

Projektbericht
Studiengang : Informatik

Schnitzeljagt Spiel

von

Felix Kurz, Felix Biedenbacher

86041, 67890

Betreuender Mitarbeiter : Dr. Marc Hermann

Einreichungsdatum : 28. Februar 2025

Eidesstattliche Erklärung

Hiermit erkläre ich, **Felix Kurz**, dass ich die vorliegenden Angaben in dieser Arbeit wahrheitsgetreu und selbständig verfasst habe.

Weiterhin versichere ich, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, dass alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ort, Datum

Unterschrift (Student)

Eidesstattliche Erklärung

Hiermit erkläre ich, **Felix Biedenbacher**, dass ich die vorliegenden Angaben in dieser Arbeit wahrheitsgetreu und selbständig verfasst habe.

Weiterhin versichere ich, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, dass alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ort, Datum

Unterschrift (Student)

Kurzfassung

Die kommende Dokumentation zum Scavenger-Hunt-Spiel gibt einen weitläufigen Einblick in ein Webbasiertes Schnitzeljagdspiel. Bei dem Projekt handelt es sich um ein Erweiterungsprojekt des schon vorhandenen Scavenger-Hunt-Spiels. Bei der Verbesserung wurde vor allem auf die Folgenden Punkte Wert gelegt.

- Verbesserung der User Experience
- Einfache und logische Bedienbarkeit der UI's
- und erweiterung der möglichkeiten eine Schnitzeljagt zu gestalten

Zur Verbesserung und Vereinfachung der Nutzung des Projekts Scavenger-Hunt wurde außerdem eine kleine GUI erstellt, auf welche im Verlauf der Dokumentation weiter eingegangen wird. Da einige Features dazugekommen sind und auch einiges überarbeitet werden musste, gab das Projekt einen umfassenden Einblick in alle verwendeten Tools sowie erste Einblicke in die REST-APIs.

Die Dokumentation gibt Einblicke in verwendete Technologien, Tools, Bibliotheken und Frameworks sowie Anwendungsbeispiele und die Überarbeitung des Projekts, beginnend mit einer Übersicht über das vorhandene Projekt und dessen Grundlagen, gefolgt von den geplanten Erweiterungen und deren Umsetzungen, mit abschließendem Fazit und Ausblick in die Zukunft zum Scavenger-Hunt-Spiel.

Inhaltsverzeichnis

Eidesstattliche Erklärung	i
Eidesstattliche Erklärung	ii
Kurzfassung	iii
Inhaltsverzeichnis	iv
Abbildungsverzeichnis	vii
Abkürzungsverzeichnis	x
1. Einleitung	1
1.1. Motivation	1
1.2. Ziel der Arbeit	2
1.3. Vorgehen	2
1.3.1. Einarbeitungsphase	2
1.3.2. Entwurfsphase	2
1.3.3. Implementierungsphase	3
2. Grundlagen	4
2.1. C#	4
2.2. WPF	5
2.3. Docker	5

2.4. Svelte	6
2.5. Node	7
2.6. Entity Framework Core	8
2.7. InstallForge	10
3. Problemanalyse	11
3.1. Bestandsaufnahme	11
3.1.1. Anforderungen	11
4. Erweiterungsentwurf	14
4.1. Blockansicht	15
4.2. Hunt-GUI	16
4.2.1. Hunt-Editor	18
4.3. Hunt-Web-Game	20
4.4. Spielablauf	23
5. Implementierung	27
5.0.1. Aufbau der Hunt-GUI	27
5.0.2. Aufbau des Hunt-Editor	32
5.0.3. Aufbau des Hunt-Web-Games	36
5.0.4. Hunt-Be-API	44
6. Inbetriebnahme	45
6.0.1. InstallForge	47
6.0.2. Systemtest	49
7. Zusammenfassung und Ausblick	50
7.1. Erreichte Ergebnisse	50
7.2. Ausblick	52

7.3. Eigene Meinung und Learnings	52
Literatur	54
A. Architektonische Entscheidungen	55
A.1. Architektur Hunt-GUI	55
A.2. Architektur Hunt-Web-Game	56
B. Nutzung von Generativer KI-Systeme	57
C. Anhang A	58

Abbildungsverzeichnis

3.1. Ziele der Projektarbeit	12
4.1. Kontext Diagramm der Anwendung	15
4.2. Dark Mode der GUI	17
4.3. White Mode der GUI	18
4.4. Wireframe zur Share Funktion	19
4.5. Hunt Web Game Wireframe vom Login Prozess	21
4.6. Hunt Web Game Wireframe vom Login Prozess	22
4.7. Aktivitätsdiagramm des Registrier Prozesses	23
4.8. Aktivitätsdiagramm des Spiels	25
5.1. Relationsdigramm GUI	28
5.2. Klassendigramm GUI	29
5.3. Fertige GUI (Black)	30
5.4. Starten des Editors	32
5.5. Hinzufügen eines Assignments mit Hint-Typ Video	33
5.6. Übersicht einer erstellten Hunt mit allen Assignments im Überblick .	33
5.7. Erfolgreiches Erstellen einer Hunt	34
5.8. Teilen des Links oder QR-Codes einer Schnitzeljagd	35
5.9. Home-Seite der Hunt-Web-Game-Anwendung	36
5.10. Screen zur Registrierung einer Hunt	37
5.11. Registrier-Pop-up und Eingabe eines unsicheren Passworts	37

5.12.Registrierung zu einer Hunt mit schon verwendetem Nutzernamen . .	38
5.13.Erfolgreiches Anmelden zu einer Schnitzeljagd	38
5.14.Login Screen	39
5.15.Übersicht aller vom Nutzer registrierten Hunts	39
5.16.Ansicht des Video-Hint-Typs mit zusätzlichen Informationen	40
5.17.Ansicht des Bild Hint Typ ohne zusätzlichen Informationen	41
5.18.Solution-Sektion mit richtiger Eingabe	42
5.19.Cheer-Screen	43
5.20.Completed Hunt	43
6.1. Web-Game auf Ansicht auf dem Handy	46
7.1. Trello Board	51
C.1. Hunt Web Game Wireframe vom Login Prozess	59
C.2. CMD Konsolen für docker-compose und Editor	60

Abkürzungsverzeichnis

EFC Entity Framework Core	8
ORM Object-Relational Mapping	8
UI User Interface	16

1. Einleitung

1.1. Motivation

Das Studium ist für die neuen Studierenden ein neuer Lebensabschnitt. Neue Orte, neue Begegnungen, neue Herausforderungen und neue Menschen. All das kracht auf die Studenten ein und überwältigt diese. Man bekommt so viel Information und Wissen in den ersten Wochen mit, und wenn man diese Info dann benötigt, kann man sich nