphQL Schema

Berechtigungen auf Ressourcen können auf ein ganzes Objekt sowie zusätzlich auf einzelne Attribute gesetzt werden. Dies wird mit dem `@auth` directive umgesetzt. Im folgenden Beispiel erhalten alle Nutzer, welche sich registriert haben leserechte auf die id sowie den Nutzernamen. Um die Privatsphäre zu schützen, werden Attribute wie zum Beispiel die E-Mail Adresse oder die Telefonnummer nicht für alle Nutzer erreichbar gemacht.

```
type Person
  @model
  @auth(rules: [{ allow: private, operations: [read] }]) {
    id: ID!
    username: String!
    email: AWSEmail @auth(rules: [{ allow: owner, operations: [read] }])
```

Listing 9.8: Ausschnitt GraphQL Schema

Die Berechtigungen können direkt über das Terminal validiert werden. Für jeden Benutzertyp (z.B. private, owner, admin, etc.) wird eine Matrix generiert, welche die Rechte auf die Attribute abbildet. Dies kann wie folgt aussehen:

```
mzindel@MFDINT-NBK001$ amplify status api -acm Person

userPools:private
┌──────────────────────────┬──────────┬──────────┬──────────┬──────────┐
│ (index)                  │ create   │ read     │ update   │ delete   │
├──────────────────────────┼──────────┼──────────┼──────────┼──────────┤
│ id                       │ false    │ true     │ false    │ false    │
│ type                     │ false    │ true     │ false    │ false    │
│ username                 │ false    │ true     │ false    │ false    │
│ firstname                │ false    │ true     │ false    │ false    │
│ lastname                 │ false    │ true     │ false    │ false    │
│ profilePictureFilenameS3 │ false    │ true     │ false    │ false    │
│ email                    │ false    │ false    │ false    │ false    │
│ phone                    │ false    │ false    │ false    │ false    │
│ birth                    │ false    │ false    │ false    │ false    │
│ bio                      │ false    │ true     │ false    │ false    │
│ picture                  │ false    │ true     │ false    │ false    │
│ description              │ false    │ true     │ false    │ false    │
│ history                  │ false    │ true     │ false    │ false    │
└──────────────────────────┴──────────┴──────────┴──────────┴──────────┘
```

Listing 9.9: Ausschnitt Access Control Matrix für Tabelle Person

9.8. DNS

Für die Applikation wurde die Domain www.bugchaser.ch erworben. Damit die Domain auch für die Plattform verwendet werden kann, müssen DNS-Einträge erstellt werden. Dafür wurde der Amazon eigene Dienst AWS Route 53 verwendet. Amazon Route 53 ist ein hochverfügbarer und skalierbarer DNS-Web-Service in der Cloud.

In AWS Amplify können anschliessend Weiterleitungen definiert werden, welche dann in CNAME DNS-Einträge in Route 53 resultieren. Bisher ist eine Weiterleitung von www.bugchaser.ch nach www.bugchaser.ch definiert. Es können jedoch noch weitere nach Bedarf hinzugefügt werden. Ebenfalls können Wildcard TLS Zertifikate direkt in der AWS Amplify Konfiguration ausgestellt werden.

9.9. Logging

Falls die Bug Chaser Plattform in Zukunft produktiv für das Testen von Software auf Schwachstellen verwendet wird, so werden darauf vertrauliche Informationen ausgetauscht, welche nicht an die Öffentlichkeit gelangen sollte, bevor die Schwachstellen behoben wurden.

Selbstverständlich ist auch die Bug Chaser Applikation nicht vor allen Schwachstellen sicher. Zum aktuellen Zeitpunkt nicht bekannte Schwachstellen können in Zukunft von einem Angreifer ausgenutzt werden. Dies lässt sich nicht immer vermeiden. Um auf einen solchen Sicherheitsvorfall bestmöglich vorbereitet zu sein, wurde Forensic Readiness durchgeführt.

Grundsätzlich wurden in drei verschiedenen Services in der Cloud Logging aktiviert. Der erste Dienst ist Cognito, welcher für die Authentisierung von Benutzern zuständig ist. Dafür wurde ein weiterer CloudTrail als weiterer Service benötigt. Als Zweites werden alle Anfragen an die GraphQL Schnittstelle aufgezeichnet. Diese Logs können im CloudWatch-Service ausgewertet werden. Als Letztes werden die Änderungen von Daten im S3 Bucket gespeichert. Zudem werden Daten zur Auslastung der verwendeten Services erhoben. Diese dienen jedoch nur dazu, die verwendeten Ressourcen und deren Entwicklung besser einzuschätzen zu können.

KAPITEL 10

Security

10.1. Übersicht

In diesem Kapitel wird beschrieben, welche Massnahmen ergriffen wurden, um die Sicherheit der Applikation zu gewährleisten. Anhand einem Threat Model wird die Bug Chaser Applikation analysiert und anschliessend nach STRIDE die möglichen Angriffsvektoren zu aufgezeigt.

10.2. Threat Model

Beim Threat modeling werden die Arten von Bedrohungsfaktoren ermittelt, die einer Anwendung Schaden zufügen könnte. Dabei wird die Perspektive von Angreifern eingenommen, um zu sehen, wie viel Schaden sie anrichten könnten. Es wird dazu eine gründliche Analyse der Softwarearchitektur durchgeführt.

In einem ersten Schritt wird anhand eines Diagramms aufgelistet, was der aktuelle Stand der Applikation ist und wie die einzelnen Komponenten miteinander kommunizieren. Das Threat Model wurde mit der open-source Applikation Threat Dragon entworfen. Dies ist für die Bug Chaser Plattform in der folgenden Abbildung Threat model ersichtlich.

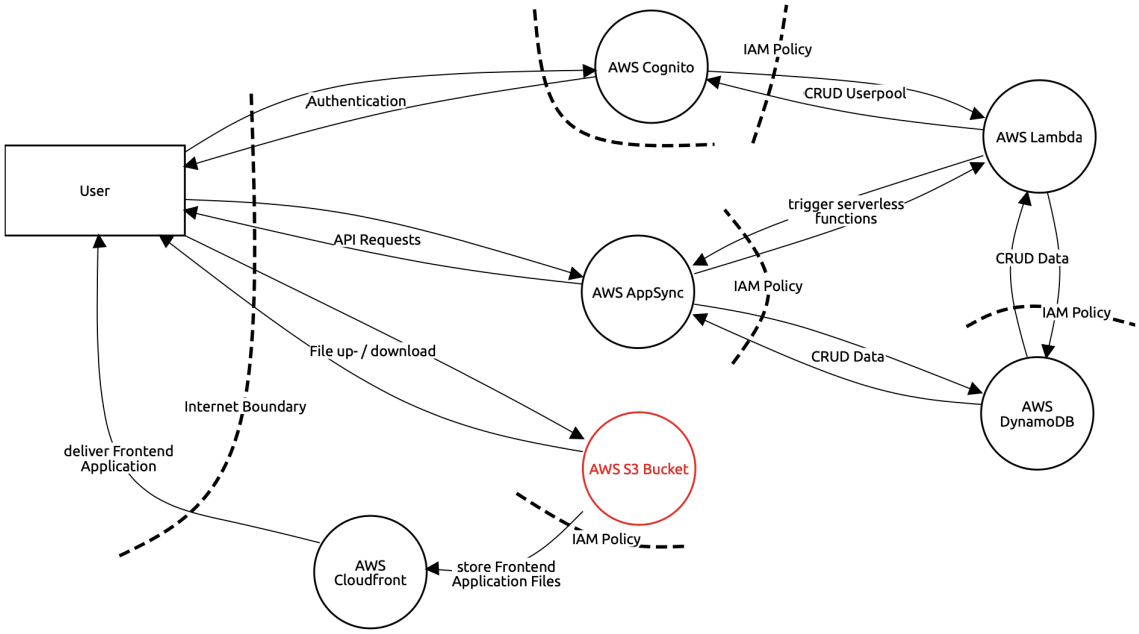


Abbildung 10.1.: Threat model

10.3. Bedrohungen

In einem weiteren Schritt werden die Bedrohungen für die Applikation identifiziert. Für die Bug Chaser Applikation wurde dies mit STRIDE umgesetzt. STRIDE ist wie folgt definiert:

Bedrohung	Eigenschaft	Definition
Spoofing	Authentifizierung	Sich als jemanden oder etwas anderes ausgeben
Tampering	Integrität	Verändern von Daten
Repudation	Nichtabstreitbarkeit	Abstreiten von Tätigkeiten
Information Disclosure	Vertraulichkeit	Verfügbarmachen von Daten, welche als vertraulich eingestuft wurden
Denial of Service	Verfügbarkeit	Einschränkung der Verfügbarkeit für andere Benutzer
Elevation of Privilege	Berechtigung	Erlangung von Fähigkeiten ohne entsprechende Berechtigung

Tabelle 10.1.: STRIDE Übersicht

Für die Bug Chaser Applikation konnten so über 50 verschiedene Bedrohungen identifiziert werden. Eine genaue Auswertung aller Bedrohungen ist im Anhang im Kapitel Threat Model zu finden. Die Bedrohungen der Data Flows wurden nur für das bessere Verständnis zum Threat Model hinzugefügt.

10.4. Offene Angriffsvektoren

Wie dem Threat Model zu entnehmen ist, konnte eine Schwachstelle nicht vollständig geschlossen werden. Dabei handelt es sich um die Vertraulichkeit von Daten, welche im Amazon S gespeichert werden.

Daten, welche im S3 Bucket abgespeichert werden, können durch die eine User Identity ID sowie dem dazugehörigen Key lesend zugegriffen werden. Gelingt es einem Angreifer, diese beiden Entität zu erlangen, so können die Dateien unberechtigt heruntergeladen werden. Eine genaue Beschreibung wieso dies so umgesetzt wurde ist im Abschnitt Limitationen AWS Amplify zu finden.

Aufgrund der terminierten Projektzeit konnte diese Schwachstelle nicht bis zum Projektende behoben werden. Für eine Folgearbeit wurde das Problem im Kapitel Erweiterung aufgenommen.

KAPITEL 11

Testing

11.1. Übersicht

Testing versichert, dass die Stabilität und Funktionalität den gewünschten Qualitätsrichtlinien entsprechen. Dazu werden diverse Tools eingesetzt. Diese Tools und Methoden werden in den folgenden Kapitel beschrieben.

11.2. Unit Testing

Unter Unit Testing wird das Testen von einzelnen Komponenten verstanden. Eine einzelne Komponente ist im Fall der Bug Chaser Applikation zum Beispiel eine Funktion. Diese wurden mit Jest implementiert.

Aufgrund der eingeschränkten Zeit des Projektes wurde der Fokus dabei nur auf die Business Logik des Backends gelegt. Die Funktionen des Backends werden von AWS Lambda ausgeführt. Ebenfalls werden in diesen Funktionen Schnittstellen zu anderen AWS Services verwendet. Beispiele dafür sind AWS Cognito oder AWS DynamoDB. Die Schnittstellen zu diesen Komponenten werden in den Unit-Tests ausgeschlossen und später in Systemtests geprüft. Dadurch fällt die abgedeckte Test-Quote eher tief aus. Durch System Testing wird jedoch auf einem höheren Level auch das gesamte Backend getestet.

11.3. System Testing

Systemtests befinden sich auf einer Teststufe, bei welcher das gesamte System gegen die gesetzten Anforderungen geprüft wird.

Im Fall der Bug Chaser Applikation werden die Systemtests direkt im Frontend mit Cypress durchgeführt. Cypress wird als All-in-one testing framework mit npm im Projekt installiert. Cypress erlaubt es, direkt in JavaScript Tests zu schreiben, welche anschliessend im Frontend ausgeführt werden. Diese sind mit der Interaktion zu vergleichen, welche ein Benutzer auf der Plattform ausführt.

11.3.1. Vorteile von Cypress

Durch Cypress wird Testing sehr vereinfacht. Wie in der Abbildung Cypress zu sehen ist, werden keine zusätzlichen Frameworks wie Mocha oder Jasmine benötigt. Cypress kombiniert die Funktionalität dieser Tools und bietet sie in einem benutzerfreundlichem Packet an.

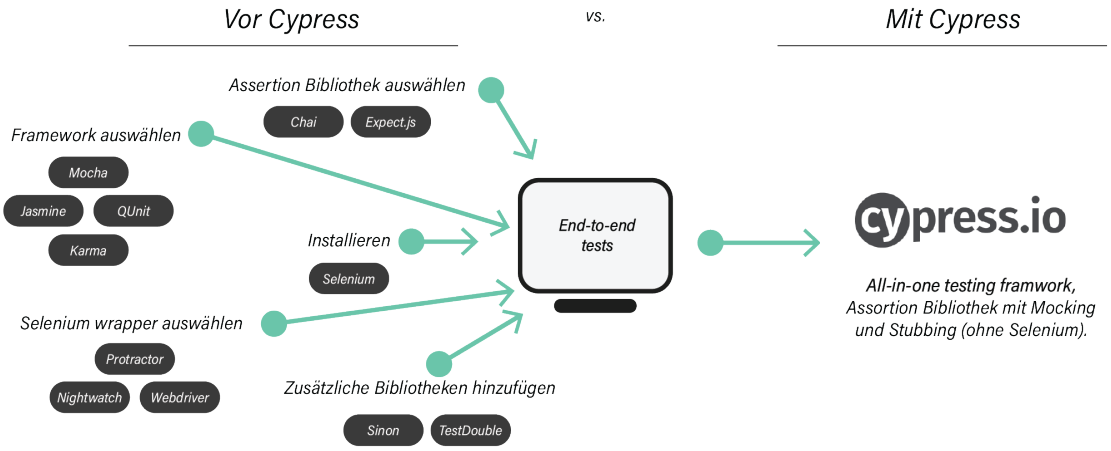


Abbildung 11.1.: Cypress

11.3.2. Nutzung Cypress

Es ist möglich Cypress in einer automatisierten CI CD laufen zu lassen. Der Fokus wurde jedoch auf halbautomatische Tests gesetzt. Das bedeutet, dass teilweise vom Benutzer bzw. Tester auch noch ein externer Input benötigt wird. Die Tests sollen ausschliesslich in der Development Umgebung durchgeführt werden, da die Tests auch auf die AWS Infrastruktur zugreifen und Daten verändern.

Um Cypress zu starten, muss der lokale Development Server laufen. Dieser wird mit `npm run dev` initialisiert und gestartet. Nachdem der Server läuft, kann cypress mit `npx cypress open` die Testumgebung gestartet werden.

Abbildung 11.2.: Cypress Start

Es öffnet sich nun ein Fenster, welches die bereitstehenden Tests darstellt.

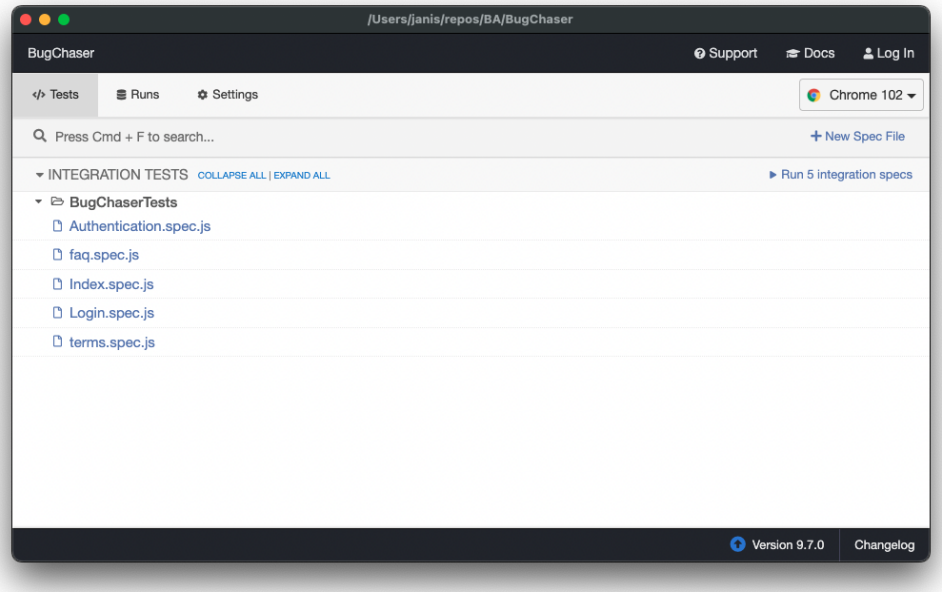


Abbildung 11.3.: Cypress Fenster

Um einen Test nun auszuführen kann dieser im Cypress Fenster angeklickt werden. Ein neues Fenster öffnet sich und der Test beginnt. Als Beispiel wird der Test `Authentication.spec.js` ausgeführt.

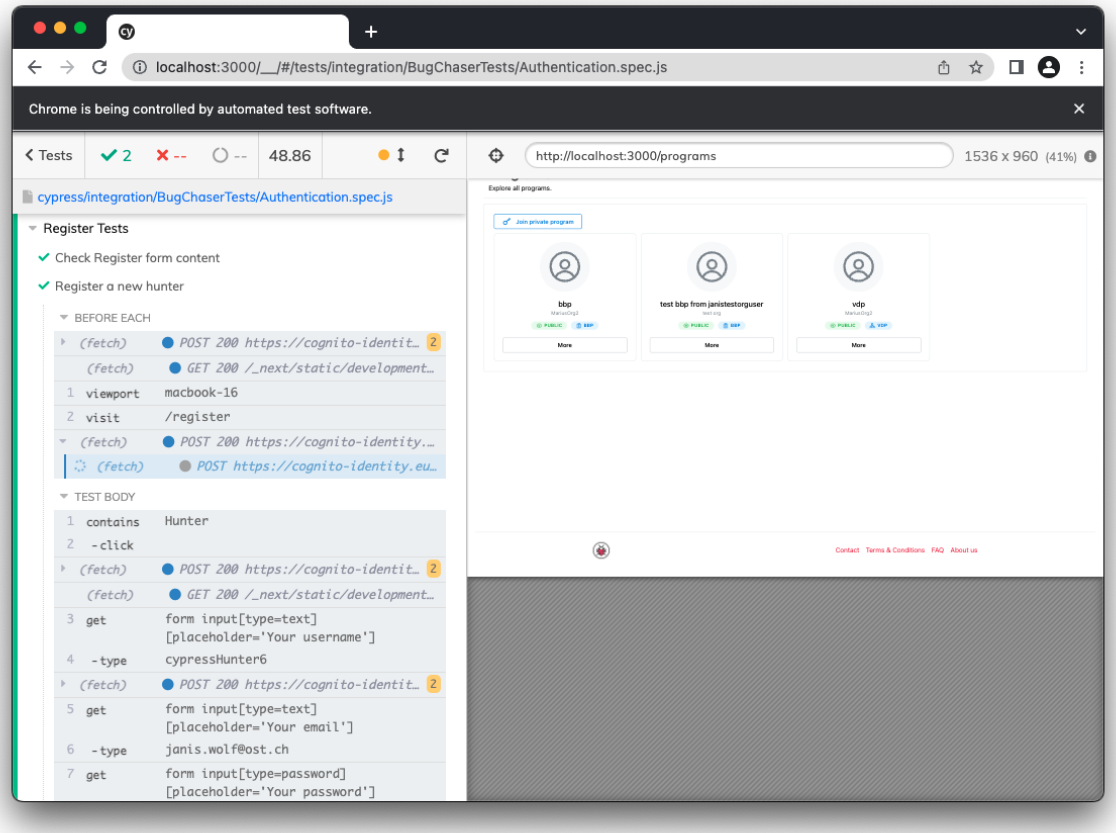


Abbildung 11.4.: Cypress Testausführung Beispiel

11.3.3. Existierende Tests

Die aktuell bereitstehenden Tests prüfen die grundlegende Funktionalität der Login und Registrierung, sowie auch die Index, FAQ und Terms Seiten. Die Tests können beliebig noch erweitert und verfeinert werden, um mehr Funktionalität zu testen. Zusätzlich wäre es möglich einen Teil der Tests vollautomatisch in der Pipeline laufen zu lassen. Mehr dazu wird im Kapitel Erweiterung beschrieben.

11.4. Static Code Analysis

Um den Sourcecode statisch zu analysieren und mögliche Fehlerquellen zu detektiert wurde Sonarqube verwendet.

Sonarqube ist eine Open Source Lösung, welche automatisch statische Code Reviews vornimmt und dabei Bugs und Code Smells detektiert. Durch die Pipeline in Github wird diese Analyse bei jedem Push ins Repository durchgeführt.

In der folgenden Abbildung werden ein paar Statistiken aus Sonarqube dargestellt. Falls Bugs oder Schwachstellen im Code detektiert werden, werden diese hier angezeigt. Die 38 detektierten Code Smells beziehen sich hauptsächlich auf noch nicht erledigte ToDo's im Code.

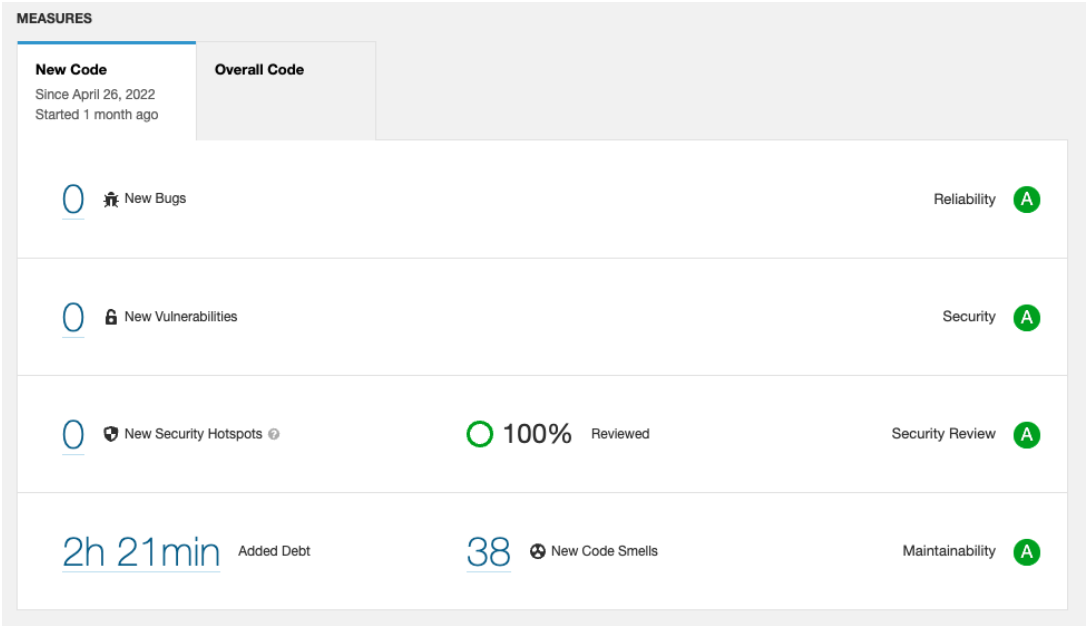


Abbildung 11.5.: Sonarqube

11.5. NFR Testing

Nichtfunktionale Anforderungen werden manuell durch die Entwickler getestet. Ein passendes Framework, welches diese Aufgabe übernimmt, wurde nicht eingesetzt.

11.6. Usability Testing

Usability Tests werden manuell durchgeführt. Diese zeigen, ob die Applikation verständlich und intuitiv für den Benutzer aufgebaut ist.

An verschiedenen Zeitpunkten im Projekt wurden Usability Tests durchgeführt, meistens von den Entwicklern gegenseitig. Während der Validierung wurde zudem Feedback von externen Personen gesammelt. Das ausgewertete Feedback deutet stark darauf hin, dass die Applikation klar zu bedienen ist. Teilweise sind Hilfestellungen aber noch erwünscht.

11.7. Zusammenfassung

Testing der Applikation ist ein wichtiger Teil, um die Qualität und Funktionalität der Applikation zu versichern. Durch Cypress und Unit Tests wird momentan ein Teil der Applikation automatisiert getestet, allerdings könnten noch viel mehr Testfälle abgedeckt werden. Die noch nicht automatisierten Tests werden zum aktuellen Zeitpunkt noch durch einen Entwickler durchgeführt.

KAPITEL 12

Continuous Integration & Continuous Delivery

12.1. Übersicht

In diesem Kapitel wird das automatisierte Deployment der Applikation beschrieben.

12.2. Continuous Integration

Sobald neue Commits auf GitHub eingecheckt werden, wird ein GitHub Workflow (Action) ausgeführt. Diese verbindet sich mit SonarQube und macht dann eine Statische Code Analyse. Die Resultate können dann nach erfolgreichem Abschliessen der Pipeline im SonarQube Web GUI analysiert werden.

```yaml
# BugChaser/.github/workflows/build.yml
name: Build
on:
  push:
    branches:
      - main
      - dev
jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
        with:
          fetch-depth: 0
      - uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
```

Listing 12.1: GitHub Action

12.3. Continuous Delivery

In AWS Amplify sind die beiden Branches **main** und **dev** mithilfe einem von AWS erzeugtem WebHook eingebunden. Der Webhook halt folgende Konfiguration und wird daher durch jeden Push in den jeweiligen Branch ausgelöst.

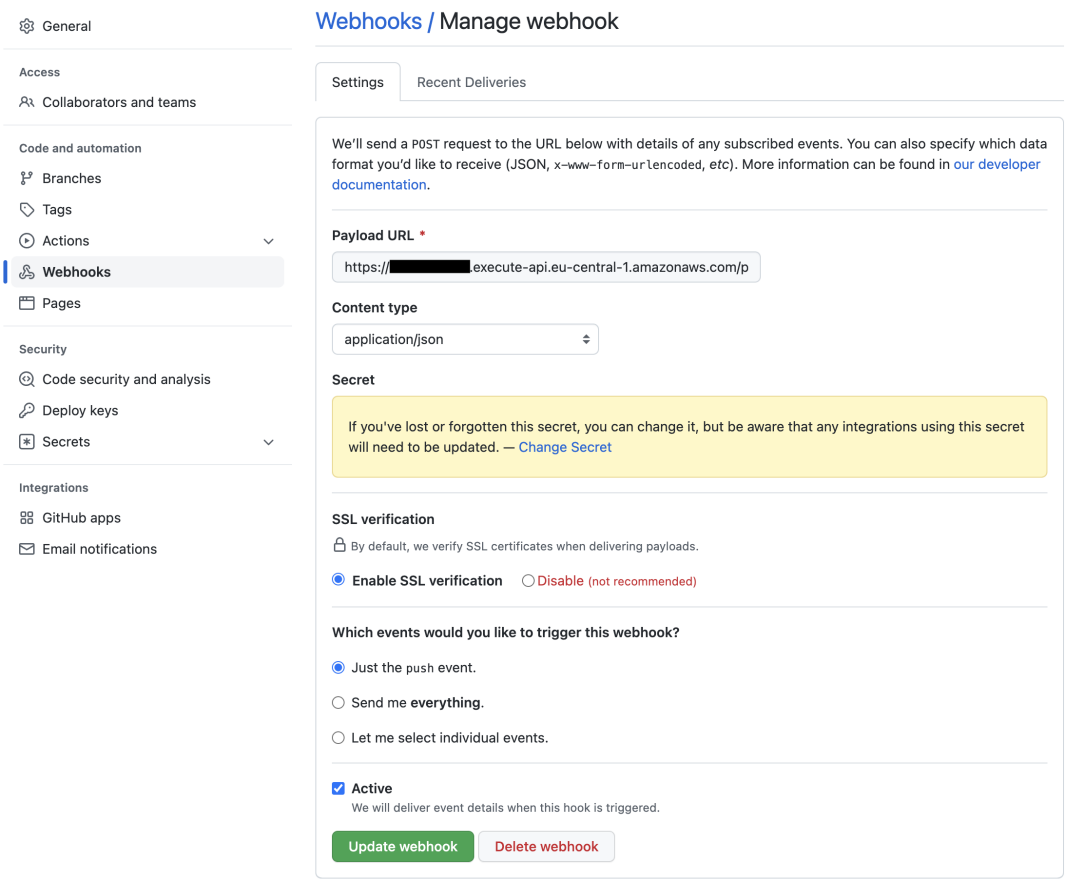


Abbildung 12.1.: Webhook Konfiguration

Sobald AWS Amplify durch den Webhook ausgelöst wird, wird das Frontend auf das passende Environment deployed.

12.3.1. Amplify Limitierung

Da AWS Amplify zur Einbindung der Repositories nur GitHub unterstützt, mussten wir den Code für die Applikation von Gitlab auf GitHub verschieben. Aus diesem Grund ist die Dokumentation der Arbeit auf einem anderen Git Provider als die Applikation selbst. Das Einbinden der Self Hosted Gitlab Instanz der FH OST hätte eine Konfiguration in der der Datenbank benötigt, welche wir nicht vornehmen konnten.

12.4. Environments

AWS Amplify bietet die Möglichkeit, mehrere verschiedene Environments zu haben. Wir haben uns dafür entscheiden eine **Main** und eine **Dev** Umgebung zu betrieben. Diese sind verknüpft mit den jeweiligen Git Branches.

Die produktive Umgebung ist unter der offiziellen URL www.bugchaser.ch zu erreichen. Die Umgebung für Testing wird auf dev.bugchaser.ch betrieben.

Diese beiden Umgebungen sind über AWS Amplify komplett getrennt. Das bedeutet einzelne Datenbanken, API Gateways und Storage Buckets.

KAPITEL 13

Kosten

13.1. Übersicht

In diesem Kapitel werden die entstanden Kosten in AWS analysiert und dokumentiert. Dazu wird das Tool **AWS Cost Management** von AWS verwendet. Es erlaubt mit Graphen und Visualisierungen die Kosten der verschiedenen Services darzustellen.

Durch die Cloud Umgebung sind die Preise sehr variabel, da man nur für die Dienste und Leistungen bezahlt, welche man verwendet.

13.2. Beanspruchte Services von AWS

Natürlich benötigen wir nur einen kleinen Teil der Services, welche AWS anbietet. Grundlegend werden folgende Kategorien von Diensten von uns benutzt.

- Registrar (Domain Registration)
- AWS Amplify (Frontend and Backend Deployment with Cognito)
- Tax (Steuern)
- Route 53 (DNS)
- S3 Bucket
- Others

13.3. Fixkosten

Auch in AWS fallen gewisse Fixkosten an. Diese werden durch die Domain (bugchaser.ch) und die DNS-Zone generiert.

13.3.1. Domain (Registrar)

Diese Kosten fallen einmal jährlich an. Pro Domain ***.ch** verrechnet AWS **13 USD + 1 USD (Tax)**, also total **14 USD**. Wird eine ***.com** Domain gekauft variieren diese Kosten natürlich.

Registrar		**$13.00**
Global		**$13.00**
Amazon Registrar DomainRegistration		$13.00
Registration of bugchaser.ch	1.000 Quantity	$13.00

Abbildung 13.1.: Registrar Kosten März 2022

13.3.2. Domain (Route 53)

Die folgenden Kosten fallen monatlich an.

Route 53		**$0.51**
Global		**$0.51**
Amazon Route 53 DNS-Queries		$0.01
$0.40 per 1,000,000 queries for the first 1 Billion queries	16,124.000 Queries	$0.01
Amazon Route 53 HostedZone		$0.50
$0.50 per Hosted Zone for the first 25 Hosted Zones	1.000 HostedZone	$0.50
Amazon Route 53 Intra-AWS-DNS-Queries		$0.00
Queries to Alias records are free of charge	1,525.000 Queries	$0.00

Abbildung 13.2.: Route 53 Kosten April 2022

13.4. Bisherige Kosten

Die bisherigen Kosten (von März 2022 bis und mit May 2022) belaufen sich auf 22.39 USD. Darin inkludiert sind alle Kosten (Domain, Hosting, Deployment etc.).

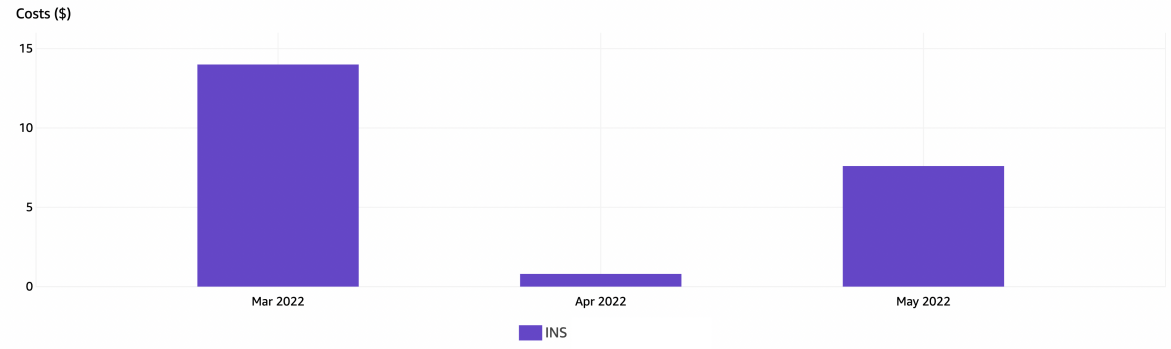


Abbildung 13.3.: Totale Kosten des INS Accounts

Wie in der Grafik zu erkennen sind die Werte der drei Monate sehr unterschiedlich. Der erste Monat war aufgrund der Domain Registrierung am teuersten, da diese 14 USD gekostet hat.

13.4.1. Kosten aufgeteilt nach Service

Um einen besseren Überblick zu schaffen, werden die Services in der folgenden Grafik nach Service aufgeteilt. Die Kosten für den Registrar sind für die Domäne. Da diese jährlich anfallen sind diese nur im ersten Monat enthalten. Im April kommen dann weitere Kosten hinzu, da das Setup des Projekts

durchgeführt wurde. Die DNS Kosten (Route 53) sind dort am spürbarsten.

Im Mai wird dann deutlich, dass das Deployment mit AWS Amplify sich auch auf die Kosten auswirkt. Mit vielen Tests und Deployments während der Entwicklung und Testing steigen entsprechend diese Kosten auch an. Allerdings sind diese Kosten in der Produktion sicherlich tiefer, da die Deployments nicht oft vorkommen.

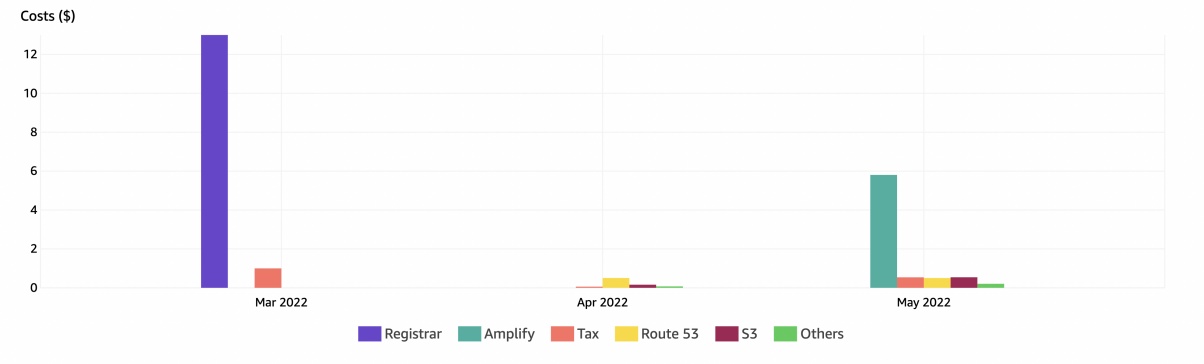


Abbildung 13.4.: Kosten nach Service aufgeteilt

13.5. Prognose

Im AWS Cost Management lassen sich auch Prognosen erstellen. Die folgende Grafik ist jedoch nicht sehr aussagekräftig, da unsere bisherigen Kosten nur über 3 Monate verteilt sind. Zusätzlich sind die Kosten für die Domain nur einmal pro Jahr einzurechnen. Würde die Applikation produktiv betrieben werden, halten sich die AWS Amplify Kosten auch tiefer, da nicht so oft eine neue Version der Applikation deployed wird.

Trotzdem ist es spannend zu sehen, wie AWS die zukünftigen Kosten einschätzt. Mit mehr Daten, das heisst längere Nutzung und Betrieb der Applikation, werden sich die Daten vermutlich noch genauer.

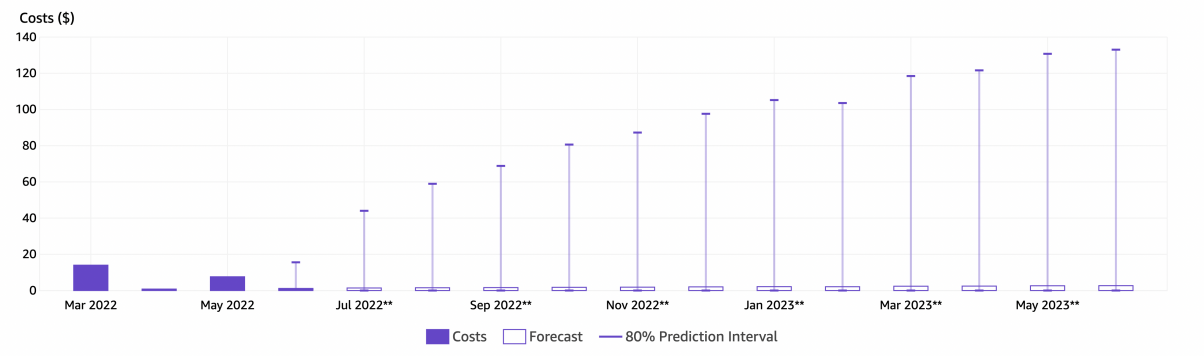


Abbildung 13.5.: Kosten Prognose für 12 Monate in die Zukunft

13.6. Zusammenfassung

Die verursachten Kosten in AWS halte sich sehr gering. Wenn die Applikation nicht aktiv verwendet wird, generiert sie kaum Kosten. Durch die Verwendung von AWS Amplify in Kombination mit dem Serverless Backend, generieren Lambda Funktionen nur bei Ausführung Kosten. Wird die Applikation nicht benutzt entstehen dadurch also keine weiteren Kosten.

Die Entscheidung, die Applikation in der Cloud zu betreiben macht darum durchaus Sinn. Müsste eine ähnliche Applikation On-Premise zur Verfügung gestellt werden, wären die Kosten nur bereits durch den Stromverbrauch höher.

KAPITEL 14

Erweiterung

14.1. Übersicht

Während der Entwicklung der Applikation und der anschlissenden Validierung tauchten noch zusätzliche Anforderungen an die Plattform auf, welche bisher nicht in den Anforderungsspezifikationen erfasst wurden.

Die folgenden Erweiterungen sind Möglichkeiten und Ideen, die noch implementiert werden könnten.

14.2. Funktionen

14.2.1. Text Editor Optimierung

Der verwendete Texteditor TipTap ist durch die API und Module sehr ausbaufähig. Die folgenden Ideen würden die User Experience und Funktionen aber noch weiter verbessern.

Styling

Damit der Benutzer den Editor leichter bedienen kann, könnten Schaltflächen angebracht werden, welche im Hintergrund verschiedene Funktionen aufrufen und mit dem Editor kommunizieren. So kann ein Text mit Knopfdruck z.B. Bold gemacht werden.
Die Funktionalität **Undo** und **Redo** zu implementieren könnte auch sehr praktisch sein.

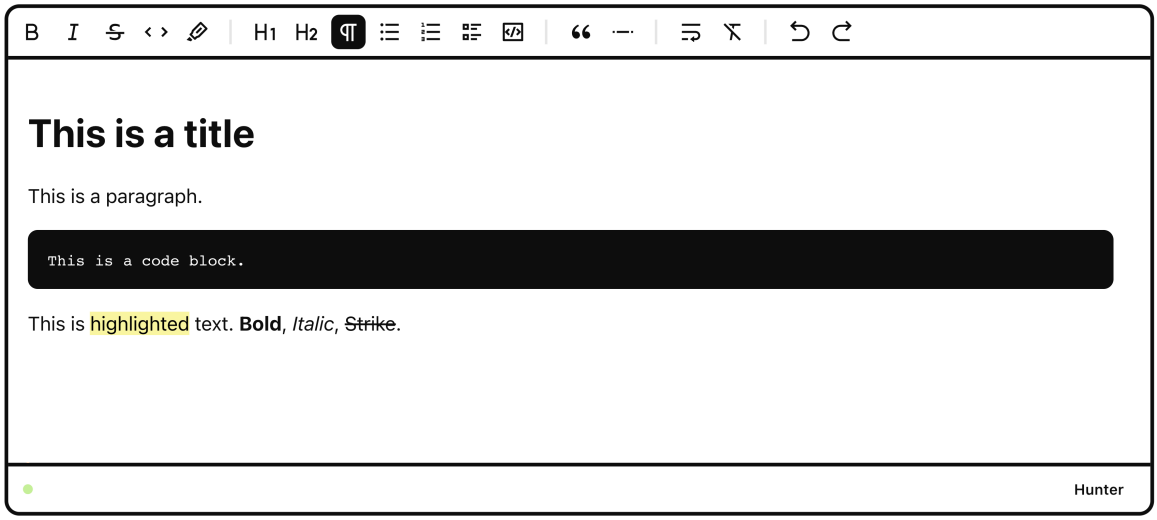


Abbildung 14.1.: TipTap Editor Erweiterung

Wie im Screenshot oben gezeigt, kann auch das Design des Editors noch angepasst werden. Der Codeblock beispielsweise wird so mehr hervorgehoben.

Ein Beispiel dazu findet sich hier.

Dateiupload im Text Editor oder Report

Damit Benutzer im Report Screenshots und eventuell Videos anhängen kann, wird im Text Editor noch einiges benötigt. Im Hintergrund muss die Datei in den S3 Bucket hochgeladen und dem Benutzer nach fertigem Upload im Editor angezeigt werden.
Ein Beispiel dazu findet sich hier.

14.2.2. Caching von Bildern

Aktuell werden die Bilder, welche vom S3 Bucket geladen werden, nicht im Browser Cache abgelegt. Es wird lediglich die URL, welche vom S3 Service geliefert wird, im LocalStorage abgelegt. Allerdings wäre es möglich, die Bilder selbst im Browser Cache abzulegen. So würden man den Datenverkehr zwischen dem Client und S3 Bucket minimieren und zugleich die User Experience durch schnellere Ladezeiten verbessern.

In der Datenbank wird bereits mit den Bildern abgelegt, wann diese hochgeladen werden. Das heisst wenn sich ein Bild aktualisiert, kann der Webclient mit einer Query auf den Datensatz, welche ohnehin gemacht werden muss, feststellen, ob sich ein Bild aktualisiert hat oder nicht.

Falls keine Aktualisierung stattgefunden hat, kann auf das Bild im Cache zugegriffen werden. Falls das Bild allerdings geändert wurde, muss dieses vom S3 Bucket geholt und das alte Bild im Browser Cache ersetzt werden.

14.2.3. Sortierung, Filter und Suche

An diversen Stellen der Applikation finden sich Listen oder Gruppen von Objekten in Listen oder Gitter Anordnungen. In diesen Komponenten wäre es für den Benutzer von Vorteil, wenn nach spe-

zifischen Werten sortiert, gefiltert oder gesucht werden kann.

Diese Funktionen müssten auf dem Frontend und nach Bedarf auf dem Backend implementiert werden.

14.2.4. Mobile Layout

Die Applikation ist aktuell nur teilweise mobile freundlich. Viele Komponenten von Mantine UI passen sich bereits an Mobile Layouts an. Spezifisch aber z.B. die Tabs Komponente passt sich nicht korrekt an. Mantine bietet aber Hilfestellungen an mit welchem die Komponenten angepasst werden können. Zusätzlich ist es gut möglich, dass die Komponenten aktualisiert werden, und somit dann Mobile Layouts unterstützen.

14.2.5. Dateiupload

Der Dateiupload kann an verschiedenen Orten noch verbessert werden. Da die Authentifizierung noch nicht optimal ist, kann es vorkommen, dass Dateien noch im S3 Bucket abgelegt sind, diese von der Applikation nicht mehr benötigt bzw. referenziert sind. Diese Objekte brauchen dann Speicherplatz, verursachen dadurch Kosten und werden aber gar nicht gebraucht.

Dieses Problem könnte man mit einer Lambda Funktion lösen, welche z.B. einmal pro Woche diese Dateien aufräumt. Dies ist natürlich nicht die beste Lösung, da sich dieses Problem mit der Behebung der Authentisierung auch lösen würde.

Sicherheit

Da die Programme sensitive Dateien wie Source Code und Executables enthalten können, muss versichert werden, dass die Daten nur von berechtigten gelesen werden können. Momentan ist dies nur teilweise implementiert, da AWS Amplify keine genaue Kontrolle auf File Level mit Gruppen unterstützt.
Eine mögliche Lösung, die in der Theorie funktionieren sollte, wäre die auf diese Dateien via Lambda zuzugreifen. Die Lambda Funktion könnte dann die benötigten Rechte prüfen.

Bildvorschau bei Dateiupload

Sofern ein hochgeladenes Bild vom Typ Bild ist, könnte dem Benutzer eine kleine Vorschau des Hochgeladenen Bildes angezeigt werden. Eine mögliche Implementierung dafür wäre React Dropzone.

14.2.6. Private Routes

Momentan wird dem Benutzer eine Fehlermeldung angezeigt, wenn er eine Route besucht, für welche er keinen Zugriff hat. Die Fehlermeldung zeigt aber einen Datenbank Fehler an, da die Daten aufgrund der fehlenden Berechtigungen nicht geladen werden können. Besser wäre es, wenn bei fehlenden Berechtigungen eine universale Fehlermeldung erscheint oder man direkt auf die Startseite zurückverwiesen wird.

Eine mögliche Lösung, wie das erreicht werden kann findet man hier.

14.2.7. Amplify GraphQL Queries Limit

AWS Amplify liefert bei einer Query an GraphQL standardmässig maximal 100 Elemente zurück. Würden aber eigentlich mehr Elemente existieren, werden diese aktuell nicht dargestellt. Dazu müsste

Pagination implementiert werden. Der next Token von AWS Amplify kann dafür verwendet werden. Die Dokumentation findet sich hier.

14.2.8. Queries optimieren

Aktuell liefern die meisten Queries noch mehr Daten, als der Nutzer eigentlich braucht. Mit Redux Toolkit und custom Queries könnten diese Abfragen jedoch noch optimiert werden.

14.2.9. Öffentliches Profil

Der Hunter hat zwar aktuell ein öffentliches Profil, welches von anderen Huntern eingesehen werden kann. Allerdings sind noch nicht viele Informationen dort vorhanden. Es könnten noch Social Media Links, eine Beschreibung des Profils und zusätzliche Informationen angezeigt werden.

14.2.10. Leaderboard

Das Leaderboard, eine Rangliste der besten Hunter, ist im Backend bereits implementiert, aber aus Zeitgründen wurde dies noch nicht im Frontend eingebaut.

14.2.11. Benachrichtigungen

Der Nutzer könnte per E-Mail Benachrichtigungen auf diverse Aktivitäten erhalten. Dies könnte zum Beispiel eine Nachricht auf einen neuen Report sein oder ein Newsletter mit neu beigetretenen Programmen.

14.2.12. Social Provider Login

Damit sich Benutzer nicht mit Username und Passwort anmelden müssten, könnte zusätzlich ein Social Provider Login angeboten werden. Das bedeutet, dass die Nutzer über Facebook, Google oder GitHub anmelden könnten. Für diese Funktionalität müsste ein Teil der Architektur und Frontend angepasst werden.

Von Cognito wird diese Art von Authentifizierung unterstützt.

Teil IV.

Projektdokumentation

KAPITEL 15

Projektplan

15.1. Projekt Übersicht

15.1.1. Ziel und Zweck

Ziel dieser Arbeit ist es, ein Produkt zu erarbeiten, welches Studierende an der Ostschweizer Fachhochschule am Campus Rapperswil-Jona (ehemals Hochschule für Technik Rapperswil) beim Kennenlernen des Vulnerability Management im Software-Entwicklungsprozess unterstützt. Der Unterricht im Modul Secure Software soll dafür um eine Sequenz im Bug Bounty Bereich erweitert werden.

15.1.2. Gültigkeitsbereich

Die Gültigkeit dieses Dokumentes beschränkt sich auf die Laufzeit dieser Bachelorarbeit. Der Projektplan wird während der Arbeit laufend aktualisiert.

15.1.3. Lieferumfang

Zum Abschluss der Arbeit werden folgende Dokumente geliefert:

- Dokumentation für E-Prints Publikation, bestehend aus:
  - Abstract
  - Management Summary
  - Technischer Bericht
  - Vorstudie
  - Software Dokumentation
  - Projektdokumentation
  - Anhang
- Repository mit Source Code
- A0-Plakat
- Dokumentation für Interne, bestehend aus:
  - Abstract
  - Management Summary
  - Technischer Bericht
  - Vorstudie
  - Software Dokumentation
  - Projektdokumentation
  - Anhang
  - Aufgabenstellung
  - Eigenständigkeitserklärung
  - Urheber- und Nutzungsrechte
  - Einverständniserklärung
  - Persönliche Berichte

15.2. Projekt Organisation

15.2.1. Projektteam

Janis Wolf

Autor
janis.wolf@ost.ch

Marius Zindel

Autor
marius.zindel@ost.ch

15.2.2. Externe Schnittstellen

Prof. Dr. Nathalie Weiler

Betreuerin
nathalie.weiler@ost.ch

Christian Birchler

Experte
christian.birchler@cnlab.ch

Prof. Dr. Mitra Purandare

Gegenleserin
mitra.purandare@ost.ch

15.2.3. Zuständigkeiten

Das Projektteam besteht aus zwei Personen: Janis Wolf und Marius Zindel. Die Arbeit wird als Team zu zweit geschrieben. Für beide gilt diese Arbeit als Bachelorarbeit an der Ostschweizer Fachhochschule. Dadurch entsteht eine flache Hierarchie. Alle Personen sind gleichermassen am Projekt beteiligt. Wesentliche Entscheide können im Team gemeinsam getroffen werden. Um bei Unstimmigkeiten die Meinungsunterschiede aufzulösen, wurde die Verantwortung aufgeteilt:

Janis Wolf

- Scrum Product Owner
- Scrum Developer
- Verantwortlich für die Qualität des Produktes

Marius Zindel

- Scrum Master
- Scrum Developer
- Verantwortlich für die Qualität der Dokumentation sowie der Zwischen- und Schlusspräsentation

15.2.4. Auftraggeber

Diese Arbeit wird ohne externe Partnerunternehmen durchgeführt. Auftraggeber dieser Arbeit ist die Betreuerin Prof. Dr. Nathalie Weiler, welche als Secure-Software-Modulverantwortliche auch die Bedürfnisse und Anforderungen der Fachhochschule vertritt.

15.2.5. Kostenvoranschlag

Das Projekt startet in der 1. Semesterwoche am 21. Februar 2022 und endet in der 17. Semesterwoche am 17. Juni 2022 mit der Abgabe aller Dokumente. Für eine Bachelorarbeit wird ein Zeitaufwand von 360 Stunden für jeden Studierenden erwartet. Das Autorenteam besteht aus zwei Studenten, das einen totalen Zeitaufwand von 720 Stunden für das ganze Projekt ergibt. Während den Frühlingsferien in der Kalenderwoche 15 werden keine Arbeiten getätigt.

	ECTS	Stundenaufwand pro Woche	Stundenaufwand gesamt
Janis Wolf	12	22.5	360
Marius Zindel	12	22.5	360
Total	24	45.0	720

Tabelle 15.1.: Stundenaufwand

Pro Woche und pro Person müssen durchschnittlich 22.5 Arbeitsstunden geleistet werden.

15.3. Projekt Management

An der OST werden in den Modulen Software Engineering 1 sowie Software Engineering 2 unterschiedliche Methoden kennengelernt, wie Projekte im Softwareentwicklungsprozess oder generell in der Informatik organisiert werden können. Zwei Beispiele dafür sind der Rational Unified Process und Scrum, welche beide ihre Vor- sowie Nachteile in gewissen Aspekten haben [9, 10].

15.3.1. RUP

RUP ist ein sehr umfassender Prozess und eignet sich gut für Langzeitplanung. Dabei werden vier verschiedene Phasen unterschieden:

- Inception
  - Ungefähre Vision
  - Festlegen des Umfangs
  - Grobe Schätzungen für den Aufwand
- Elaboration
  - Identifizierung der meisten Anforderungen
  - Iterative Umsetzung des Kernkonzeptes
  - Beseitigung von hohen Risiken
  - Realistischere Schätzungen für den Aufwand
- Construction
  - Iterative Umsetzung der Funktionalität
  - Behebung geringerer Risiken
  - Vorbereitung auf das Deployment
- Transition
  - Tests und Validierung
  - Finales Deployment

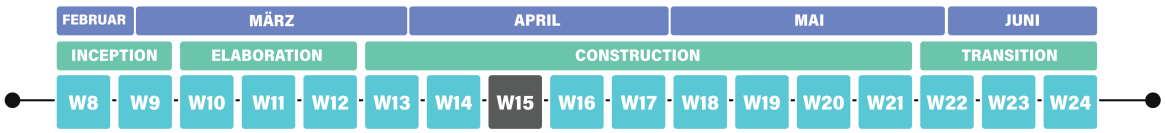


Abbildung 15.1.: Projekt Management - RUP

15.3.2. Scrum

Scrum hingegen ist ein leichtgewichtiges Framework, das Teams hilft, Lösungen für komplexe Probleme zu erreichen. Dies ermöglicht eine genauere Planung auf kurze Zeit. In Scrum wird in Iterationen gearbeitet, welche Sprint genannt werden. Hauptsächlich besteht Scrum aus 3 Rollen, 5 Events, 3 Artefakten sowie 5 Werte [11].

- Scrum Team
  - Developers
    Entwickler, welche das Projekt direkt voranbringen
  - Product Owner
    Person, welche für Eigenschaften und Erfolg des Produkts verantwortlich ist
  - Scrum Master
    Person, welche für das Einhalten von Scrum zuständig ist
- Scrum Events
  - The Sprint
    Eine Iteration / Arbeitsabschnitt
  - Sprint Planning
    Planung des nächsten Sprints
  - Daily Scrum
    Kurze Besprechung, was an diesem Tag umgesetzt wird
  - Sprint Review
    Ende des Sprints, Product Backlog anpassen
  - Sprint Retrospective
    Ende eines Sprints, Zukunft effizienter und effektiver gestalten
- Scrum Artifacts
  - Product Backlog
    Geordnete Auflistung der Anforderungen an das Produkt
  - Sprint Backlog
    Geordnete Auflistung der Anforderungen für den aktuellen Sprint
  - Increment
    Summe aller fertig gestellten Anforderungen
- Scrum Values
  - Commitment
  - Focus
  - Openness
  - Respect
  - Courage
    Werte, die jedes Scrum Mitglied einhalten sollte, um eine bestmögliche Zusammenarbeit zu erreichen.

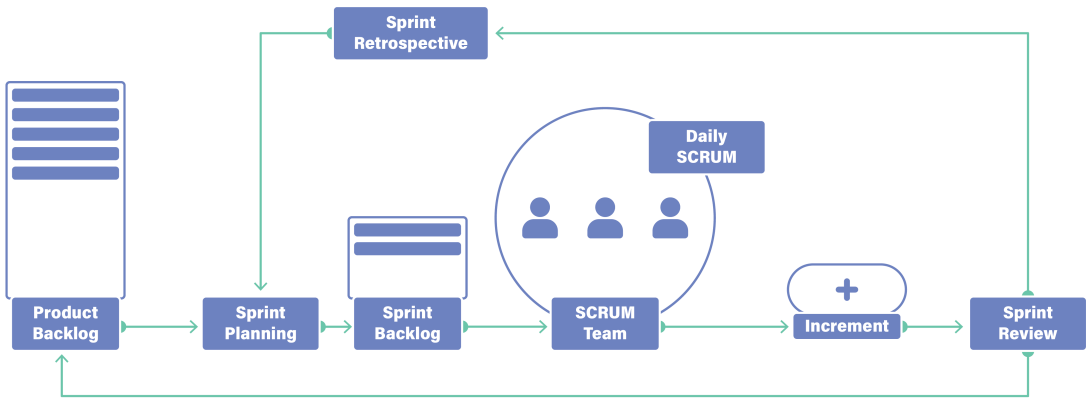


Abbildung 15.2.: Projekt Management - SCRUM Diagram

15.3.3. Scrum+

Eine weitere Methode, wie Projekte organisiert werden können ist Scrum+. Dies ist eine Kombination aus den besten Teilen von RUP und Scrum. Scrum+ ist nicht offiziell vom Scrum Framework vorgegeben, sondern eine Erweiterung, welche an der FH OST unterrichtet wird. Einsatzort der RUP und Scrum Modelle ist hauptsächlich die Softwareentwicklung, jedoch können beide (wie auch die Kombination Scrum+) auf fast jedes andere Projekt adaptiert werden. Ziel ist es so, das Projekt mit Scrum+ in vier Phasen aufzuteilen, welche dann mit mehreren agilen Sprints durchgeführt werden. Dieses Projekt wird daher nach Scrum+ organisiert.

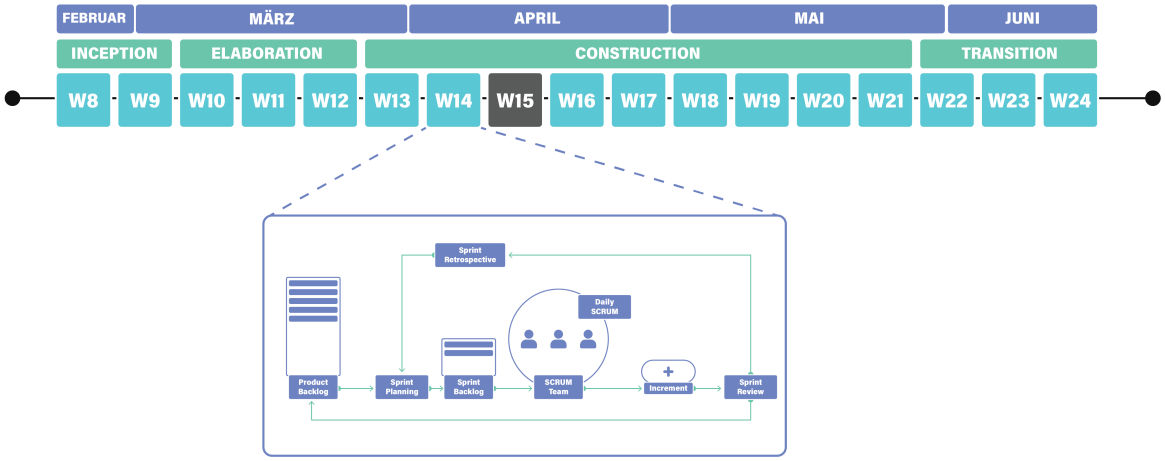


Abbildung 15.3.: Projekt Management - SCRUM+ Diagram

15.4. Zeitplan

Es werden die verwendeten Phasen und der Zeitplan des Projekts beschrieben. Der Zeitplan gibt einen klaren Überblick, welcher Stand aktuell im Projekt verfolgt wird.

15.4.1. Phasen und deren Iterationen

Es wurde sich für die Phasen *Inception*, *Elaboration*, *Construction* und *Transition* entschieden. Diese bieten eine gute Einteilung und lassen sich auf viele Projekte adaptieren.
Die 4 Phasen werden dann in total 17 Sprints eingeteilt. Ein Sprint dauert grundsätzlich eine Woche. Durch Ferientage und andere Abwesenheiten sind die Sprints teilweise kürzer als eine Woche. Sprint 8 wird zwar definiert, jedoch werden keine Arbeiten erledigt.
Kurze Sprints erlauben eine agile Arbeitsweise. Sie bieten Spielraum für Anpassungen und ermöglichen dadurch eine gute Flexibilität im Projekt.

Inception

Dauer: 2 Wochen
Zeitraum: 21.02.2022 - 06.03.2022
In diesen ersten zwei Wochen des Projekts werden grundlegende administrative Arbeiten erledigt. Es wird dabei auch der Rahmen der Arbeit festgelegt und sich eine Übersicht verschaffen.

Elaboration

Dauer: 3 Wochen
Zeitraum: 7.03.2022 - 27.03.2022
Nachdem die Inception Phase abgeschlossen wurde, werden in der Elaboration Phase nun die Anforderungen präziser definiert. Es wird dabei auch die nötige Research betrieben, welche zwingend notwendig ist, damit die Anforderungen auch machbar aufgestellt werden. Das erlernte Wissen sowie die getroffenen Entscheide werden ebenfalls in dieser Phase in die Dokumentation aufgenommen.

Construction

Dauer: 8 Wochen
Zeitraum: 28.03.2022 - 30.05.2022
In der Construction Phase wird nun die definieren Anforderungen umgesetzt. Diese Phase ist ein grosser Teil der Arbeit und beträgt darum auch acht Wochen. Während dieser Phase wird der Zwischenstand des Produkts an den Meetings besprochen. Allfällige Änderungen der Anforderungen können besprochen werden und fliessen bei Bedarf dann in den nächsten Sprint ein.
Die Kalenderwoche 15 ist als Ferienwoche eingetragen und wird daher nicht zu der Dauer dieser Phase gerechnet.

Transition

Dauer: 3 Wochen
Zeitraum: 31.05.2022 - 17.06.2022
In dieser letzten Phase des Projekts werden offene Arbeiten nun abgeschlossen. Das Produkt sollte zu diesem Zeitpunkt fertiggestellt werden. Allfällige wichtige Tests und Fehlerbehebungen dürfen noch durchgeführt werden. Es werden keine neuen Funktionen mehr implementiert.
Die Dokumentation und andere Dokumente der Arbeit werden auch fertiggestellt.

15.4.2. Meilensteine

Die definierten Meilensteine sollen das Autorenteam unterstützen, den Projektfortschritt messbar zu machen. Dafür wurden fünf Meilensteine definiert, welche über das Projekt verteilt immer wieder einen Teil des Projektes abschliesst.

M1 **End of Inception**	**Datum:** 06.03.2022 **Phase:** Inception Der Projektplan, die Risikoanalyse sowie das Qualitätsmanagement wurden definiert und abgeschlossen. Die Infrastruktur wurde eingerichtet. Dazu zählt alles, was für eine erfolgreiche Durchführung des Projektes benötigt wird.
M2 **End of Elaboration**	**Datum:** 27.03.2022 **Phase:** Elaboration Die Anforderungen wurden genauer identifiziert und mit der Betreuerin besprochen. Recherchen über die Materie wurden durchgeführt sowie dokumentiert. Eine Technologie Evaluierung wurde durchgeführt und dokumentiert.
M3 **Zwischen-präsentation**	**Datum:** 26.04.2022 **Phase:** Construction Ein erster Prototyp wurde umgesetzt und ist bereit für ein Review. Der Stand der Arbeit wurde in einer Zwischenpräsentation den Betreuenden präsentiert. Rückmeldungen wurden aufgenommen und dokumentiert.
M4 **End of Construction**	**Datum:** 29.05.2022 **Phase:** Construction Die Implementation des definierten Produkts wurde gemäss den definierten Anforderungen abgeschlossen. Das Produkt ist bereit für Auslieferung.
M5 **Abgabe**	**Datum:** 17.06.2022 **Phase:** Transition Die Dokumentation und der dazugehörige Source Code wurde auf dem AVT abgegeben. Zusätzlich wurde eine digitale Kopie aller Dokumente gemäss Lieferumfang der Betreuerin per E-Mail zugestellt. Die Abschlusspräsentation wird zu einem späteren Zeitpunkt durchgeführt.

15.4.3. Gantt Diagramm

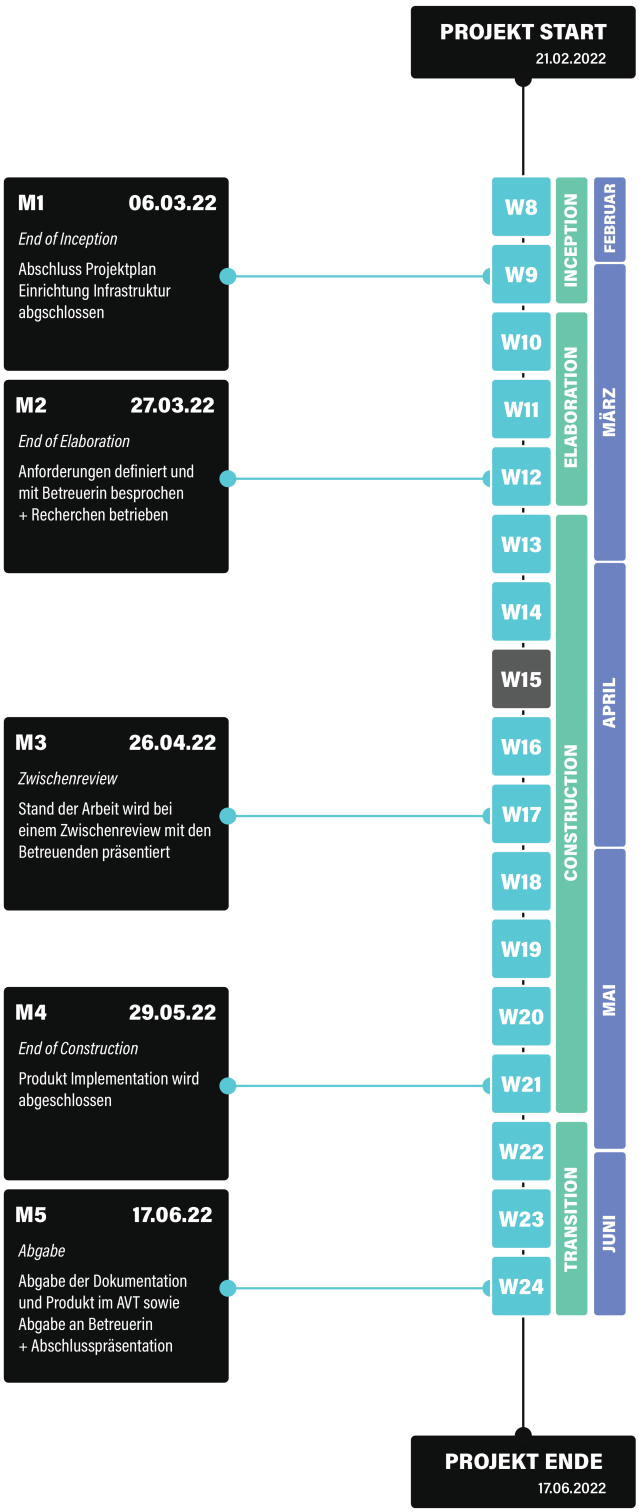


Abbildung 15.4.: Gantt Diagramm

15.5. Infrastruktur

15.5.1. Server

Von der Fachhochschule wird ein Server bereitgestellt, welcher während der Dauer des Projektes zur freien Verfügung steht. In erster Linie wird dieser für das Testen der Applikation selbst verwendet. Zusätzlich werden diverse Dienste für interne Zecke betrieben, welche vom Projektteam während der Arbeit benötigt werden. Dabei handelt es sich um mehrere Applikationen, welcher in einer Docker Umgebung laufen. Der Server umfasst folgende virtualisierte Hardware:

Hardware	Spezifikation
Hersteller	VMWare
CPU	2 vCPU 2.2GHz
RAM	4 GB
Speicherplatz	80 GB
Betriebssystem	Ubuntu 20.04 LTS

Tabelle 15.2.: Server - Virtuelle Hardware

15.5.2. Dashboard

Um eine Übersicht über alle laufenden Dienste und verwendeten Tools zu erhalten, wurde ein Dashboard eingerichtet. Dafür wird die Open-Source-Applikation Heimdall verwendet, welche selbst betrieben wird. Das Dashboard ist unter der URL https://ba.zice.ch/ während der Dauer der Arbeit erreichbar. Marius Zindel, als Inhaber der Domain hält sich das Recht vor, die Domain nach Abschluss der Arbeit zu löschen, oder anderweitig zu verwenden.

15.5.3. Version Control System

Für das Projekt wird als Version Control System (VCS) Git eingesetzt. Eine GitLab Instanz wird von der Fachhochschule zur Verfügung gestellt. Für das Projekt wurde eine neue Gruppe erstellt welche folgende Repositories enthält:

- `Documentation`
  Repository für die Dokumentation des Projektes in LaTeX
- `Deployment`
  Repository für die Serverinfrastruktur

Das Repository mit der Applikation selbst wird ebenfalls mit Git auf GitHub verwaltet. Grund für diesen Entscheid ist, dass AWS keine selbst betriebenen VCS unterstützt.

15.5.4. Build Services

Stets eine lauffähige Version der Applikation oder kompilierbare Version der Dokumentation zu besitzen ist von hoher Bedeutung. Um dies während der Projektdauer zu gewährleisten, wurde eine Jenkins Instanz aufgesetzt. Der Aufgabenbereich von Jenkins kann in zwei Kategorien aufgeteilt werden:

Continuous Integration

Automatisierter Prozess um

- Applikation zu kompilieren
- Applikation zu testen
- Dokumentation zu kompilieren

Continuous Deployment

Automatisierte Auslieferung der

- Applikation auf dem produktiven Server
- Dokumentation als PDF auf Webseite

15.5.5. Qualitätssicherung

Für die Qualitätssicherung wird der Source Codes statisch analysiert. Dafür wird die Applikation SonarQube verwendet. Diese wird auf dem eigenen Server betrieben. Eine genauere Definition und Erklärung der Qualitätssicherung ist im Abschnitt Qualitätsmanagement zu finden.

15.5.6. Entwicklungsumgebung

Folgende Tools werden zum Entwickeln der Arbeit verwendet:

Projekt	Sprache	Hersteller	Software
Frontend	TypeScript	Microsoft	Visual Studio Code
Backend	TypeScript	JetBrains	IntelliJ IDEA
	TypeScript	Microsoft	Visual Studio Code
Dokumentation	LaTeX	JetBrains	IntelliJ IDEA
	LaTeX	Microsoft	Visual Studio Code

Tabelle 15.3.: Entwicklungsumgebungen

15.5.7. Monitoring

Die internen Dienste werden mit der Open-Source Applikation Uptime-Kuma überwacht. Es wird die Verfügbarkeit sowie die Gültigkeit der Zertifikate überwacht. Bei Fehlern oder Ausfällen wird das Team direkt im Microsoft Teams Channel durch das Monitoring System informiert. Dies wurde mittels einer Webhook-Verbindung realisiert. Das Monitoring wird selber betrieben.

15.5.8. Test Infrastruktur

Für Usability Tests wird die Applikation im Main-Branch direkt betrieben. Die Auslieferung der aktuellsten Version geschieht mittels Continuous Deployment seitens Jenkins.

15.5.9. Schlüssel / Passwörter

Für das Projekt werden verschiedene Dienste verwendet. Damit entstehen einige Logins und API-Keys, welche sicher zu verwalten sind. Ebenfalls sollten alle Teammitglieder darauf Zugriff haben. Eine Bitwarden Instanz wurde dafür aufgesetzt. Bitwarden ist ein Passwort Manager, welcher unter anderem auch das Teilen vereinfacht.

15.5.10. Kommunikation

Interne Kommunikation zwischen den Autoren geschieht über Microsoft Teams. Informationen werden in den dafür eingerichteten Channels kategorisiert und archiviert:

- `General`
  Allgemeine Informationen zum Projekt
- `Alerts`
  Alarme (werden vom Monitoring versendet)
- `Documentation`
  Informationen zur Dokumentation
- `Review`
  Anstehende Code Reviews (werden von GitLab versendet)

Kommunikation mit externen Parteien wird ausschliesslich per E-Mail getätigt.

15.5.11. Projekt Management

Das Projekt wird mit der Software YouTrack von JetBrains geplant und gemanagt. Die Webapplikation wird als eigene Instanz direkt bei JetBrains in der Cloud betrieben. Folgende Aufgaben werden in der YouTrack erfasst, durchgeführt und ausgewertet:

- Product Backlog
- Scrum Board
- Stundenabrechnung
- Zeitplan
- Gantt Diagramm

15.5.12. Prototyping

Für das Erstellen eines Prototypen wird Figma eingesetzt. Abläufe können mittels Sceen Flows validiert werden.

15.5.13. Grafiken

Für die Dokumentation werden verschiedene Grafiken benötigt, welche dem Leser das geschriebene visuell besser veranschaulichen sollen. Die Grafiken werden durch das Autorenteam selber gezeichnet, wenn nicht anders angegeben. Dafür wird die Software Adobe Illustrator verwendet.

15.5.14. Notizen

Notizen wie zum Beispiel Sitzungsprotokolle werden mit Notion direkt in Markdown erstellt. Ein dafür eingerichtetes Projekt erlaubt den Austausch zwischen den Autoren intern. Das Tool wird nur für Entwürfe verwendet. Neue Anforderungen werden immer darauffolgend im Backlog in YouTack festgehalten und sind nur dort gültig.

15.5.15. Datensicherung

Für die Datensicherung werden zwei private Server verwendet, welche sich an zwei getrennten Standorten befinden. Eine genauere Beschreibung der Backup-Strategie ist im Abschnitt Datensicherung zu finden.

15.6. Projektmonitoring

In diesem Abschnitt werden zum Projektende alle aufgewendeten Arbeiten dokumentiert. Dazu wird der Soll- mit dem Ist-Zustand des Projektplans verglichen.

15.6.1. Meilensteine

Alle Meilensteine konnten wie geplant eingehalten werden. Die Bug Chaser Plattform konnte erfolgreich zum Ende des Meilensteins 4 umgesetzt werden. Nach einer Validierung in der Transition Phase gab es nur noch kleine Anpassungen an der Applikation.

Bezeichnung	Deadline	Status
M1 - End of Inception	06.03.2022	Meilenstein wurde eingehalten
M2 - End of Elaboration	27.03.2022	Meilenstein wurde eingehalten
M3 - Zwischenpräsentation	26.04.2022	Meilenstein wurde eingehalten
M4 - End of Construction	29.05.2022	Meilenstein wurde eingehalten
M5 - Abgabe	17.06.2022	Meilenstein wurde eingehalten

Tabelle 15.4.: Auswertung aller Meilensteine

15.6.2. Zeiterfassung

Allgemein

Für die Arbeit wurde ursprünglich ein Zeitaufwand von 720 Stunden geschätzt. Der effektive Zeitaufwand beträgt 744 Stunden und 40 Minuten. Dies ergibt einen Mehraufwand von 24 Stunden und 40 Minuten.

Der zusätzlich aufgewendete Mehraufwand von 3.4 % ist akzeptabel und hauptsächlich auf die Startschwierigkeiten mit AWS Amplify zurückzuführen.

Zeitaufwand pro Teammitglied

Beide Teammitglieder hielten sich gut an die vorgegebene Zeitbeschränkung. Beide überschritten die geschätzte Zeit nur minimal.

	Stundenaufwand geschätzt	Stundenaufwand geleistet	Stundenaufwand durchschnittlich pro Woche geschätzt	Stundenaufwand durchschnittlich pro Woche geleistet
Janis Wolf	360h 00min	370h 30min	22h 30min	23h 09min
Marius Zindel	360h 00min	374h 10min	22h 30min	23h 23min
Total	720h 00min	744h 40min	45h 00min	46h 32min

Tabelle 15.5.: Stundenaufwand geleistet

Die aufgewendete Zeit ist im Vergleich zwischen den Teammitgliedern ausgewogen. Dies ist auf eine sehr gute Kommunikation innerhalb des Teams zurückzuführen. Viele Treffen vor Ort steigerten die Motivation des Teams. Zudem wurde während jeder Sprint Retrospective die bisher aufgewendete Zeit im Team verglichen und bei Unterschieden bestmöglich im folgenden Sprint korrigiert.

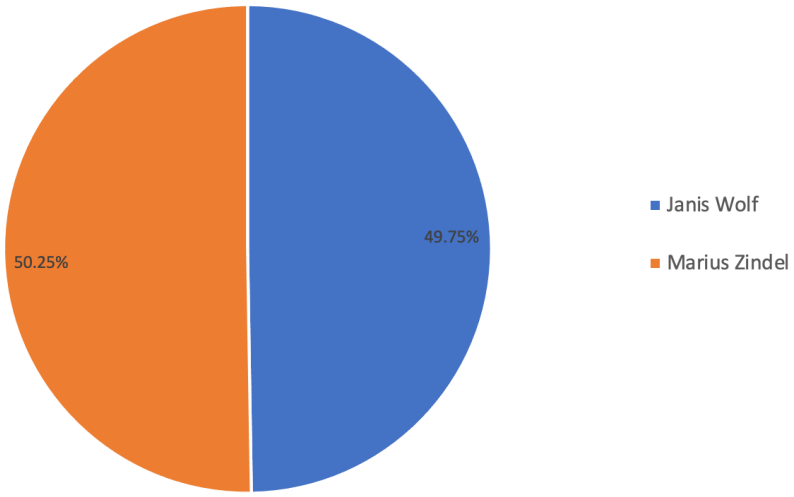


Abbildung 15.5.: Zeitaufwand pro Teammitglied

Zeitaufwand pro Kategorie

Wie im Abschnitt Zeiterfassung definiert, wurden alle getätigten Arbeiten in Kategorien erfasst.

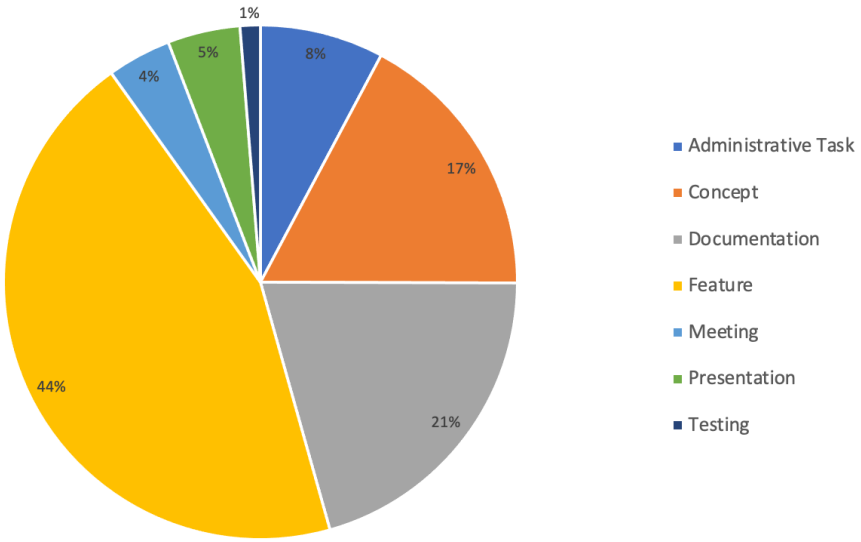


Abbildung 15.6.: Zeitaufwand pro Kategorie

Kategorie	Aufwand in Stunden
Administratives	58.0
Dokumentation	128.5
Implementation	153.3
Meeting	331.3
Präsentation	30.0
Research	34.0
Testing	9.0

Tabelle 15.6.: Zeitaufwand pro Kategorie

Auffallend ist, dass für das Produkt die meiste Zeit aufgewendet wurde. Der hohe Aufwand ist auf die vielen neuen Technologien zurückzuführen, welche während dieser Arbeit eingesetzt wurden. Dazu musste neues Wissen angeeignet werden, bevor überhaupt erst mit der Implementierung begonnen werden konnte. Die beiden Weiteren grossen Teile sind Dokumentation sowie Erarbeitung des Konzeptes. Das Testing viel am kleinsten aus. In dieser Kategorie sind jedoch keine Unit- oder Integration-Tests enthalten. Dabei handelt es sich lediglich um die Zeit, welche aufgewendet wurde, um das Produkt mit dem dahinterstehenden Konzept zu Bug Bounty zu validieren.

Zeitaufwand pro Iteration

Für die Arbeit wurden 17 Wochen vorgegeben, wovon eine Woche als Frühlingsferien gilt, an welcher nicht am Projekt gearbeitet werden darf. Aufgrund von privaten Terminen waren beide Teammitglieder jeweils eine Woche abwesend und konnten nicht weiterarbeiten. Nach Absprache mit der Betreuerin wurde entschieden, dass die Iteration 8, während der Frühlingsferien durchgearbeitet werden darf. Im Gegenzug bezog Marius Zindel in der Iteration 12 und Janis Wolf in der Iteration 14 seine freie Woche.

Dies ist auch gut in der folgenden Abbildung ersichtlich.

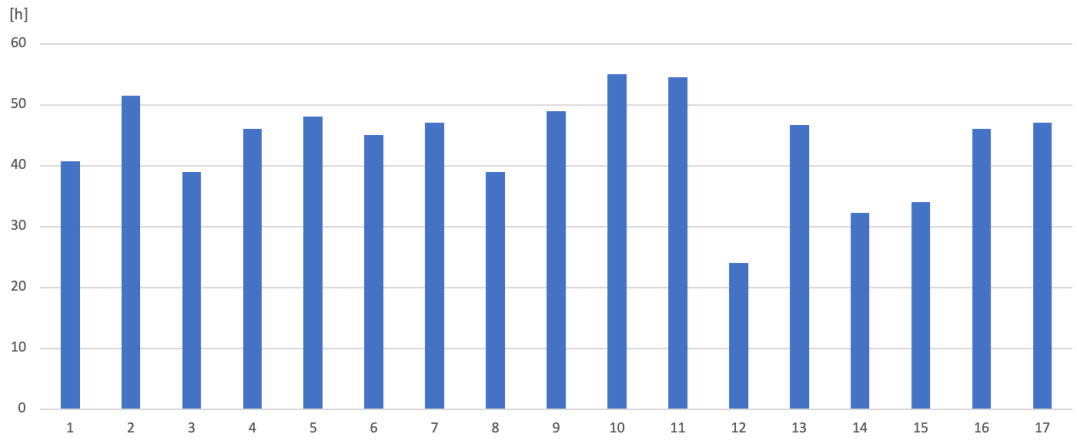


Abbildung 15.7.: Zeitaufwand pro Iteration

Iteration	Aufwand in Stunden
1	40.75
2	51.5
3	39
4	46
5	48
6	45
7	47
8	39
9	49
10	55
11	54.5
12	24
13	46.6
14	32.25
15	34
16	46
17	47

Tabelle 15.7.: Zeitaufwand pro Iteration

Wie aus der Abbildung und der Tabelle zu entnehmen ist, fallen Iteration 14 und 15 fast gleich aus, obwohl wie beschrieben Janis Wolf in Iteration 14 seine Ferien nachholte. Dies liegt daran, dass zudem in dieser Iteration ein Meilenstein definiert war und noch einige Arbeiten vor dem Abschluss umgesetzt werden mussten. Der erhöhte Aufwand von Marius Zindel wurde dann anschliessend in der Folgewoche wieder kompensiert, sodass beide wieder Teammitglieder eine ausgeglichene Arbeitszeit erreichten.

KAPITEL 16

Qualitätsmanagement

16.1. Übersicht

In diesem Abschnitt wird beschrieben welche Massnahmen ergriffen werden, um die Qualität der Arbeit auf dem definierten Standard zu halten. Dabei wird einerseits auf die Qualität des Produktes sowie auch auf diejenige der Dokumentation eingegangen.

16.2. Qualitätsmassnahmen - Applikation

16.2.1. Erfassen eines Arbeitspakets

Zum Beginn jedes neuen Sprints wird in einem Sprint Planning durchgeführt. Neue Funktionen werden während dieser Besprechung als Arbeitspakete aufgenommen. Diese werden in YouTrack als Issue, respektive als Scrum Work Item in den Product Backlog hinzugefügt. Je nach Grösse des Themas können weitere Informationen als Beschreibung in YouTrack erfasst werden. Zudem werden Labels für Frontend, AWS Amplify und Datenbank hinzugefügt, um schneller einen Überblick zu erhalten, welche Arbeiten noch offenstehen. Die dafür aufgewendete Zeit wird ebenfalls im Work Item erfasst. Die Kategorie ist immer auf Feature zu setzen.

16.2.2. Zyklus eines Arbeitspakets

Im Sprint Planning wird das Work Item einer Person zugewiesen, je nachdem was für ein Label das Item hat. Das Arbeitspaket wird in den Sprint Backlog verschoben und erhält den Status `Open`. Aufgrund der Cloud-Anbindung wird kein neuer Feature-Branch erstellt. Dies würde das Deployment um einiges erschweren. Mit dem Beginn der Arbeiten wird das Arbeitspaket in den Status `In Progress` verschoben. Wird die Arbeit am Arbeitspaket beendet, muss die aufgewendete Zeit dafür eingetragen werden. Nach Fertigstellung des Arbeitspakets wird ein Review angefordert. Das Review dient zur Überprüfung der Qualität, wird jedoch direkt im selben Branch ausgeführt. Sobald ein Arbeitspaket bereit für ein Review ist, wird dieses in YouTrack vom Status `In Progress` zu `In Review` verschoben. Der Reviewer wird sich das Arbeitspaket anschauen und ein Feedback dazu geben. Nachdem der Reviewprozess abgeschlossen wurde, markiert der Autor das Arbeitspaket als abgeschlossen. Bei einem späteren Merge in den Main-Branch wird erneut ein Review im Team durchgeführt.

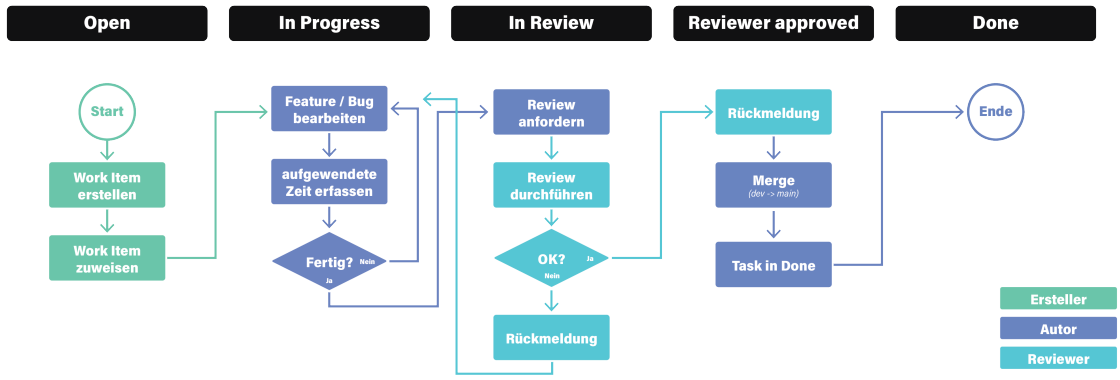


Abbildung 16.1.: Zyklus eines Arbeitspakets - Applikation

16.2.3. Style Guidelines

Als Code Style Guide wird Prettier verwendet. Prettier ist ein Code-Formatierer mit Unterstützung für verschiedene Programmiersprachen. Dieser lässt sich einfach in unser IDE einschalten und verwenden. Im Projekt wird die Standardkonfiguration verwendet, welche auch empfohlen wird.

16.2.4. Definition of Done

Ein Arbeitspaket gilt als abgeschlossen, sobald der definierte Zyklus durchlaufen wurde. In YouTrack befindet sich dann das Arbeitspaket in der Spalte `Done`.

16.2.5. Continuous Integration

Durch CI wird bei jedem push in das Git Repository eine Statische Code Analyse durchgeführt. Dies wird in dem selbst betrieben SonarQube Server umgesetzt. Im SonarQube Dashboard wird anschliessend angezeigt, wo sich mögliche Fehler im Sourcecode befinden, falls welche erkannt werden. Dies ermöglicht die Sicherstellung der Qualität des Codes.

16.2.6. Continuous Deployment

Der aktuelle Zustand der Applikation wird automatisch bei jedem Push in den Dev oder Main Branch in die Cloud hochgeladen und betrieben. Dadurch kann einfach und schnell entwickelt werden, ohne dass manuell Dateien hochgeladen und das Deployment aktualisiert werden muss. Im Fehlerfall wird automatisiert ein Rollback in der Cloud ausgeführt und die Entwickler benachrichtigt, sodass immer eine lauffähige Version der Applikation zur Verfügung steht.

16.3. Qualitätsmassnahmen - Dokumentation

16.3.1. Erfassen eines Arbeitspakets

Arbeitspakete werden nur in der Grösse einer LaTeX-Section oder Subsection erfasst. Diese werden in YouTrack als Issue, respektive als Scrum Work Item in den Product Backlog hinzugefügt. Je nach Grösse des Themas können weitere Informationen als Beschreibung in YouTrack erfasst werden. Die dafür aufgewendete Zeit wird ebenfalls im Work Item erfasst. Die Kategorie ist immer auf Documentation zu setzen.

16.3.2. Zyklus eines Arbeitspakets

Im Sprint Planning wird das Work Item einer Person zugewiesen. Das Arbeitspaket wird in den Sprint Backlog verschoben und erhält den Status `Open`. Anschliessend erstellt der Verantwortliche einen neuen Branch. Mit dem Erstellen des neuen Branches wird das Arbeitspaket in den Status `In Progress` verschoben. Wird die Arbeit am Arbeitspaket beendet, muss die aufgewendete Zeit dafür eingetragen werden. Nach Fertigstellung des Arbeitspakets wird ein neuer Merge Request erstellt und ein Review angefordert. GitLab informiert automatisiert im Microsoft Teams Channel über anstehende Reviews.
Das Review dient zur Überprüfung der Qualität. Sobald ein Arbeitspaket bereit für ein Review ist, wird dieses in YouTrack vom Status `In Progress` zu `In Review` verschoben. Der Reviewer wird sich das Arbeitspaket anschauen und ein Feedback dazu geben. Änderungsvorschläge können direkt in GitLab mit der Review Funktion gegeben werden. Diese werden vom Autor anschliessend angeschaut und verbessert.
Nachdem der Reviewprozess abgeschlossen wurde, führt der Autor ein Merge in den Main-Branch durch und markiert das Arbeitspaket als abgeschlossen.

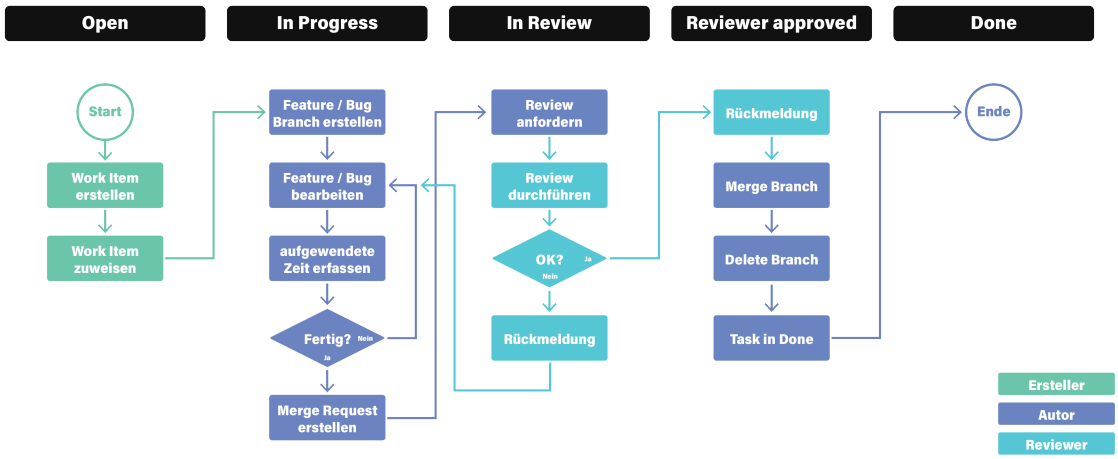


Abbildung 16.2.: Zyklus eines Arbeitspakets - Dokumentation

16.3.3. Definition of Done

Ein Arbeitspaket gilt als abgeschlossen, sobald der definierte Zyklus durchlaufen wurde. In YouTrack befindet sich dann das Arbeitspaket in der Spalte `Done`.

16.3.4. Style Guidelines

Ein Style Guide wurde für die Dokumentation nicht definiert. Allerdings wurden einige Farben für die Konsistenz der Arbeit definiert. Diese Farben wurden im Template definiert und können nun in der Dokumentation verwendet werden.

Abbildung 16.3.: Farbschema

16.3.5. Continuous Integration

Durch CI wird sichergestellt, dass immer eine aktuelle Version der Dokumentation in PDF Form bereit steht. Eine Jenkins Pipeline definiert, dass pro Git Branch jeweils bei einem Push die *main.tex* Datei kompiliert wird. Dabei wird geprüft ob die Syntax etc. korrekt ist. Bei einem Fehler wird der Status im Build Monitor als fehlgeschlagen angezeigt.

16.3.6. Continuous Deployment

Damit eine aktuelle Version der Dokumentation eingesehen werden kann, wird mit einem Webserver die aktuelle Version des Main Branch auf https://docs.ba.zice.ch veröffentlicht. Dazu wird mit traefik eine Login Middleware verwendet, damit die Arbeit nicht öffentlich zugänglich ist.
Alle alten Versionen der Dokumentation sind ebenfalls auf der Webseite verfügbar.
Die Dokumentation wird über eine Jenkins Pipeline erstellt und auf den Webserver hochgeladen.

16.4. Naming Convention

16.4.1. Branches

Neue Branches werden nach der Folgenden Naming Convention erstellt:

```
<TYPE>-<ISSUE_NR>-<NAME>
```

- **TYPE**
  Der erste Buchstabe der Kategorie des Arbeitspaketes. Zum Beispiel `b` (für Bug) oder `d` (für documentation).
- **ISSUE_NR**
  Erhaltene Issue Nummer in YouTrack. Zum Beispiel 42
- **NAME**
  Kurzer Name, welcher das Arbeitspaket eindeutig beschreibt. Werden mehrere Wörter verwendet, so müssen diese mit einem Unterstich getrennt werden. Alle Wörter werden kleingeschrieben. Zum Beispiel `section_risk_analysis`

Beispiele für valide Namen sind:

- d-42-section_risk_analysis
- b-12-user_authorisation
- f-23-sso_login

Beispiele für ungültige Namen sind:

- D-42-sectionRiskAnalysis
- x-12-user_Authorisation
- f-23-SSO_login

16.5. Besprechungen

16.5.1. Meeting mit Betreuerin

Wöchentlich findet eine Besprechung zwischen dem Projektteam und der Betreuerin statt. Falls nicht anders besprochen, findet dieses mittwochs um 14:00 Uhr statt.

Das Projektteam stellt für die Sitzung alle erledigten Arbeiten zusammen und präsentiert den aktuellen Stand. Ebenfalls werden offene Fragen notiert und an der Sitzung mit der Betreuerin diskutiert. Protokolle werden vom Projektteam geschrieben, jedoch nicht veröffentlicht.

16.5.2. Sprint Review und Retrospective

Jeweils montags trifft sich das Projektteam, um den aktuellen Stand zu besprechen. Hauptsächlich wird besprochen, wie der kommende Sprint verbessert werden kann, welche Arbeiten nicht erledigt werden konnten und damit verbunden in den kommenden Sprint übernommen werden

16.5.3. Sprint Planning

Das Sprint Planning Meeting findet anschliessend an das Sprint Review statt. Es wird im Team besprochen, welche Work Items aus dem Product Backlog in den Sprint Backlog verschoben werden.

16.5.4. Daily Scrum

Ein Daily Scrum Meeting wird ungefähr drei bis viermal pro Woche gehalten. Dabei handelt es sich um eine kurze Besprechung, wer welche Aufgaben im Verlauf des Tages abarbeitet. Falls es nötig ist, wird die Anzahl an Daily Scrum Meeting erhöht.

16.6. Zeiterfassung

Die Zeiterfassung dient zur Überprüfung der aufgewendeten Stunden. Diese Stunden werden kategorisiert direkt auf dem Work Item in YouTrack erfasst. Folgende Kategorien wurden definiert:

- `Administrative Task`
  Arbeiten, welche für die Organisation des Projektes benötigt werden. Darunter fallen zum Beispiel arbeiten an der Infrastruktur und internen Diensten.
- `Concept`
  Aufwand, welcher für das Konzept, Einarbeitung und Literaturstudium betrieben wird.
- `Documentation`
  Arbeiten an der Dokumentation werden unter diesem Tag gebucht.
- `Feature`
  Neue Funktionen werden unter der Kategorie Feature erstellt.
- `Meeting`
  Besprechungen, welche zeitlich relevant sind, werden unter dieser Kategorie geführt.
- `Presentation`
  Die Zwischen- und Schlusspräsentation wird separat unter diesem Tag rapportiert.
- `Testing`
  Zeit, welche für Tests aufgewendet werden. Hauptsächlich sind damit Tests gemeint, welche nicht automatisiert durch die CI ausgeführt werden können. Darin sind keine Unit- oder Integration-Tests enthalten. Diese werden direkt im Feature getestet.

16.7. Risikomanagement

Durch eine Risikoanalyse werden die grössten Risiken identifiziert. Alle identifizierten Risiken sind im Abschnitt Risikoanalyse zu entnehmen. Nach dem Abschluss eines Meilensteins wird die Risikoanalyse erneut evaluiert und angepasst. Der Verlauf der identifizierten Risiken wird dokumentiert.

16.8. Datensicherung

Für die Arbeit wurde eine 3-2-2 Backup-Strategie definiert:

- 3 Datenkopien
  Originaldaten und mindestens zwei Backups
- 2 verschiedene Speichermedien
  Beide Backups auf zwei getrennten Speichermedien
- 2 Offsite-Kopien
  Beide Backups befinden sich an zwei geografisch getrennten Standorten

Der Server führt täglich selbstständig eine Datensicherung folgender Dienste durch:

- Heimdall
- Jenkins
- SonarQube
- Uptime-Kuma
- Bitwarden
- Webserver (Dokumentation und Backup)

Auf dem produktiven Server ist ein Cronjob eingerichtet, welcher um 4 Uhr morgens jeweils die Repositories aktualisiert und ein ZIP Archiv des Home Verzeichnisses des docker Benutzer erstellt. In diesem Archiv sind alle relevanten Daten für das Projekt enthalten. Das Archiv wird auf dem Server abgelegt und über einen Webserver auf https://backups.ba.zice.ch bereitgestellt. Diese URL wird mit Traefik und Basic Auth geschützt, damit nur autorisierte Personen darauf Zugriff haben.

Auf zwei externen standortunabhängigen Servern läuft ebenfalls ein Cronjob. Dieser lädt einmal pro Tag das aktuelle Backup von der erwähnten URL herunter. Schlägt der Download fehlt wird ein E-Mail an den Serveradministrator gesendet.

Die Git-Repositories werden durch die beiden Backup Server täglich gesichert. Durch die Verwendung von Git sind auch ältere Versionen immer gesichert. Zusätzlich befinden sich alle Repositories in aktuellem Stand auf den persönlichen Endgeräten der Autoren.

YouTrack als Projektmanagementtool kann nicht automatisiert gesichert werden, da es direkt bei JetBrains in der Cloud betrieben wird. Manuelle Backups der Datenbank sind jedoch möglich und werden zum Ende jedes Sprints durchgeführt. Der E-Mail-Verkehr mit externen Parteien wird in den Backups der persönlichen Notebooks der Autoren gesichert.

Keine Datensicherung werden für die Dienste Microsoft Teams sowie Notion erstellt. Auf den beiden Plattformen befinden sich keine kritischen Daten für das Projekt.

16.9. Testing

16.9.1. Unit testing

In der Applikation werden automatisierte Unit Tests durchgeführt. Eine genauere Beschreibung aller durchgeführten Tests sind im Kapitel Testing vorzufinden.

16.9.2. System testing

In der Applikation werden halb automatisierte System Tests mit Cypress durchgeführt.

16.9.3. NFR testing

Die Nichtfunktionalen Anforderungen werden manuell validiert.

16.9.4. Usability testing

Die Usability der Applikation wird in einem ersten Schritt mit einem Prototyp geprüft. Zu einem späteren Zeitpunkt wird die Usability der Software selbst manuell getestet.

16.9.5. Validierung des Konzeptes

Das erarbeitete Konzept wird zum Schluss Validiert. Die Validierung ist im Teil Technischer Bericht beschrieben.

KAPITEL 17

Risikoanalyse

17.1. Identifizierte Risiken

In diesem Abschnitt werden alle Risiken für das Projekt aufgeführt, welche Einfluss auf den Verlauf der Arbeit haben könnten. Nach jedem Erreichen eines Meilensteins wird die Risikoabschätzung erneut durchgeführt und dokumentiert. Folgende Risiken wurden identifiziert:

ID	Risiko	Schaden	Eintritts-wahrscheinlichkeit
R01	Anforderungen	mittel	mittel
R02	Komplexität	hoch	hoch
R03	Neue Technologien	mittel	hoch
R04	Kommunikation	hoch	mittel
R05	Externe Komponenten	hoch	mittel
R06	Datenverlust	sehr hoch	niedrig
R07	Scope zu gross	hoch	mittel
R08	Krankheit / Unfall	mittel	niedrig

Tabelle 17.1.: Identifizierte Risiken

R01 - Anforderungen

Beschreibung	Es werden während des Projektes neue Anforderungen hinzugefügt, oder die definierten Anforderungen ändern sich markant. Es wird nicht unterschieden zwischen bewussten oder unbewussten Änderungen der Anforderungen.
Vorbeugung	Der Scope des Projektes muss in der Planungsphase klar und detailliert definieren werden.
Verhalten beim Eintreten	Den Scope den neuen, respektive geänderten Anforderungen anpassen. Gegebenenfalls andere Anforderungen aus dem Scope entfernen oder in ihrer Funktionalität minimieren.

R02 - Komplexität

Beschreibung	Das Produkt wird zu komplex, die Funktionalität ist zu hoch oder die Architektur wurde zu schlecht geplant und umgesetzt.
Vorbeugung	Die Komplexität wird mit Prototypen vorbeugend abgeschätzt.
Verhalten beim Eintreten	Die Funktionalität verringern, falls möglich die Architektur so umbauen, dass die Probleme behoben werden können.

R03 - Neue Technologien

Beschreibung	Für die Arbeit werden neue Technologien eingesetzt, für welche das Know-How der Studierenden noch nicht vorhanden ist.
Vorbeugung	Genaue Auswertung und Evaluierung der Technologien während der Elaboration Phase auch in Bezug auf das Know-How der Studierenden.
Verhalten beim Eintreten	Neues Wissen aneignen oder Funktionalität verringern.

R04 - Kommunikation

Beschreibung	Ungenügende Kommunikation im Projektteam oder mit externen Schnittstellen.
Vorbeugung	Wöchentliche Meetings werden durchgeführt. Zusätzlich wird ein Verantwortlicher pro Bereich definiert.
Verhalten beim Eintreten	Zusätzliche Sitzungen einführen.

R05 - Externe Komponenten

Beschreibung	Fehlende Dokumentation oder Support zu externen Komponenten
Vorbeugung	Genaues Prüfen der Schnittstellenspezifikation bei der Evaluierung
Verhalten beim Eintreten	Andere Komponenten evaluieren und gegebenenfalls auf diese wechseln oder Anforderungen anpassen.

R06 - Datenverlust

Beschreibung	Verlust von projektrelevanten Daten durch Hardware- oder Anwendungsfehler.
Vorbeugung	Backup Strategie definieren und umsetzen
Verhalten beim Eintreten	Daten aus Backup einspeisen. Gegebenenfalls Funktionalität anpassen.

R07 - Scope zu gross

Beschreibung	Der Scope der Anforderungen ist für die geplante Zeit zu gross.
Vorbeugung	Anforderungen sauber und genau definieren. Klar kommunizieren, was in der geplanten Zeit möglich / realistisch ist.
Verhalten beim Eintreten	Scope mit dem Kunden verkleinern.

R08 - Krankheit / Unfall	
Beschreibung	Durch die aktuelle Pandemiesituation können Ausfälle krankheitshalber nicht ausgeschlossen werden.
Vorbeugung	Empfehlungen des Bundesamt für Gesundheit, respektive des Krisenstabs der OST bestmöglich einhalten.
Verhalten beim Eintreten	Anforderungen des Projekts anpassen.

17.1.1. Risikomatrix

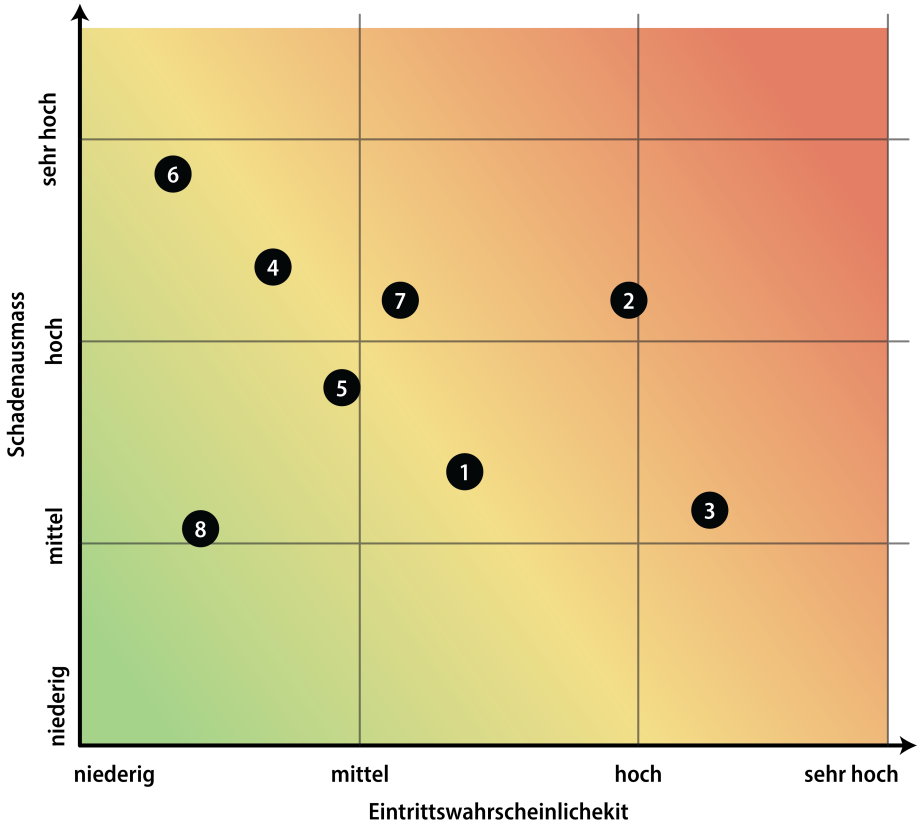


Abbildung 17.1.: Risikomatrix

17.2. Umgang mit Risiken

Da damit gerechnet wird, dass gewisse Risiken eintreffen werden, muss der Umgang mit Risiken definiert werden. In den folgenden Abschnitten wird auf diese Punkte eingegangen.

17.2.1. Zeitreserven

Der erstellte Zeitplan ist zur groben Übersicht. Die definierten Projektphasen sollen, wenn möglich eingehalten werden. Gegen Ende der Phasen sind aber immer noch Zeitreserven eingeplant. Diese können verwendet werden, falls man sich bei der Planung verschätzt hat.

17.2.2. Risikoüberwachung

Durch die Risikoanalyse können die verschiedenen Risiken gut überwacht werden. Allerdings muss die Analyse während dem Projekt aktualisiert werden, um bedeutend zu sein. Die Neubeurteilung findet jeweils nach dem Abschluss eines Meilensteins statt.

17.3. Verlauf

17.3.1. Neubeurteilung Meilenstein 1 - End of Inception

Ein Projektplan wurde aufgestellt sowie der Hauptanteil an administrativen Arbeiten konnte abgeschlossen werden. Es sind keine Risiken eingetroffen. Zudem konnten folgende zwei Risiken minimiert werden:

- R04 - Kommunikation
  Der wesentliche Ablauf des Projektes wurde im Projektplan festgehalten. Bei Unklarheiten kann darauf zurückgegriffen werden. Die Eintrittswahrscheinlichkeit wurde von mittel zu niedrig eingestuft.
- R06 - Datenverlust
  Eine Backupstrategie wurde umgesetzt. Das Risiko, Daten zu verlieren gilt nun als sehr unwahrscheinlich. Die Eintrittswahrscheinlichkeit wurde daher von niedrig zu sehr niedrig eingestuft.

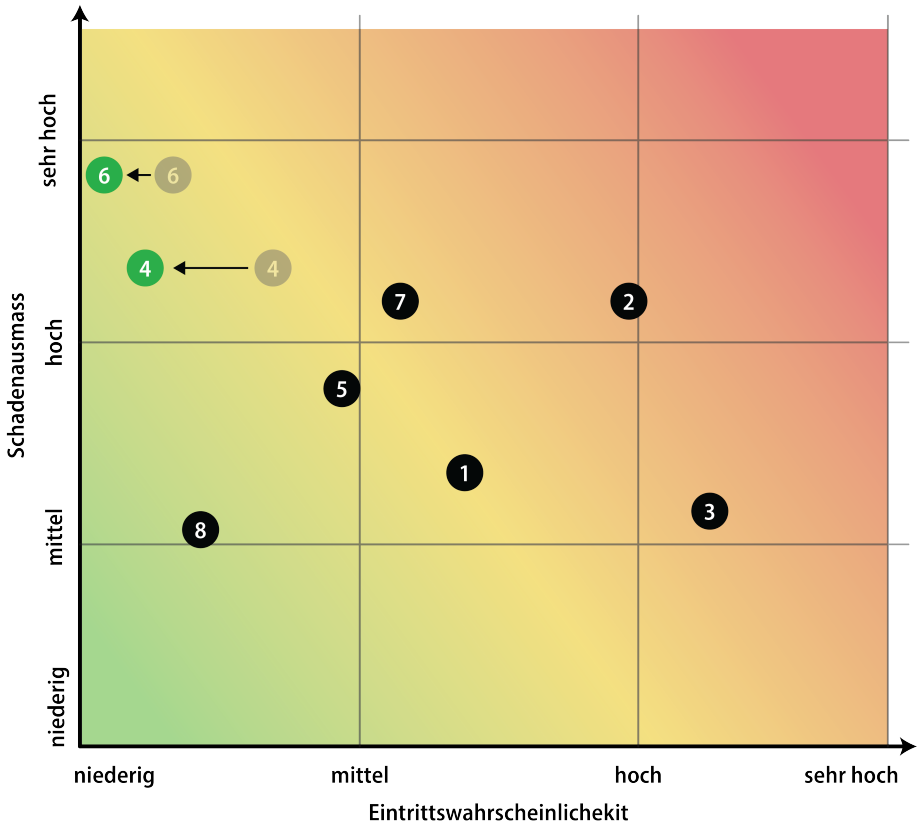


Abbildung 17.2.: Aktualisierte Risikomatrix - M1

17.3.2. Neubeurteilung Meilenstein 2 - End of Elaboration

Die Anforderungen an das Projekt wurden genauer definiert und eine Analyse der Technologien wurde durchgeführt. Zum aktuellen Stand sind keine Risiken eingetroffen. Vorbeugenden wird folgende Eintrittswahrscheinlichkeit neu evaluiert.

- R03 - Neue Technologien
  Es wurde entschieden, dass die Applikation mit verschiedenen neuen Technologien umgesetzt wird. Dies führt dazu, dass die Eintrittswahrscheinlichkeit nach oben verschoben werden muss. Das Risiko wird nun neu als sehr hoch eingestuft. Es ist mit erhöhtem Mehraufwand zu rechnen, bevor mit der Implementation gestartet werden kann.

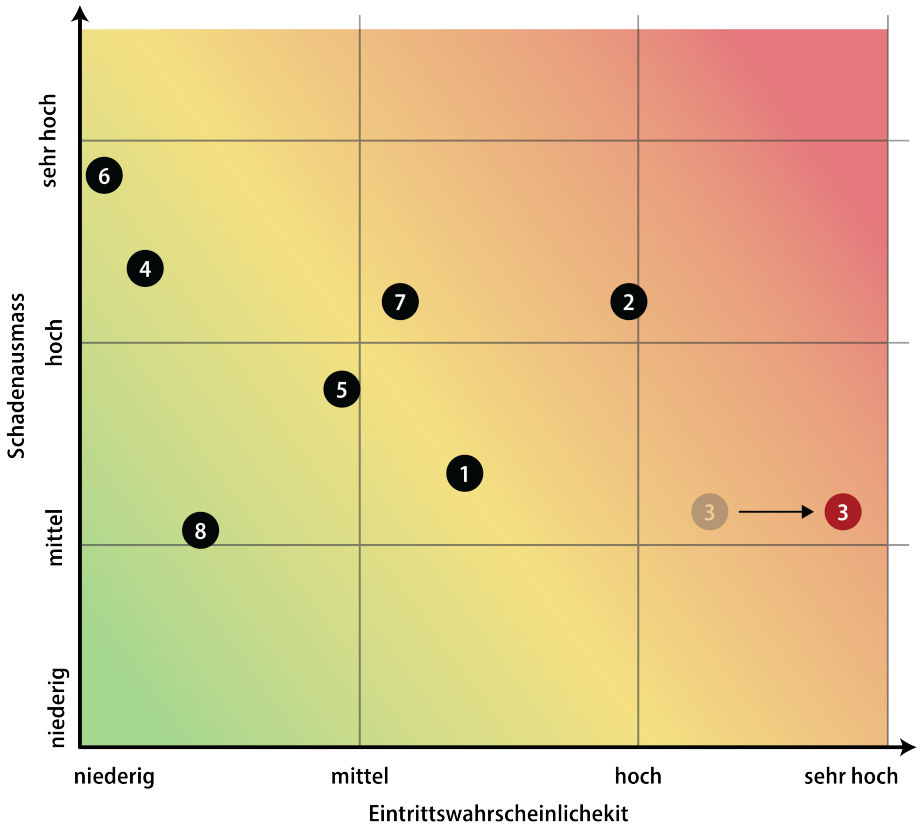


Abbildung 17.3.: Aktualisierte Risikomatrix - M2

17.3.3. Neubeurteilung Meilenstein 3 - Zwischenpräsentation

Durch die Einarbeitung in die Materie wurde ein tieferes Verständnis für die gewählten Technologien geschaffen. Der Start mit AWS Amplify wurde durch die unvollständige Dokumentation erschwert und das initiale Setup dauerte länger als geplant. Folgende Risiken sind eingetroffen:

ID	Risiko	Beschreibung	Getroffene Handlungen
R03	Neue Technologien	Bisher konnte noch keiner der beiden Teammitglieder Erfahrungen bei der Entwicklung von Applikationen in der Cloud sammeln. Wie erwartet benötigt die Einarbeitung zusätzliche Zeit.	Die Eintrittswahrscheinlichkeit wird weiterhin als sehr hoch eingeschätzt. Durch das agile Projektmanagement sind direkt keine Auswirkungen zu spüren. Es wird jedoch entschieden, dass zunächst auf die Hauptfunktionalität der Applikation fokussiert wird.
R05	Externe Komponenten	Die Dokumentation von AWS Amplify ist sehr minimalistisch gehalten. Zudem gibt es Probleme mit der AWS Amplify CLI. Support aus der Community gibt es zwar direkt auf GitHub, jedoch sind die vorgeschlagenen Lösungen nicht immer zufriedenstellend.	Zusätzliche Unterstützung wird bei der Community eingeholt. Des Weiteren wird direkt mit der Dokumentation von AWS gearbeitet, welche die Services hinter AWS Amplify besser dokumentiert.

Tabelle 17.2.: Eingetroffene Risiken - M3

Dadurch wurde die identifizierten Risiken wie folgt aktualisiert:

- R02 - Komplexität
  In den zurückgelegten Phasen konnte ein gutes Verständnis für die benötigte Komplexität erarbeitet werden. Durch gute Planung und der notwendigen Recherche wird auch diese Eintrittswahrscheinlichkeit verringert.

- R04 - Kommunikation
  Die Kommunikation funktioniert bisher sehr gut. Das Autorenteam versteht sich ausgesprochen gut und es entstehen kaum Meinungsverschiedenheiten. An der Zwischenpräsentation konnten zudem der Experte und die Gegenleserin kennengelernt werden. Die Eintrittswahrscheinlichkeit dieses Risikos wird daher verringert.

- R05 - Externe Komponenten
  Die eher schlechte Dokumentation von AWS Amplify führt zu zusätzlichem Mehraufwand. Die Eintrittswahrscheinlichkeit dieses Risikos wurde dadurch erhöht.

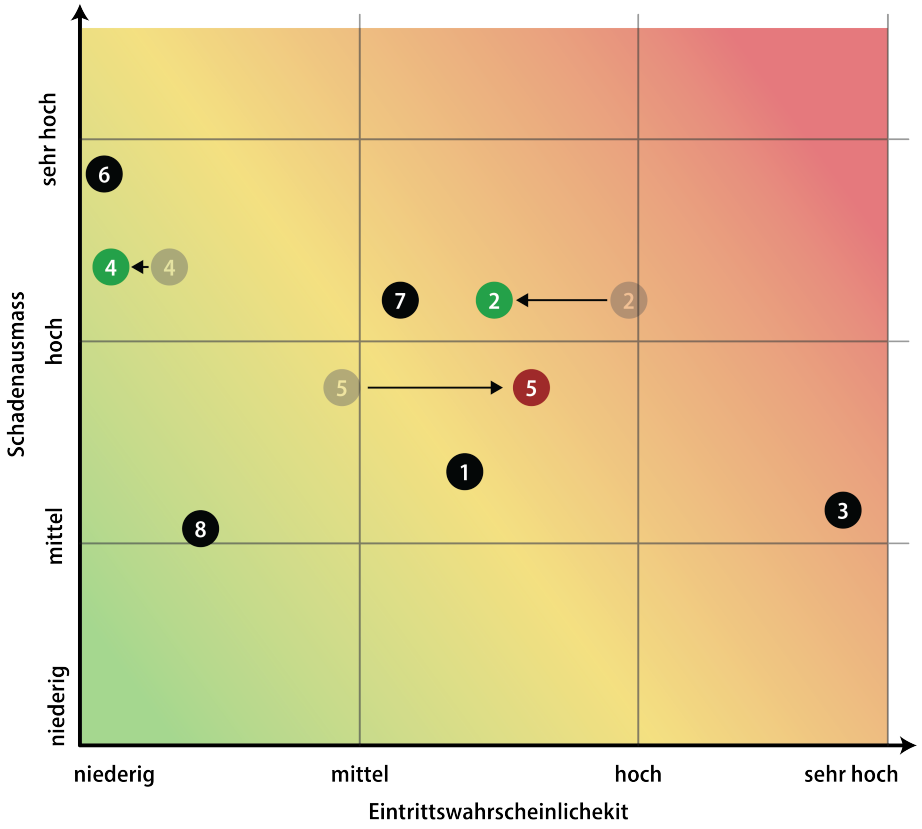


Abbildung 17.4.: Aktualisierte Risikomatrix - M3

17.3.4. Neubeurteilung Meilenstein 4 - End of Construction

Die Entwicklungsphase der Applikation wurde erfolgreich abgeschlossen. Das Produkt kann in der nächsten Phase validiert werden. Folgende Risiken können daher wie folgt aktualisiert werden:

- R01 - Anforderungen
  Mit dem Ende der Entwicklungsphase werden keine neuen Anforderungen hinzukommen. Alle benötigten Anforderungen wurden umgesetzt. Die Applikation kann für Ihren Zweck verwendend werden.
- R02 - Komplexität
  Ebenfalls kann das Risiko Komplexität verringert werden. Es werden keine weiteren Features umgesetzt.
- R03 - Neue Technologien
  Für die Validierung und Dokumentation des Projektes werden nur noch bereits bekannte Technologien eingesetzt. Dieses Risiko kann deshalb ausgeschlossen werden.
- R05 - Externe Komponenten
  Für die letzte Phase werden keine weiteren externen Komponenten benötigt. Das Risiko kann mitigiert werden.
- R07 - Scope zu gross
  Das Risiko kann minimiert werden. Ein grosser Teil der Arbeit wurde umgesetzt.
- R08 - Krankheit / Unfall
  Der noch zu leistende Aufwand für das Projekt ist im Vergleich zum Beginn der Arbeit natürlich verkleinert worden. Ein Ausfall kann als sehr unwahrscheinlich eingeschätzt werden. Zudem ist das Schadenausmass deutlich kleiner.

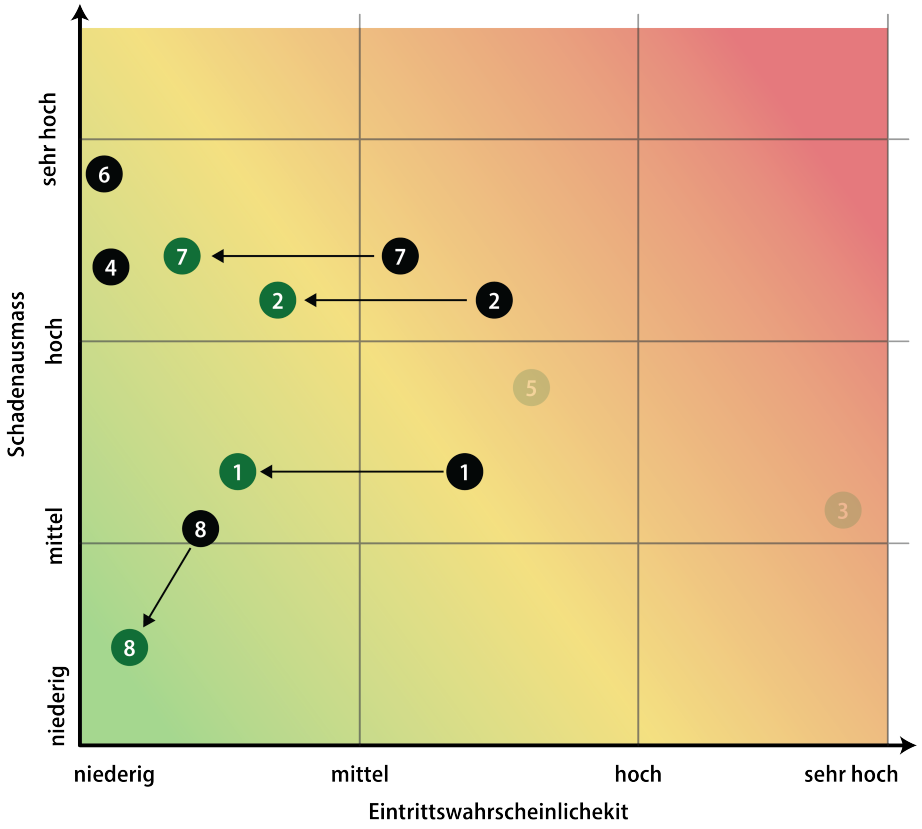


Abbildung 17.5.: Aktualisierte Risikomatrix - M4

17.3.5. Neubeurteilung Meilenstein 5 - Abgabe

Das Projekt konnte in der dafür vorgesehenen Zeit abgeschlossen werden. Die restlichen, noch offenen Risiken können daher alle ganz ausgeschlossen werden.

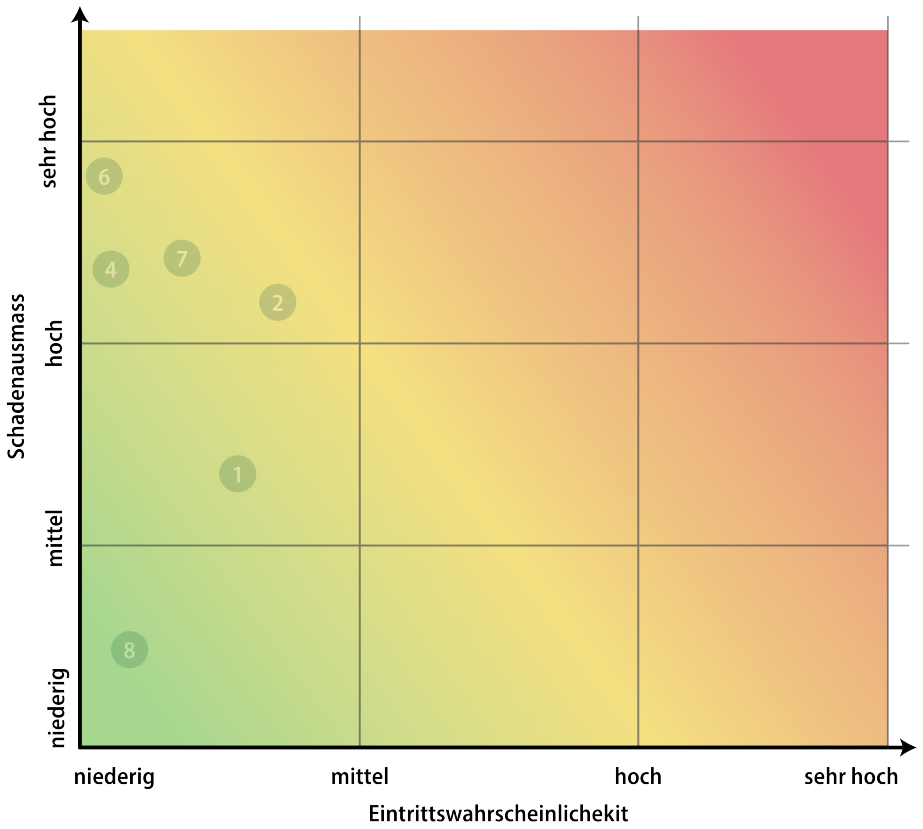


Abbildung 17.6.: Aktualisierte Risikomatrix - M5

Teil V.

Anhang

ANHANG A

Verzeichnisse

A.1. Akronyme

AWS Amazon Web Services. 11, 12, 42, 44, 45, 47, 57, 63, 65, 68, 79, 80, 85, 87–90, 105, 128

BBP Bug Bounty Programm. 24

CDN Content Deliviery Network. 47

CRUD Create Read Update Delete. 16

EDU Education. 24

IAM Identity and Access Management. 34

JWT JWT. 74

MFA Multi-Faktor-Authentifizierung. 21

SSO Single Sign On. 20

VCS Version Control System. 105

VDP Vulnerability Disclosure Program. 24

A.2. Glossar

Amazon Web Services Amazon Web Services ist ein US-amerikanischer Cloud-Computing-Anbieter, welcher weltweit Rechenzentren betreibt. 57

Amazon Web Services Amazon Web Services ist ein Cloud provider, welcher diverse Services anbietet. Alternative dazu sind Google Cloud Microsoft Azure. 1, 4

AWS Amplify Amplify ist eine Sammlung von Services in der AWS Cloud, welche das entwickeln von mobilen Web Applikationen vereinfacht. Es wird dabei die Konfiguration vieler Services erleichtert und lässt dadurch eine schnelle Entwicklung für solche Applikationen zu. 4, 11, 44, 45, 47, 49, 54, 57, 58, 61–63, 67–69, 71, 75, 85–87, 89, 90, 93, 94, 109, 113, 128

AWS AppSync AppSync ist ein Dienst in AWS, welcher eine GraphQL Schnittstelle als API anbietet. So kann auf die Datenbank mit GraphQL zugegriffen werden. 47, 49, 61, 62, 69

CloudTrail CloudTrail ist zuständig für die Sicherung und Monitoring diverser Aktivitäten im AWS Konto. Diese Daten und Logs können dann von CloudWatch durchsucht werden. 75

CloudWatch CloudWatch ist für die Analyse, Sammlung und das Korrelieren von Log Daten in AWS zuständig. Die Log Daten, gesammelt meistens in einem S3 Bucket, können in diesem Dienst durchsucht und analysiert werden. 75

Cognito Das IAM System von AWS. Es ist zuständig für die Verwaltung von Benutzer und Gruppen, sowie verschiedenen Userpools und die Authentifizierung. 47, 65, 67, 71, 73–75, 79, 87, 94

Content Deliviery Network Dieses Netzwerk von Servern ist auf der gesamten Welt verteilt und wird zur schnellen Auslieferung von statischem Web Content verwendet. Cloudfront, das CDN von AWS, hosted die Bug Chaser Applikation und stellt diese dem Benutzer am nächst möglich geografischen Standort zur Verfügung. 47

DynamoDB DynamoDB ist die NoSQL Datenbank, welche AWS in der Cloud anbietet. Sie basiert auf dem Key-Value Konzept und ist durch das extrem performant. Es lässt sich aber durch diese Eigenschaft kein klassisches relationales Datenbankmodel konfigurieren. 69, 79

Forensic Readiness Vorarbeit, um im Falle einer forensischen Untersuchung von Sicherheitsvorfällen bestmöglich vorbereitet zu sein. 75

Git Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien. 105, 119

Lambda Lambda Funktionen sind Code Funktionen, welche basierend auf Ereignissen durch z.B. die API ausgeführt werden. Die Funktionen können sehr einfache Operationen übernehmen, jedoch auch komplexe Operationen in verschiedenen AWS Service durchführen. Die Funktionen sind sehr kostengünstig, da nur verrechnet wird, wie viel Zeit und Ressourcen die Funktion verwendet hat. 4, 42, 45, 47, 48, 65–67, 73, 74, 79, 90, 93

MIT Lizenz Open Source Lizenz für Software und ihren Code. 50, 51

Route 53 Route 53 ist ein hochverfügbares Domain Name System in der AWS Cloud. Es wird für das Hosting von Web Applikationen und die Nutzung von Zertifikaten benötigt. 75, 87, 89

S3 Bucket Der Dienst S3 wird in der AWS Cloud für die Datenspeicherung von Daten verwendet. Er kann über die API direkt angesprochen werden. Ein Bucket kann auch in Amplify eingebunden und verwendet werden. 45, 47, 56, 67, 71, 72, 75, 78, 87, 92, 93

SEO Search Engine Optimization. 44

Serverless Functions Datenverarbeitungsservice, welcher Code beim Eintreten eines bestimmten Ereignisses ausführt. 33, 35, 42, 63

Token Ein Token ist ein eindeutiger String / Text, welcher in der Bug Chaser Applikation zum Beitreten eines privaten Programs verwendet wird. 26, 94

Vulnerability Management Die Verwaltung und de Umgang von Schwachstellen in der Software Entwicklung wird als Vulnerability Management definiert. 3, 96

A.3. Abbildungsverzeichnis

A.4. Tabellenverzeichnis

A.5. Literaturverzeichnis

[1] ISO/IEC 25010, *ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*, Std., 2011.

[2] Web Accessibility Initiative, *Web Content Accessibility Guidelines (WCAG)*, Std. [Online]. Available: https://www.w3.org/WAI/WCAG2AA-Conformance

[3] I. Meta Platforms. React documentation. [Online]. Available: https://reactjs.org/

[4] Vercel. Nextjs documentation. [Online]. Available: https://nextjs.org/

[5] Google. Angular documentation. [Online]. Available: https://angular.io/

[6] A. W. Services. Cloud computing with aws. [Online]. Available: https://aws.amazon.com/what-is-aws/

[7] G. Inc. Magic quadrant for cloud infrastructure and platform services. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-271OE4VR&ct=210802&st=sb

[8] G. Foundation. What is the graphql foundation? [Online]. Available: https://graphql.org/foundation/

[9] F. D. Mehta, "Introduction to software engineering," *Software Engineering 1*, p. 18, September 2020.

[10] T. Kälin, "Project planning," *Software Engineering 2*, pp. 16–21, Februar 2021.

[11] J. Sutherland and K. Schwaber. The 2020 scrum guide. [Online]. Available: https://scrumguides.org/scrum-guide.html

ANHANG B

Threat Model

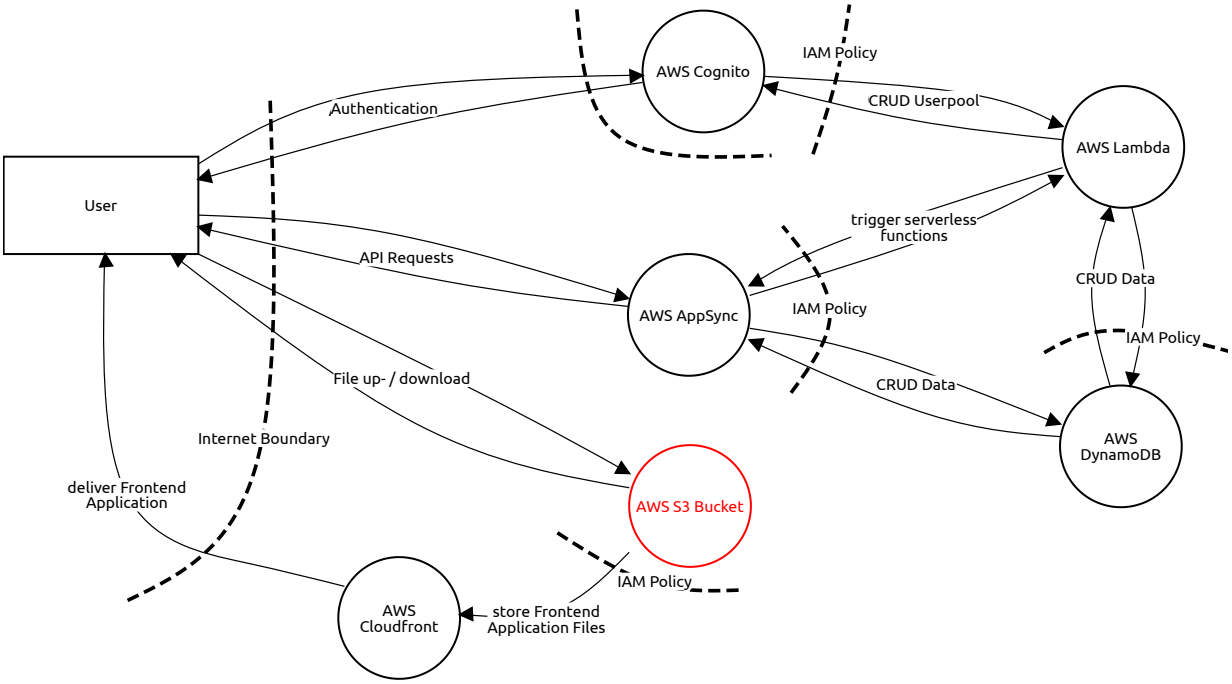
Threat model report for New threat model

Owner:
Marius Zindel
Reviewer:
Janis Wolf
Contributors:

High level system description

Report printing failed: cancelled

BugChaser Backend



User (External Actor)

Description:

Generic spoofing threat

Spoofing, Mitigated, Medium Priority

Description:
A generic spoofing threat

Mitigation:
Authentication with AWS Cognito

Generic repudiation threat

Repudiation, Mitigated, Low Priority

Description:
A generic repudiation threat

Mitigation:
Secure logging of any API action

AWS Cognito (Process)

Description:

Generic spoofing threat

Spoofing, Mitigated, Medium Priority

Description:
A generic spoofing threat

Mitigation:
PKI with SSL/TLS certificates,

Generic tampering threat

Tampering, Mitigated, Medium Priority

Description:
A generic tampering threat

Mitigation:
Mitigated over IAM ACL Policies.

Generic repudiation threat

Repudiation, Mitigated, Medium Priority

Description:
A generic repudiation threat

Mitigation:
Logging of user loggon

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Data is only accessible when owner of user object and logged in.

Generic DoS threat

Denial of service, Mitigated, Medium Priority

Description:
A generic DoS threat

Mitigation:
Brute force attacks are mitigated by Cognito itself

Generic elevation threat

Elevation of privilege, Mitigated, Medium Priority

Description:
A generic elevation of privileges threat

Mitigation:
Mitigated with privilege ownership

AWS AppSync (Process)

Description:

Generic spoofing threat

Spoofing, Mitigated, Medium Priority

Description:
A generic spoofing threat

Mitigation:
PKI with SSL/TLS certificates.

Generic tampering threat

Tampering, Mitigated, Medium Priority

Description:
A generic tampering threat

Mitigation:
Mitigated over IAM ACL Policies.

Generic repudiation threat

Repudiation, Mitigated, Medium Priority

Description:
A generic repudiation threat

Mitigation:
All actions of the client are logged in AWS CloudFront.

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Connects with the client over a secured TLS connection.

Generic DoS threat

Denial of service, Mitigated, Medium Priority

Description:
A generic DoS threat

Mitigation:
AppSync uses security best practices that AWS has developed operating large systems at scale in the cloud, with built-in DDoS protection in all its GraphQL API endpoints.

Generic elevation threat

Elevation of privilege, Mitigated, High Priority

Description:
A generic elevation of privileges threat

Mitigation:
Authorization by group memberships

AWS Lambda (Process)

Description:

Generic spoofing threat

Spoofing, Mitigated, Medium Priority

Description:
A generic spoofing threat

Mitigation:
The service is only available internally in the cloud.

Generic tampering threat

Tampering, Mitigated, Medium Priority

Description:
A generic tampering threat

Mitigation:
Mitigated over IAM ACL Policies.

Generic repudiation threat

Repudiation, Mitigated, Medium Priority

Description:
A generic repudiation threat

Mitigation:
Secure logging by AWS AppSync

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Trigger of serverless functions is protected by group membership and ownership checks

Generic DoS threat

Denial of service, Mitigated, Medium Priority

Description:
A generic DoS threat

Mitigation:
Protected by AWS AppSync

Generic elevation threat

Elevation of privilege, Mitigated, Medium Priority

Description:
A generic elevation of privileges threat

Mitigation:
Protected by AWS AppSync and IAM Policies

Authentication (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

Authentication (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, High Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

API Requests (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

trigger serverless functions (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

CRUD Userpool (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

CRUD Userpool (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

trigger serverless functions (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

API Requests (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, High Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

AWS DynamoDB (Process)

Description:

Generic spoofing threat

Spoofing, Mitigated, Medium Priority

Description:
A generic spoofing threat

Mitigation:
The service is only available internally in the cloud.

Generic tampering threat

Tampering, Mitigated, Medium Priority

Description:
A generic tampering threat

Mitigation:
Mitigated over IAM ACL Policies.

Generic repudiation threat

Repudiation, Mitigated, Medium Priority

Description:
A generic repudiation threat

Mitigation:
Secure logging by AWS AppSync

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Access on private data protected by group membership

Generic DoS threat

Denial of service, Mitigated, Medium Priority

Description:
A generic DoS threat

Mitigation:
Protected by AWS AppSync

Generic elevation threat

Elevation of privilege, Mitigated, Medium Priority

Description:
A generic elevation of privileges threat

Mitigation:
Protected by AWS AppSync and IAM Policies

CRUD Data (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

CRUD Data (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

CRUD Data (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

CRUD Data (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit in a private network

AWS S3 Bucket (Process)

Description:

Generic spoofing threat

Spoofing, Mitigated, Medium Priority

Description:
A generic spoofing threat

Mitigation:
PKI with SSL/TLS certificates

Generic tampering threat

Tampering, Mitigated, Medium Priority

Description:
A generic tampering threat

Mitigation:
Mitigated over IAM ACL Policies.

Generic repudiation threat

Repudiation, Mitigated, Medium Priority

Description:
A generic repudiation threat

Mitigation:
Server access logging is enabled

Generic information disclosure threat

Information disclosure, Open, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Data is stored encrypted.
Files are only accessible over user identity ID and file key.

Generic DoS threat

Denial of service, Mitigated, Medium Priority

Report printing failed: cancelled

Description:
A generic DoS threat

Mitigation:
Mitigation or prevention for the threat

Generic elevation threat

Elevation of privilege, Mitigated, Medium Priority

Description:
A generic elevation of privileges threat

Mitigation:
Authorization by user ID.

File up- / download (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

File up- / download (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, High Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

AWS Cloudfront (Process)

Description:
CDN

Generic information disclosure threat

Information disclosure, Mitigated, Low Priority

Description:
A generic information disclosure threat

Mitigation:
Public available data

Generic DoS threat

Denial of service, Mitigated, Medium Priority

Description:
A generic DoS threat

Mitigation:
Secured over CDN of AWS CloudFront

Generic spoofing threat

Spoofing, Mitigated, Medium Priority

Description:
A generic spoofing threat

Mitigation:
PKI with SSL/TLS certificates

deliver Frontend Application (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, High Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

store Frontend Application Files (Data Flow)

Description:

Generic information disclosure threat

Information disclosure, Mitigated, Medium Priority

Description:
A generic information disclosure threat

Mitigation:
Encryption of data in transit

ANHANG C

Wireframes (Figma)

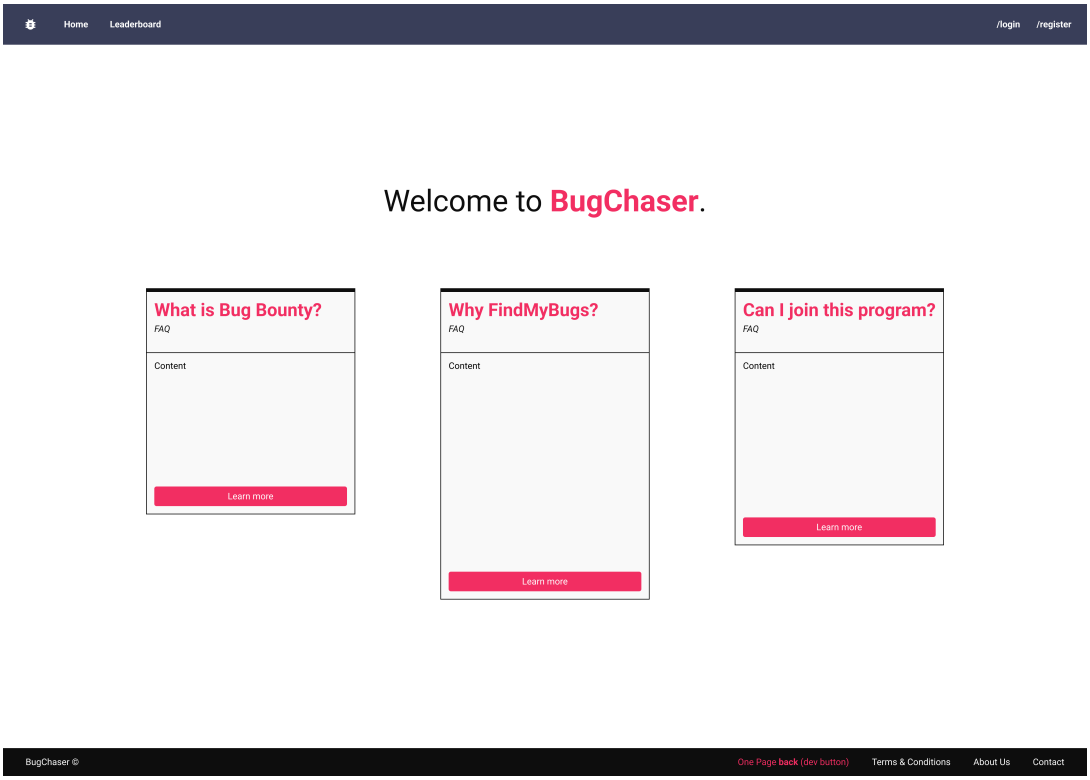


Abbildung C.1.: Index Seite

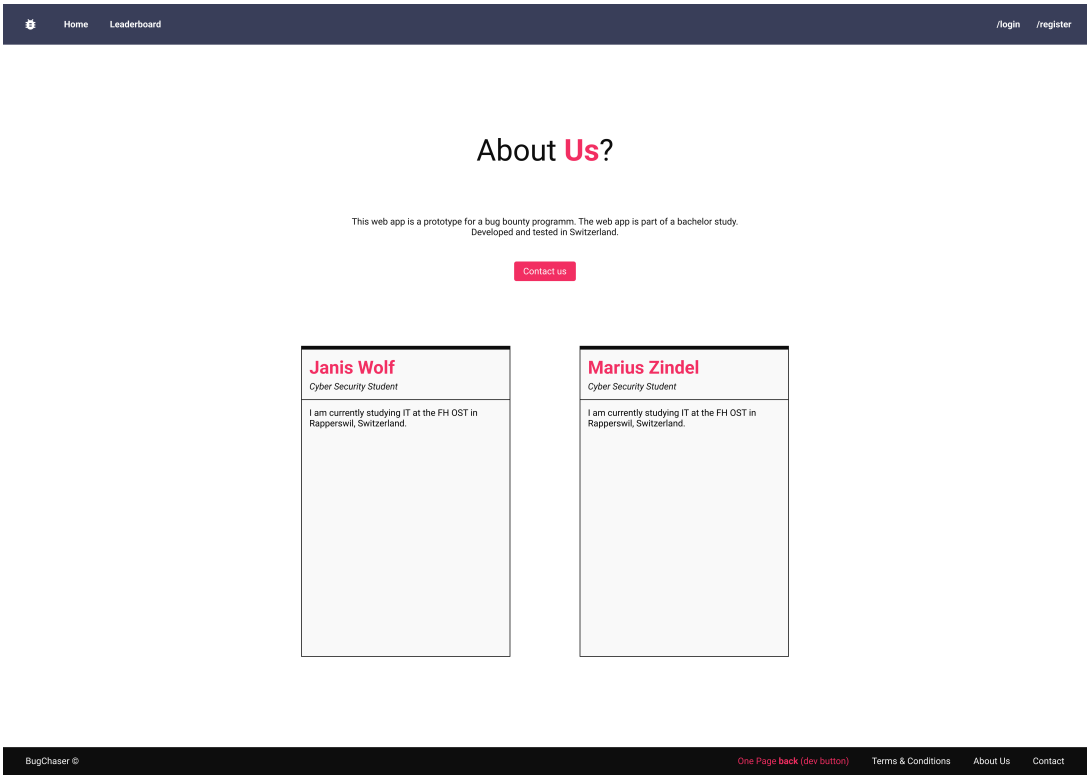


Abbildung C.2.: Über uns Seite

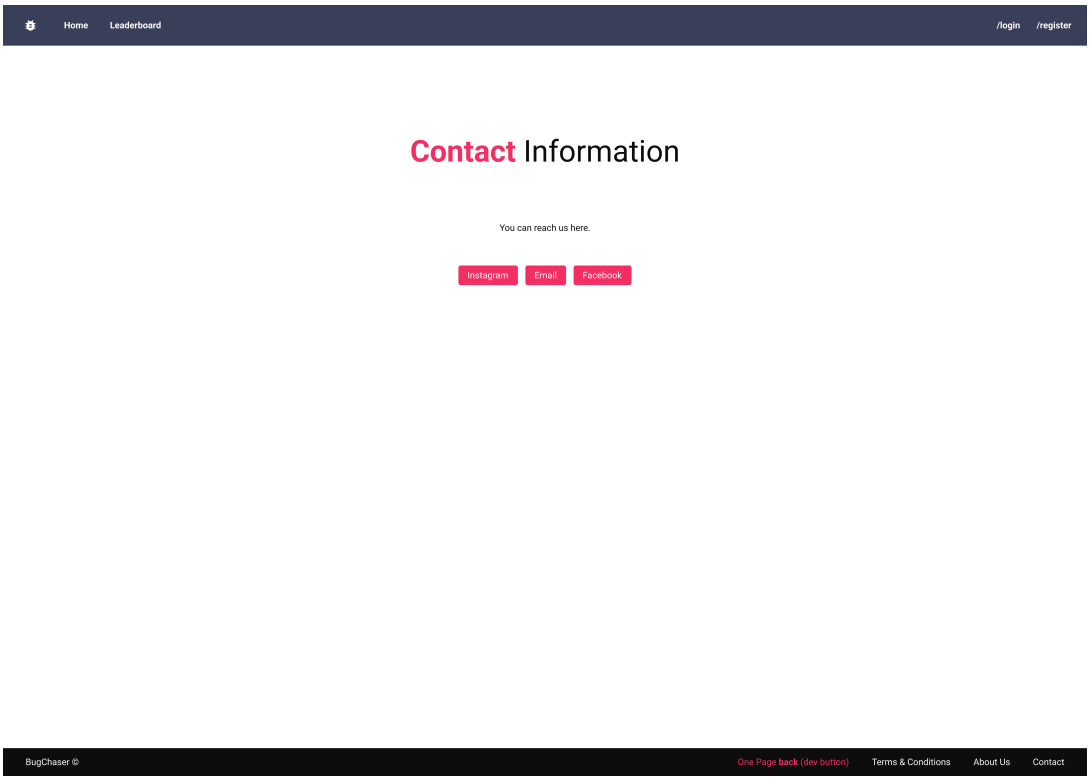


Abbildung C.3.: Kontakt Seite

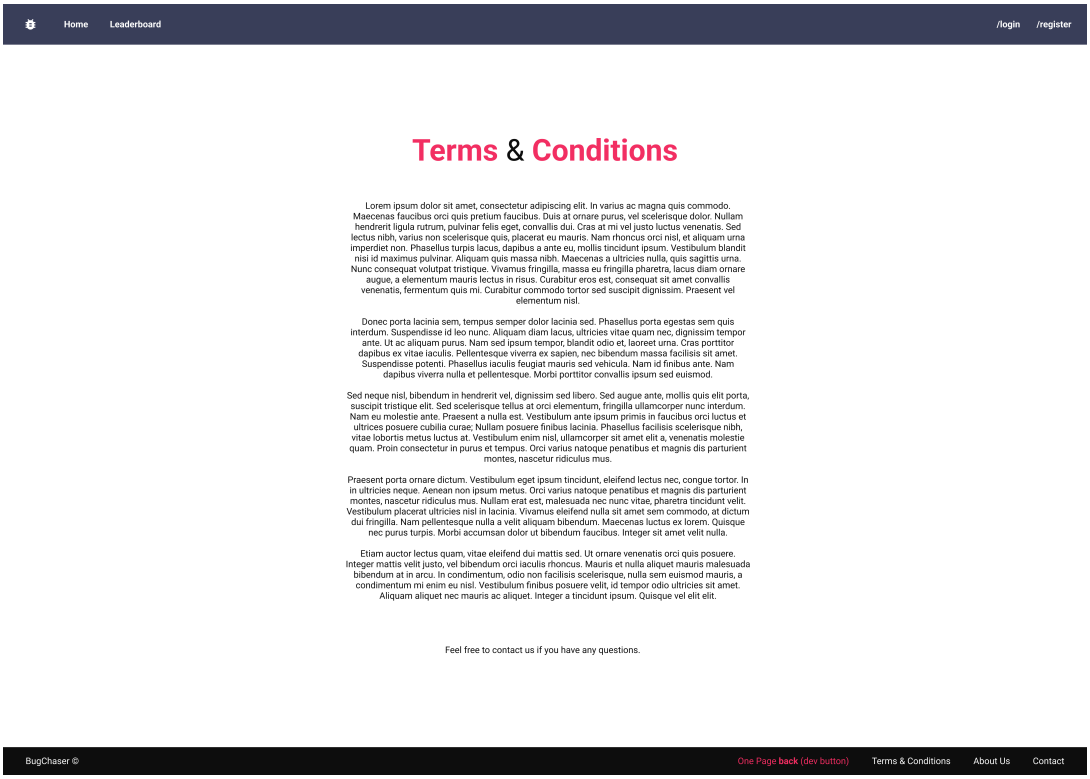


Abbildung C.4.: Terms and conditions Seite

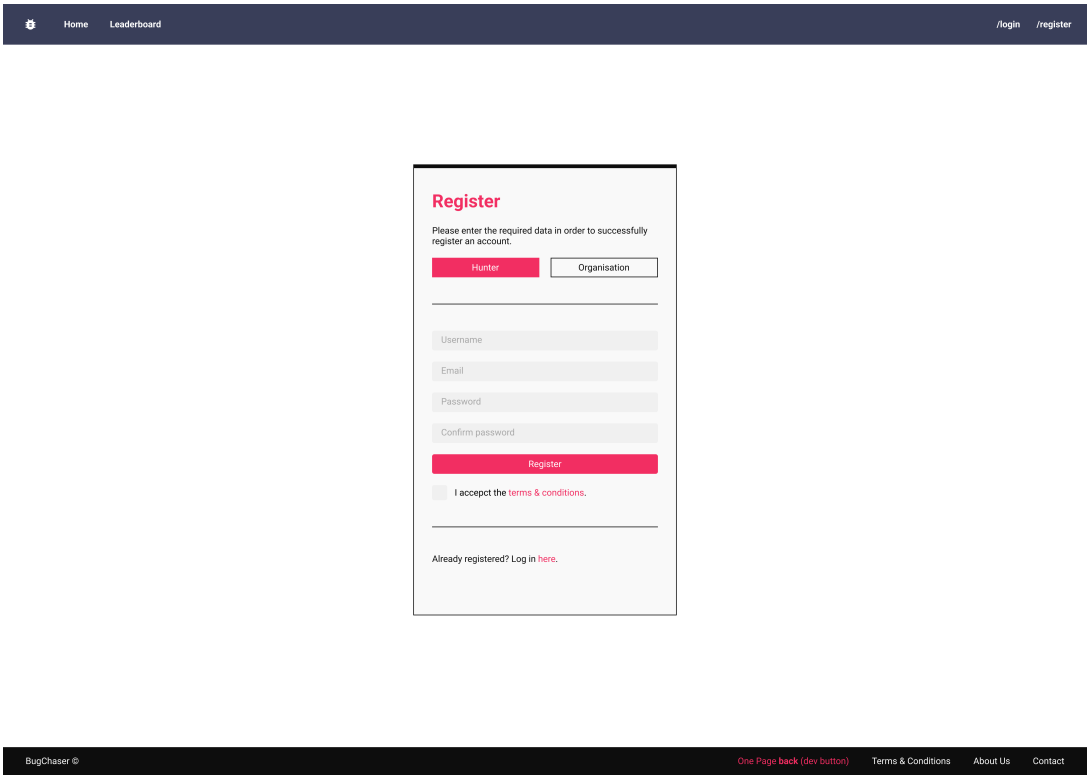


Abbildung C.5.: Registrierung Hunter

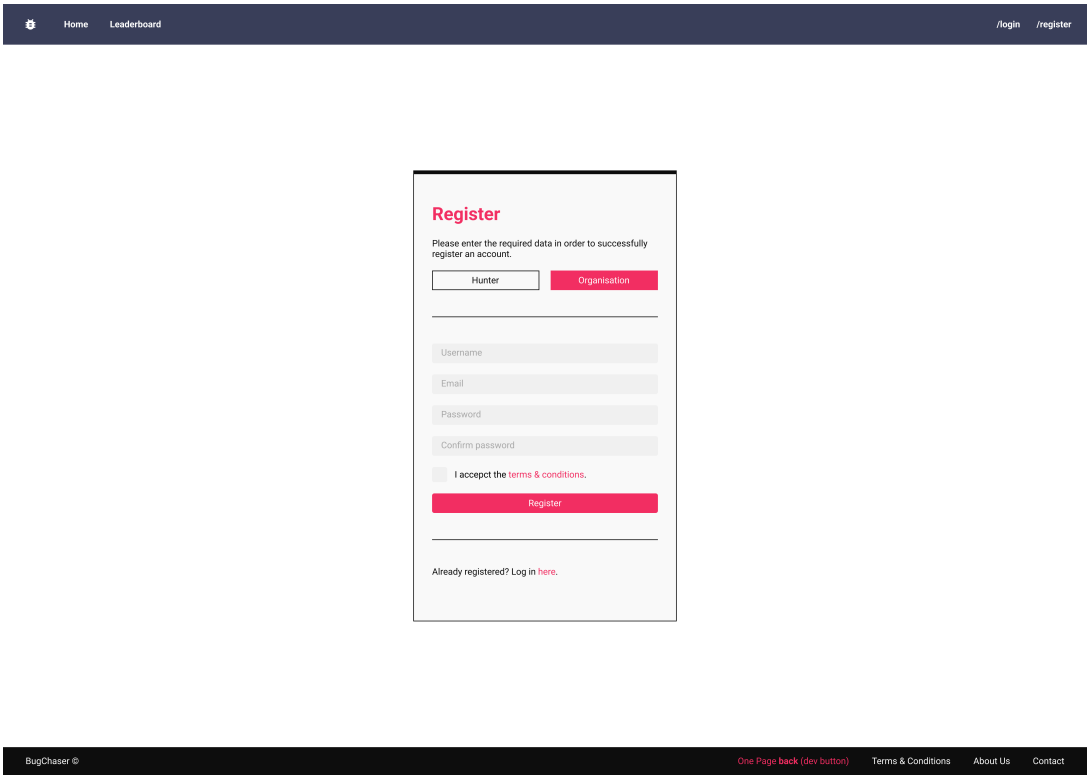


Abbildung C.6.: Registrierung Orguser

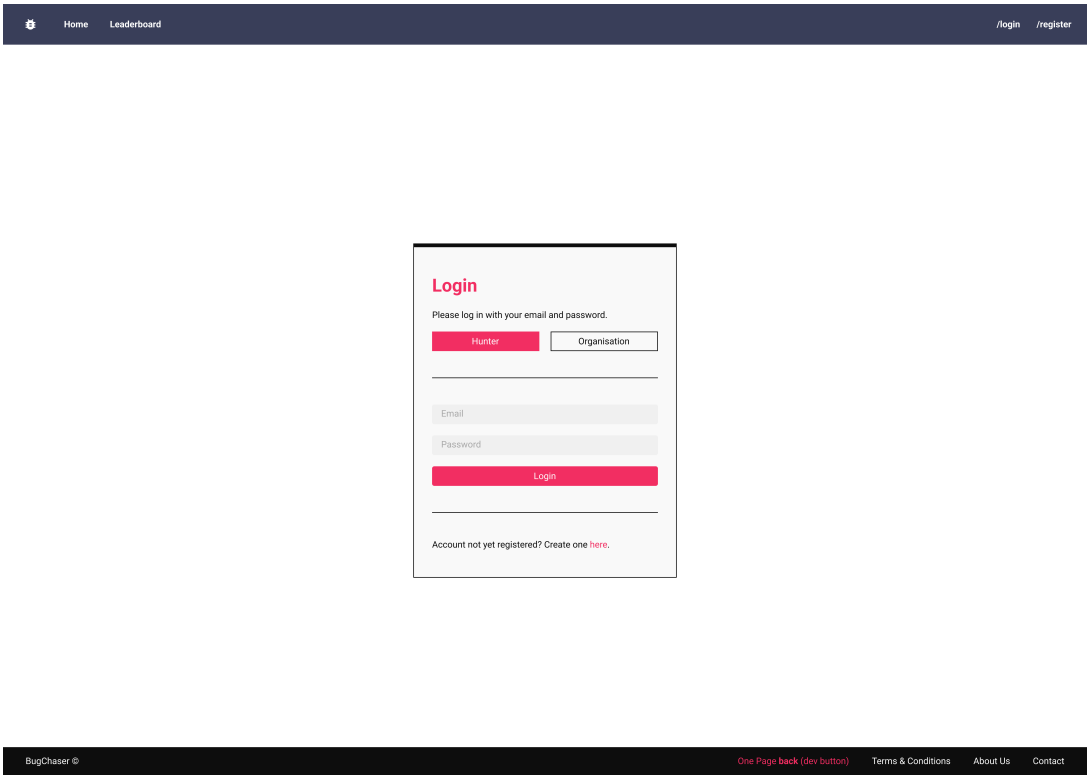


Abbildung C.7.: Login Hunter

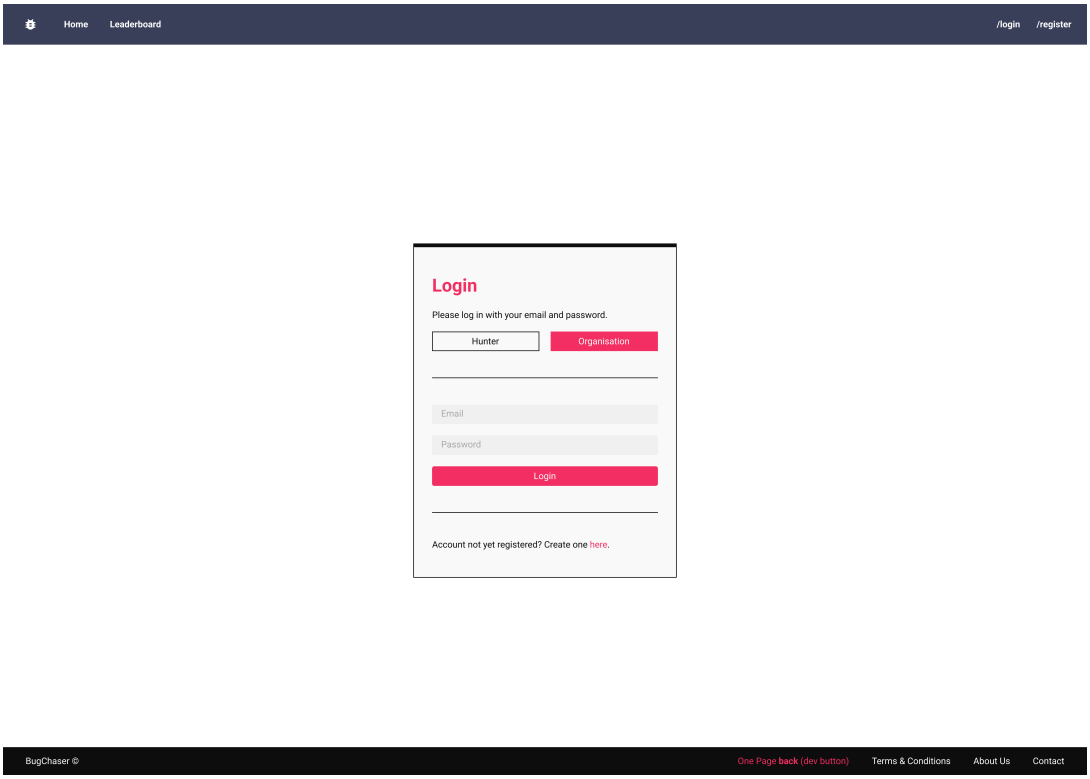


Abbildung C.8.: Login Orguser

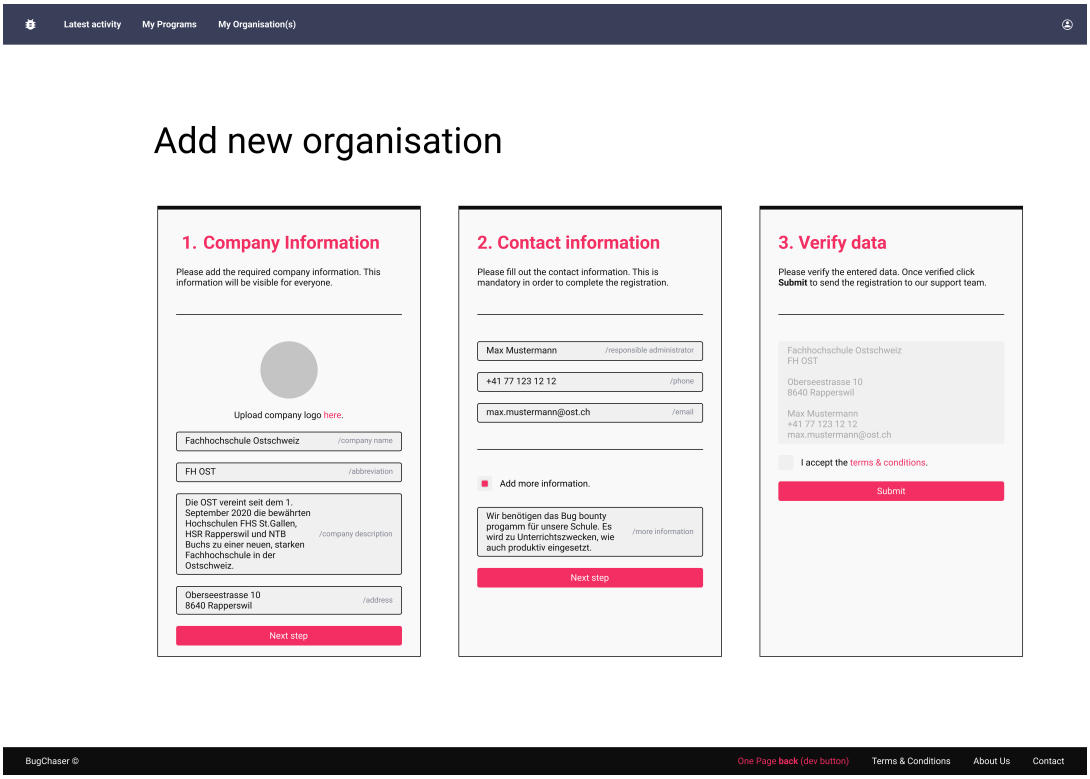


Abbildung C.9.: Neue Organisation hinzufügen

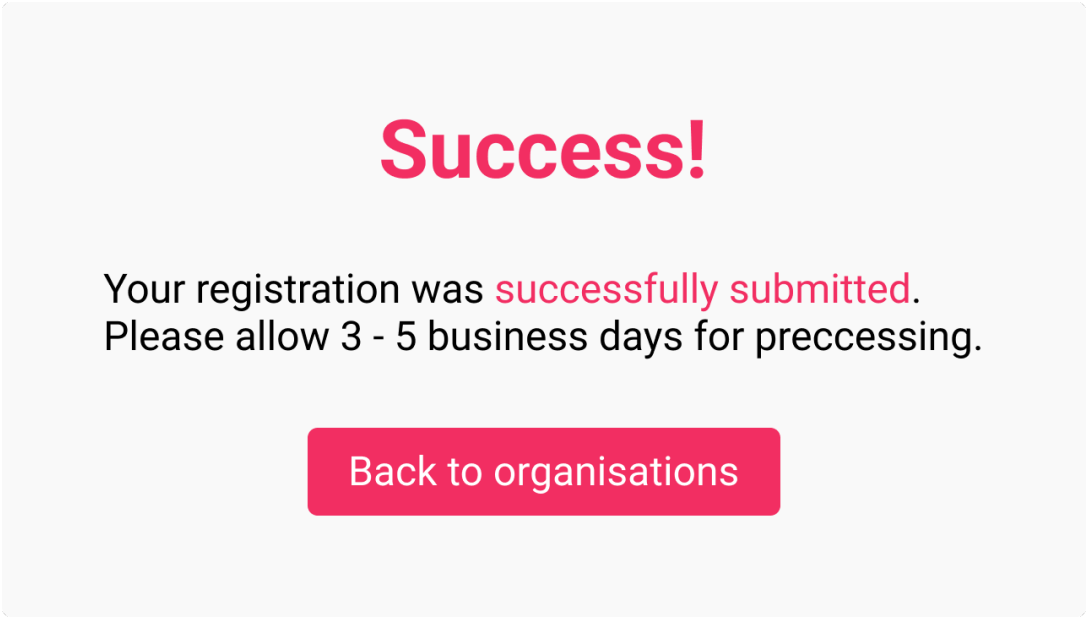


Abbildung C.10.: Organisation erfolgreich erstellt

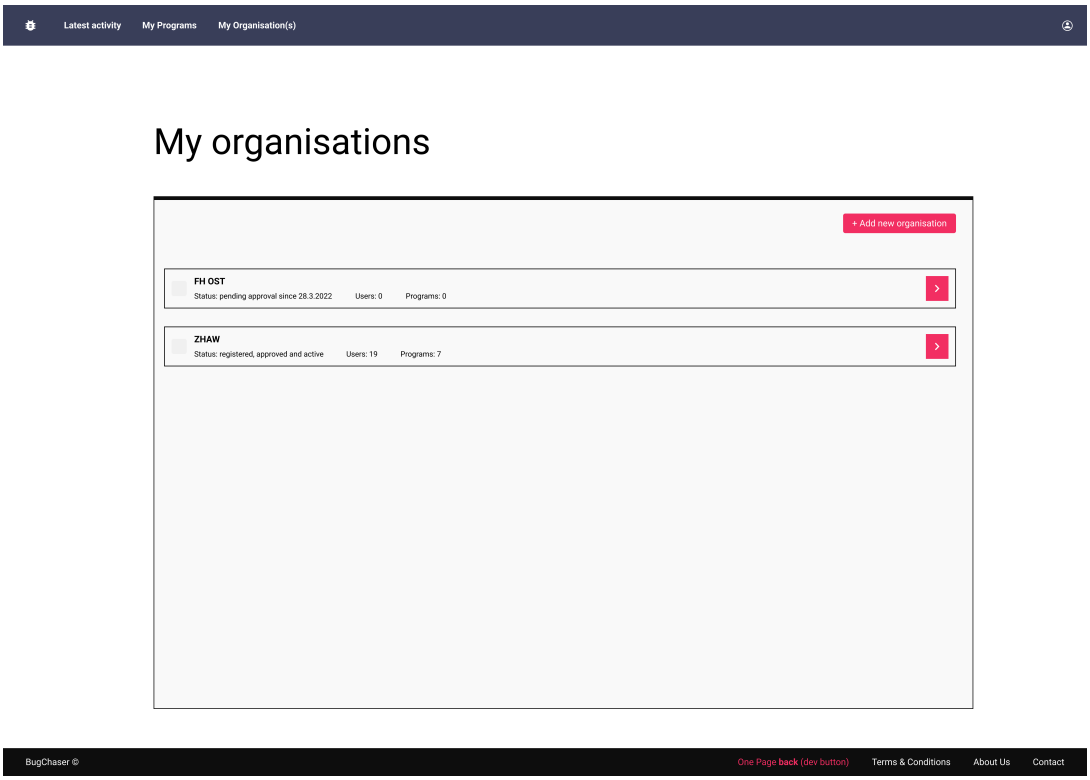


Abbildung C.11.: Meine Organisationen (Orguser Sicht)

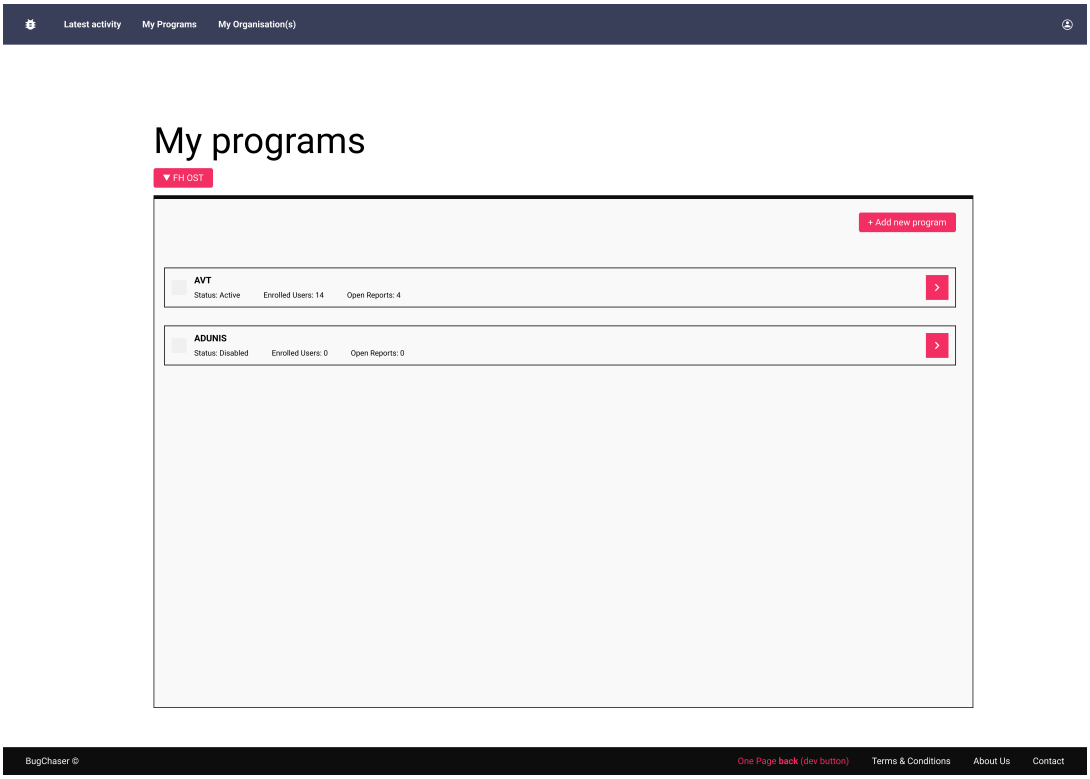


Abbildung C.12.: Meine Programme (Orguser Sicht)

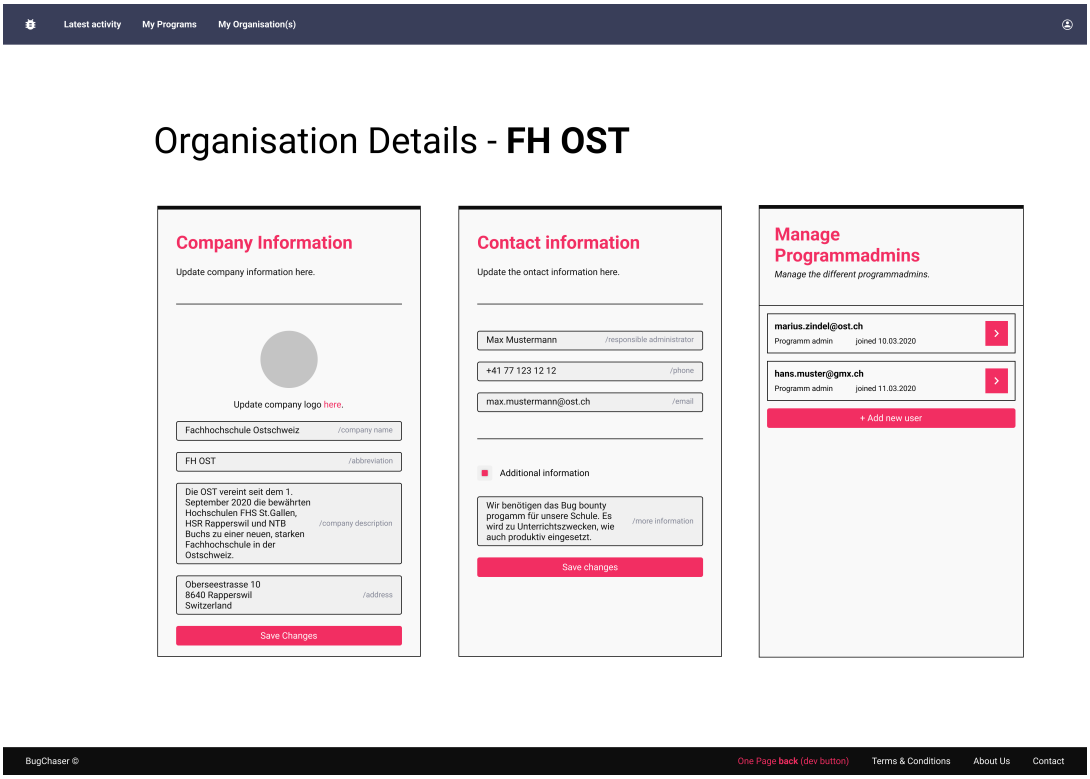


Abbildung C.13.: Details Organisation (Orguser Sicht)

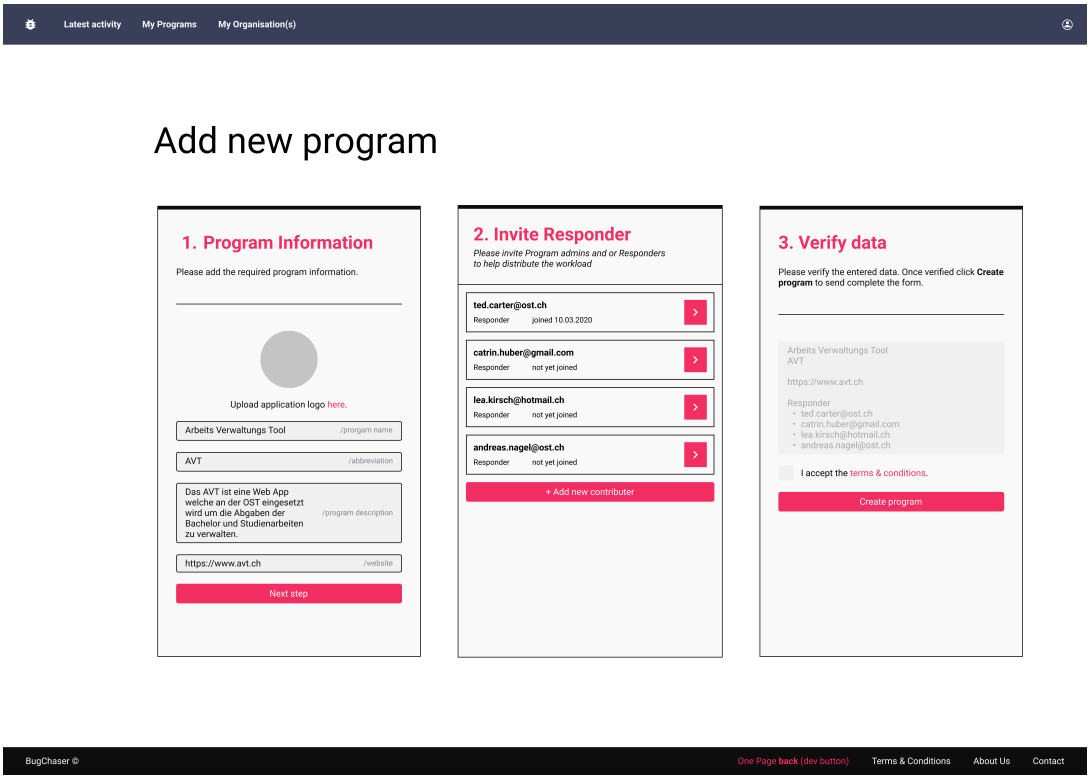


Abbildung C.14.: Neues Programm erstellen

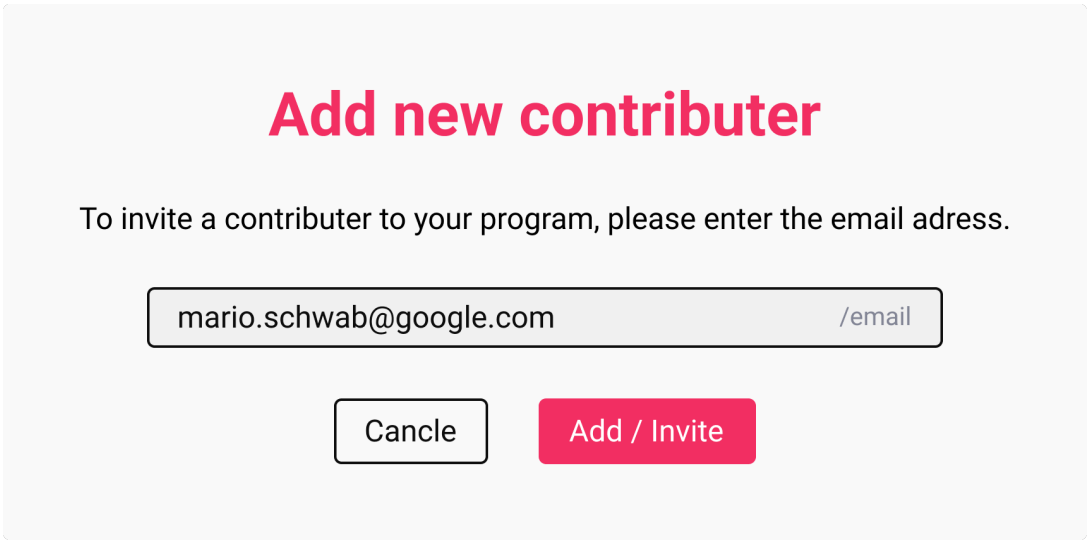


Abbildung C.15.: Neuer Benutzer zum Programm hinzufügen

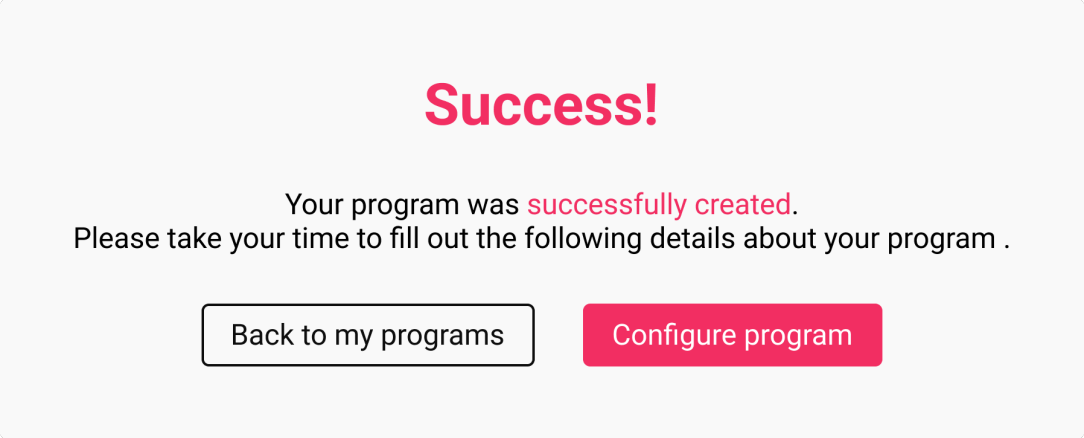


Abbildung C.16.: Programm erfolgreich erstellt

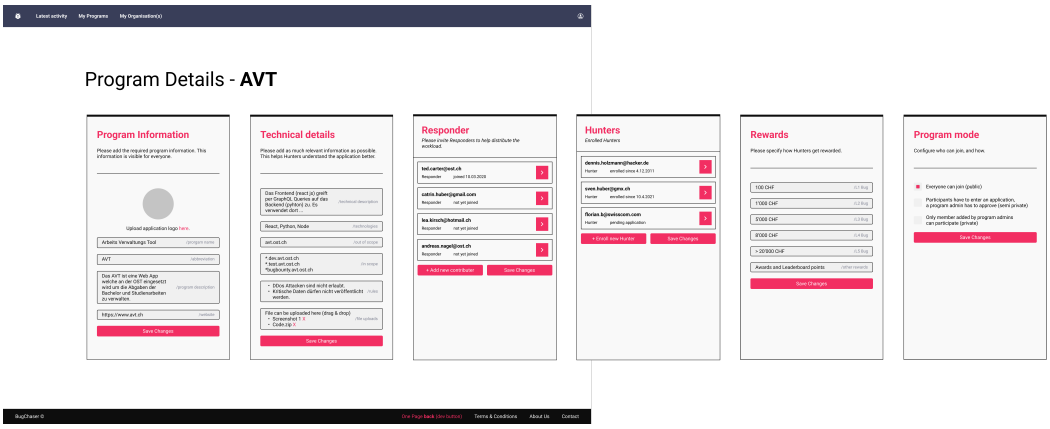


Abbildung C.17.: Program Details (Orguser Sicht)

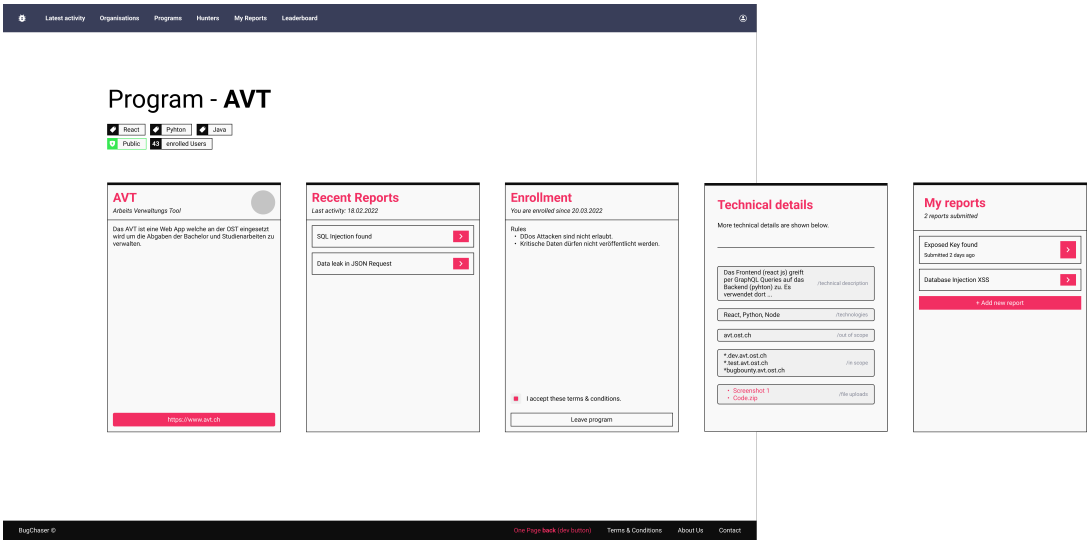


Abbildung C.18.: Beispiel Programm

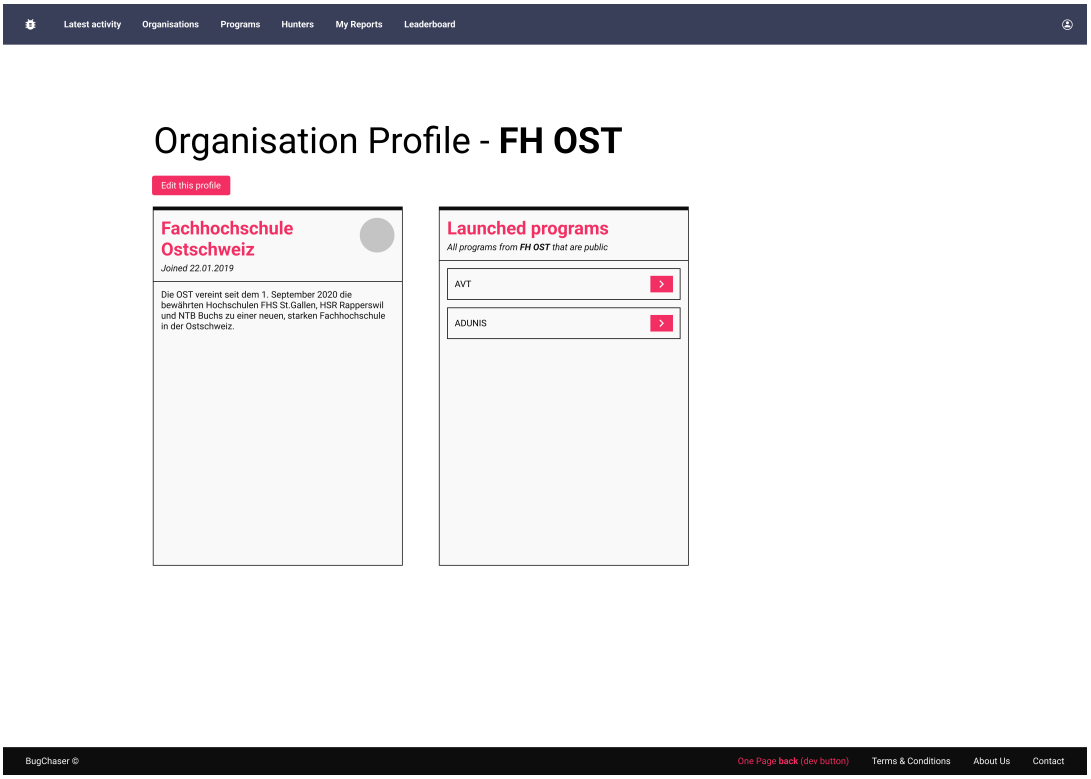


Abbildung C.19.: Beispiel Organisation

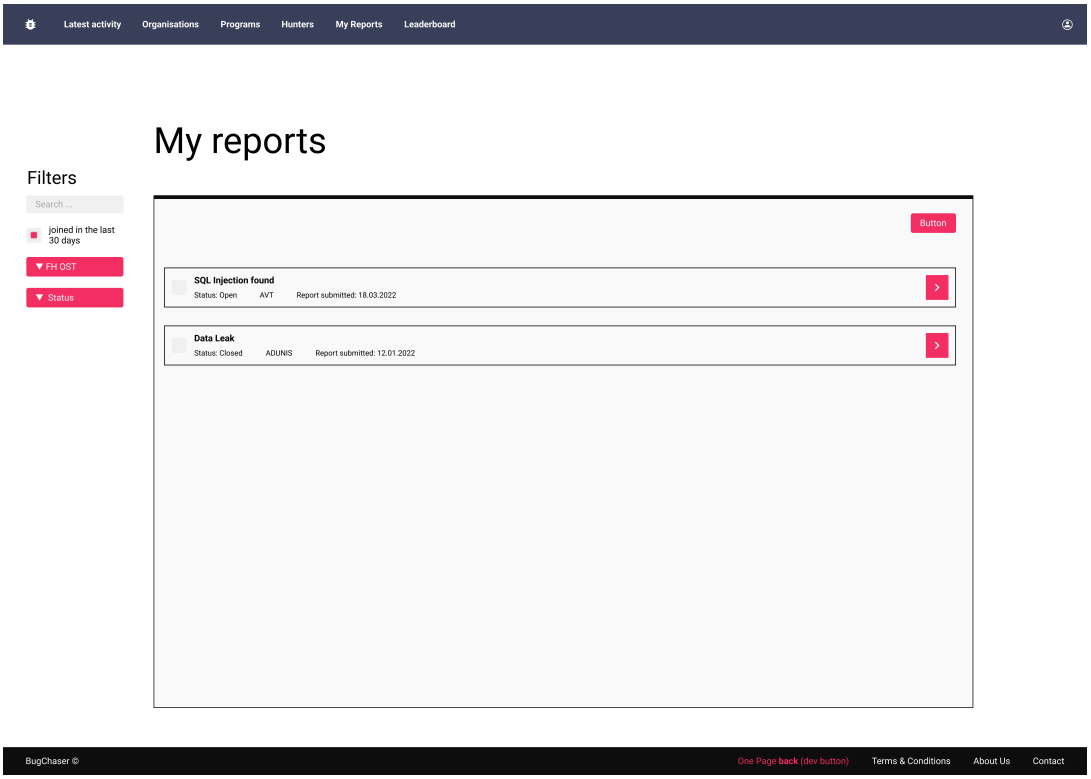


Abbildung C.20.: Meine Bug Reports (Hunter Sicht)

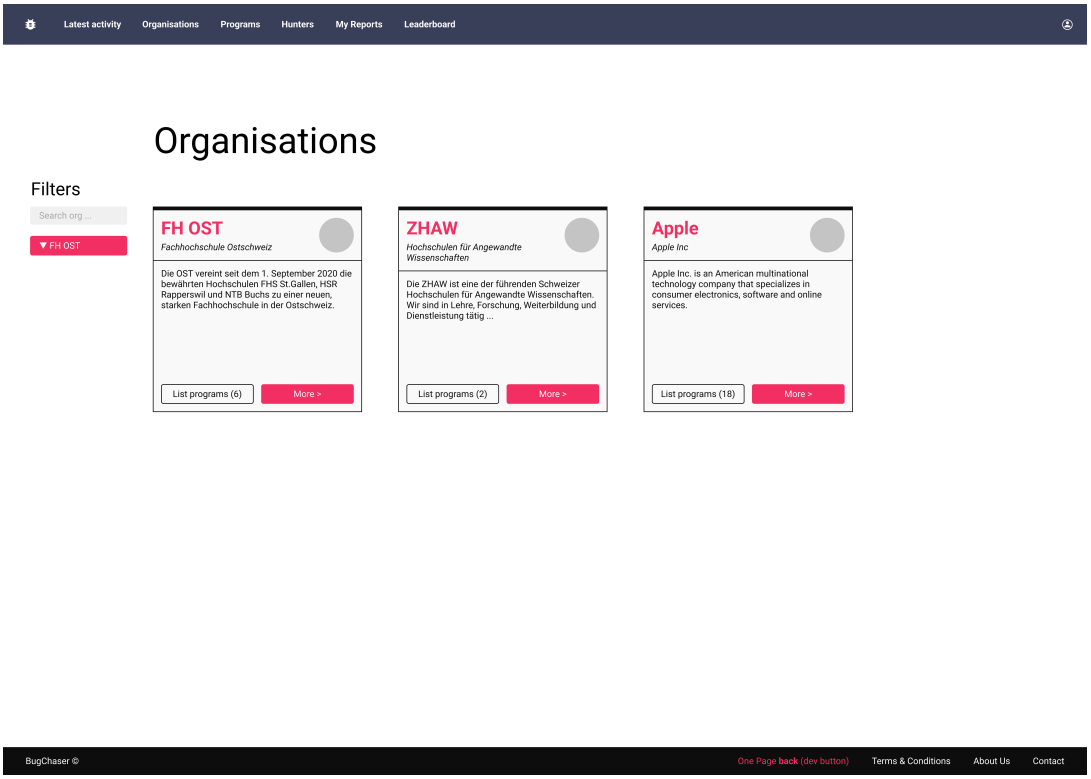


Abbildung C.21.: Liste von Organisationen (Hunter)

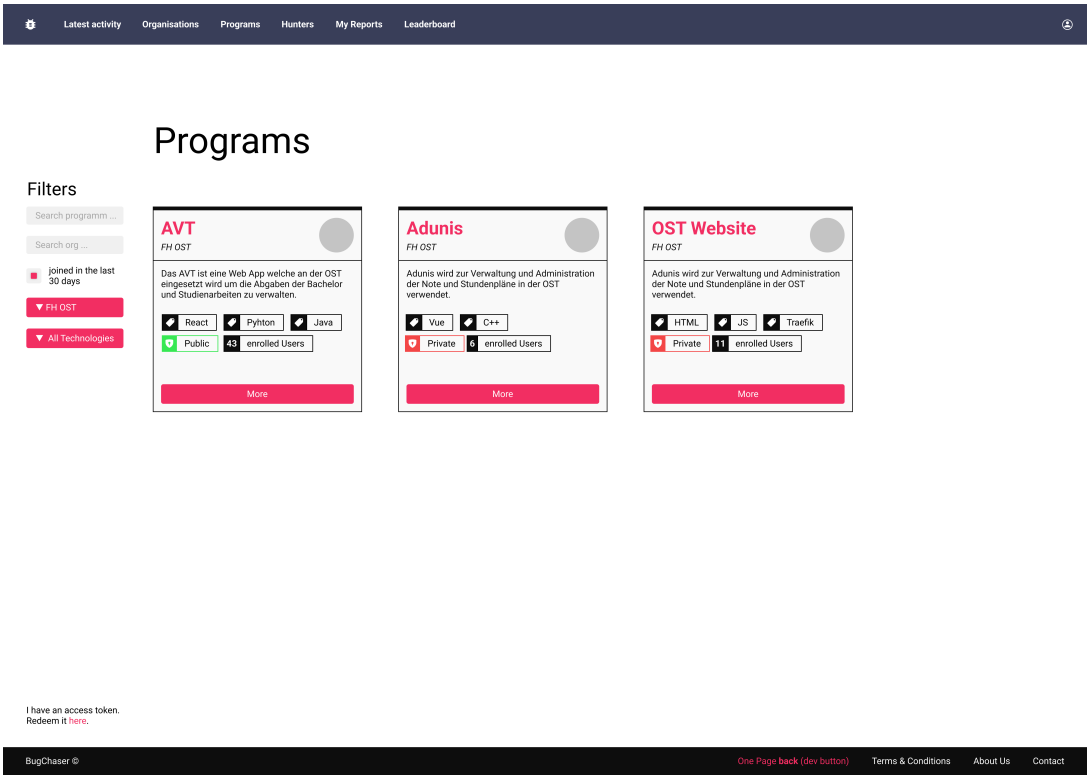


Abbildung C.22.: Liste von Programmen (Hunter)

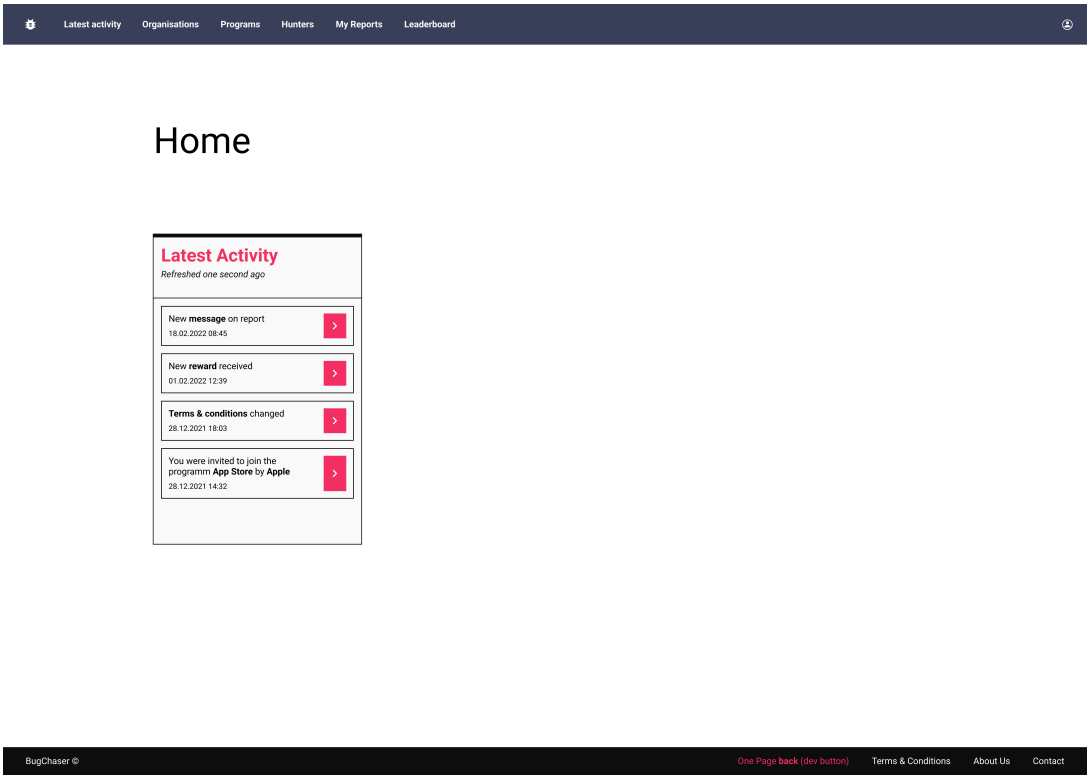


Abbildung C.23.: Home Seite Hunter

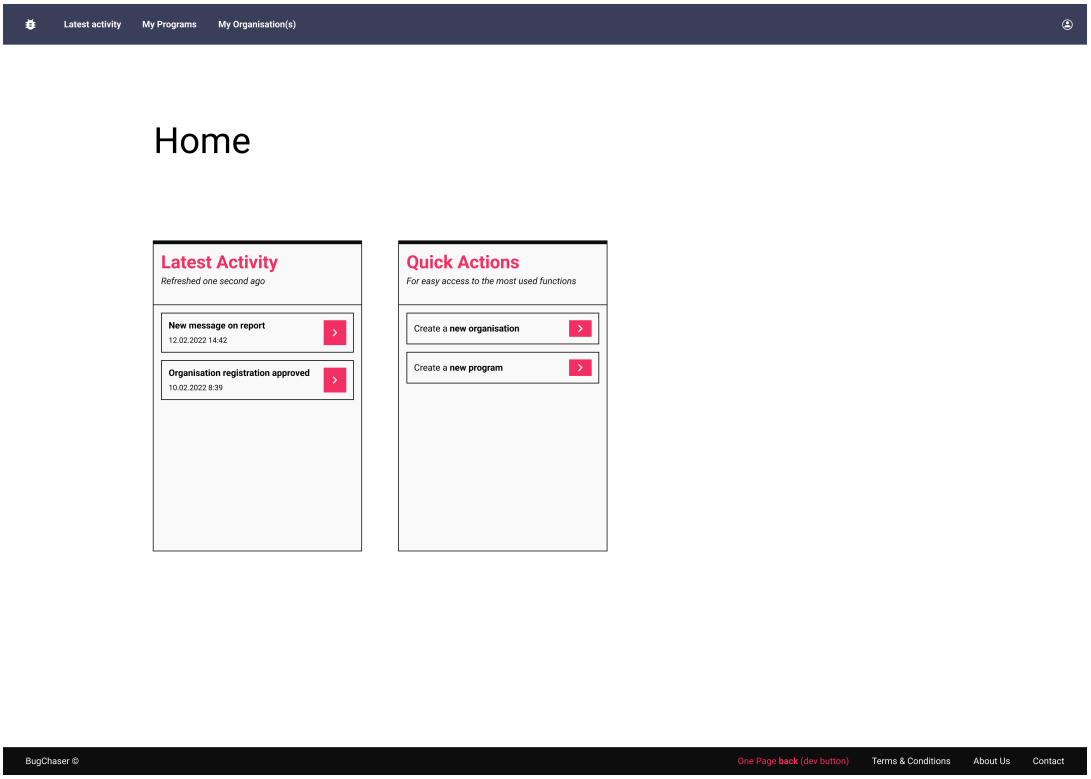


Abbildung C.24.: Home Seite Orguser

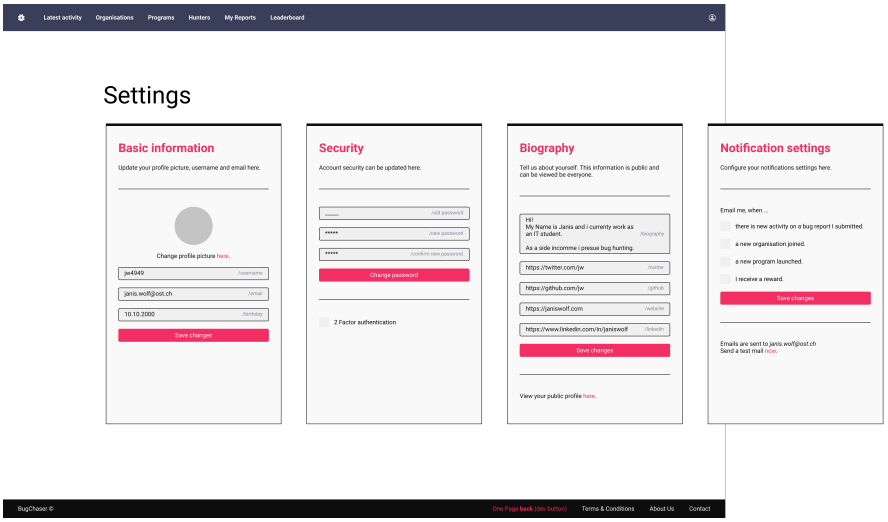


Abbildung C.25.: Einstellungen Hunter

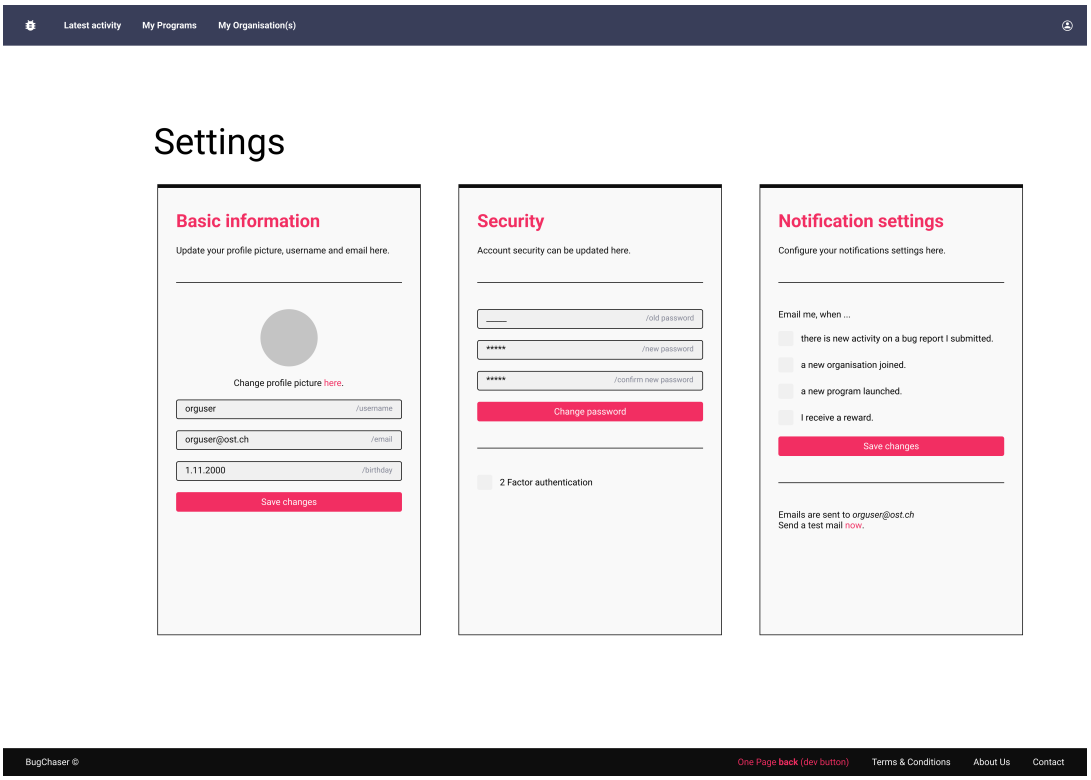


Abbildung C.26.: Einstellungen Orguser

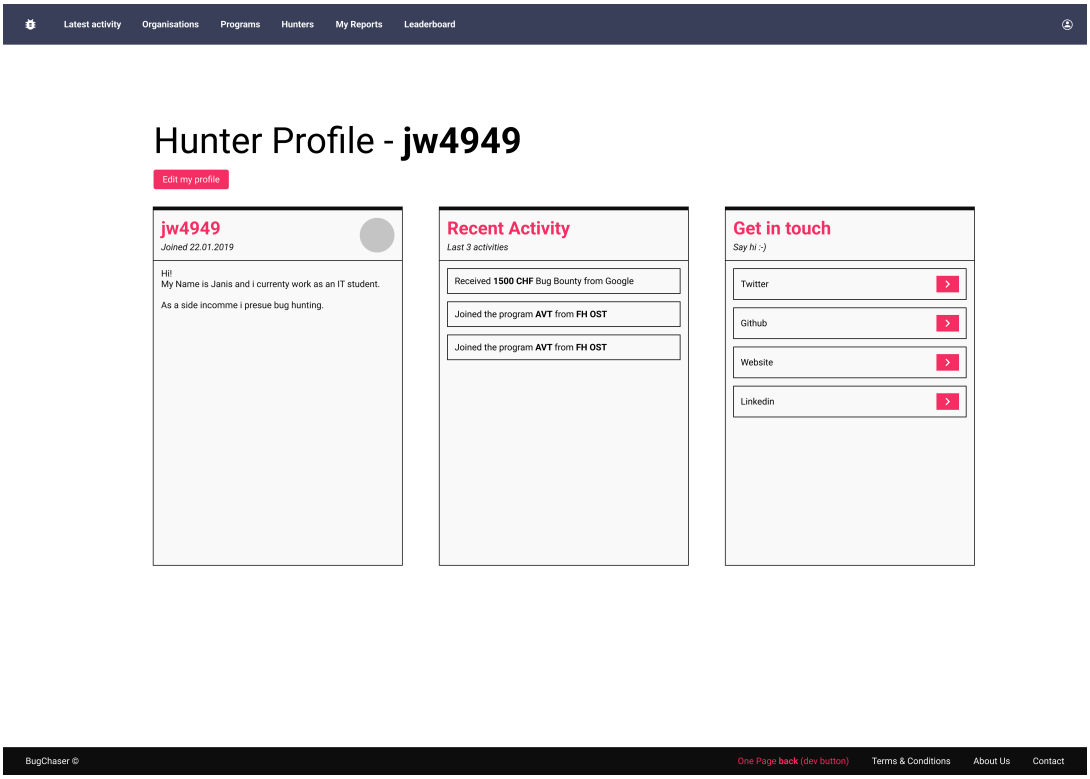


Abbildung C.27.: Öffentliches Profil Hunter

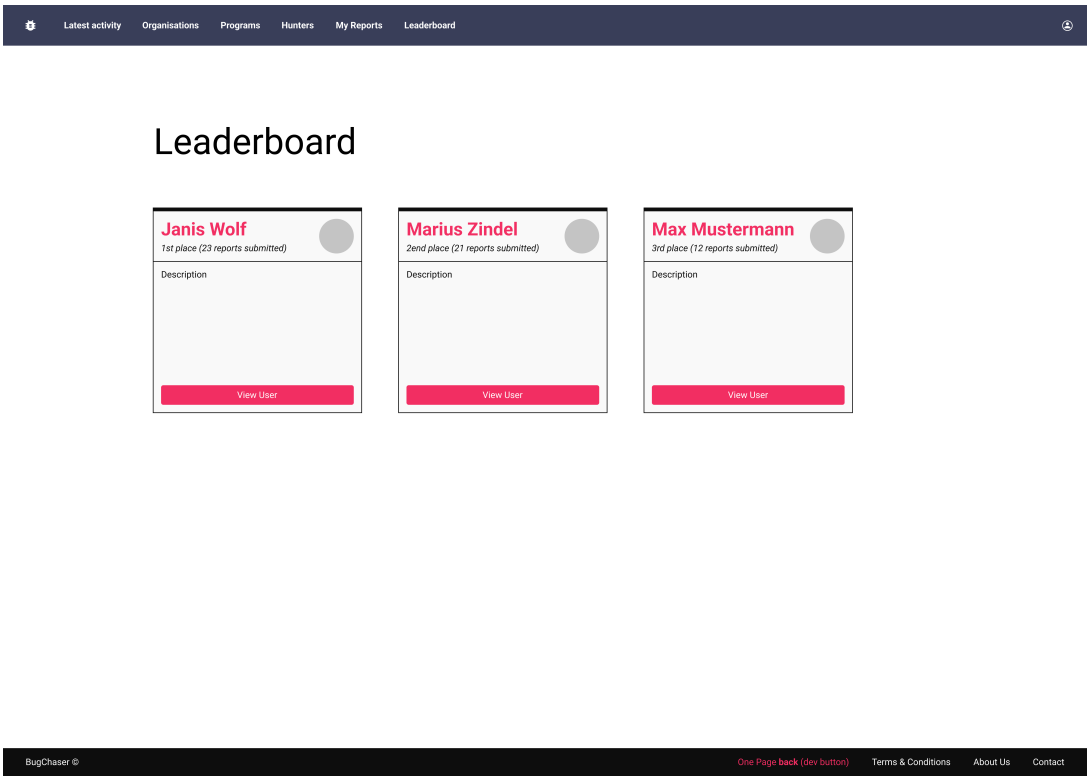


Abbildung C.28.: Leaderboard (Hall of Fame)

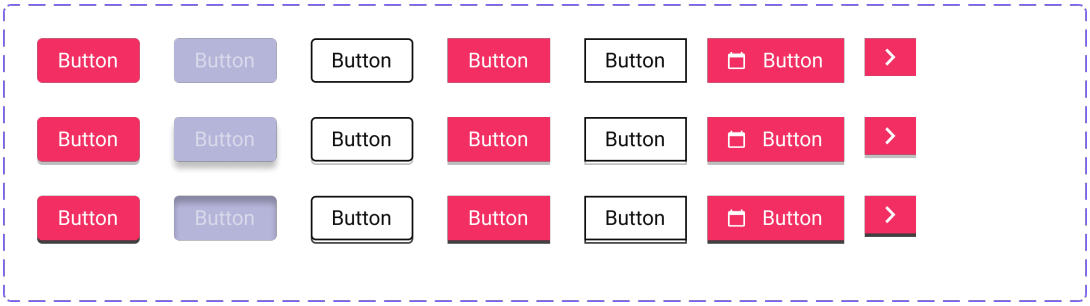


Abbildung C.29.: Button Komponente

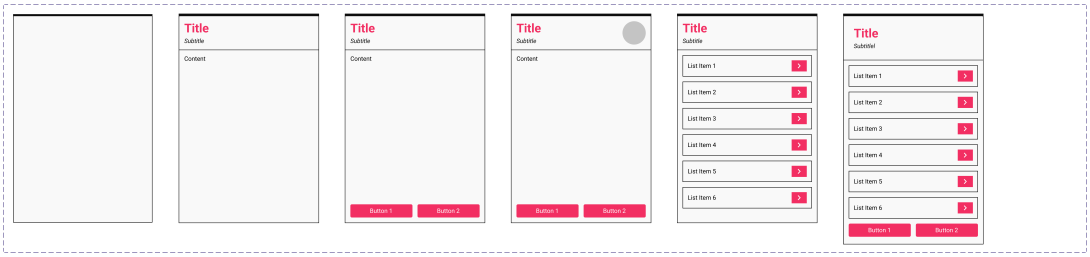


Abbildung C.30.: Karte Komponente

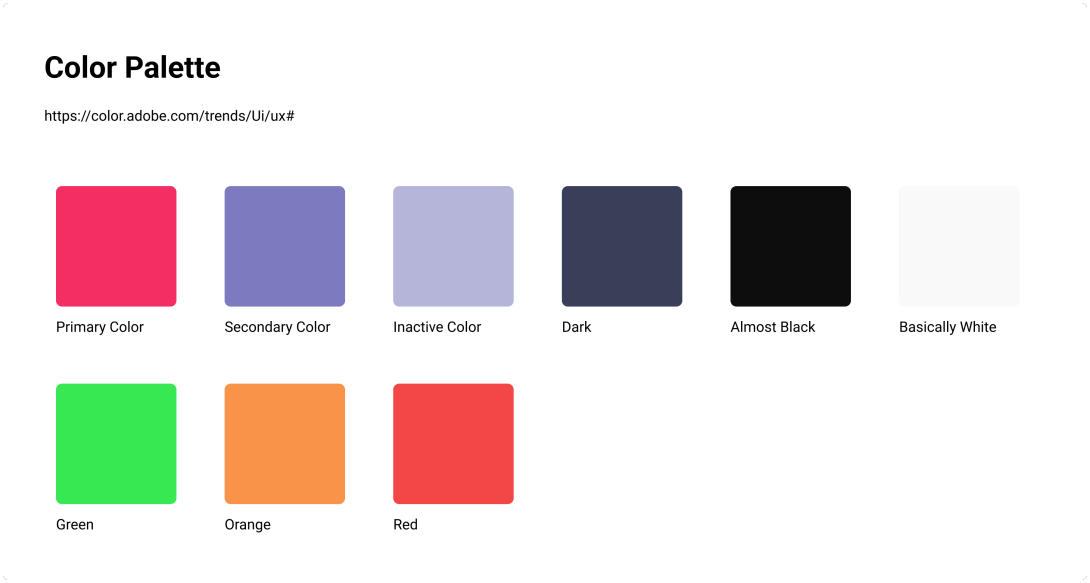


Abbildung C.31.: Farbschema Komponente

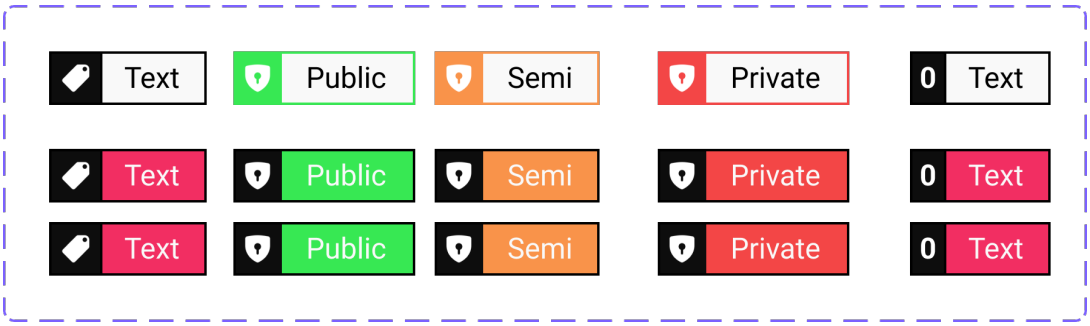


Abbildung C.32.: Tag Komponente

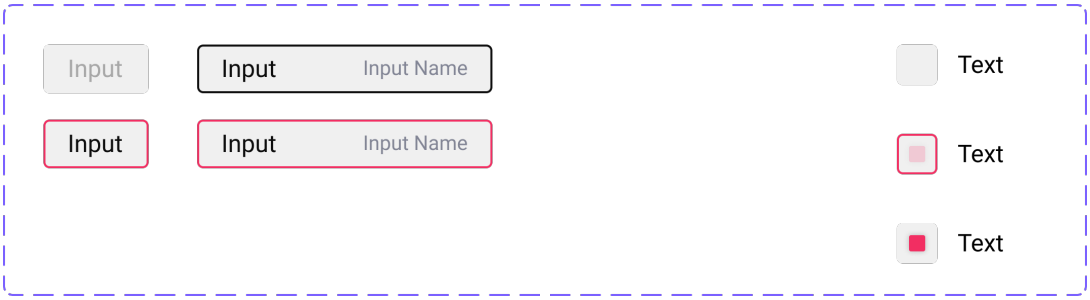


Abbildung C.33.: Input Komponente

Abbildung C.34.: Listen Element Komponente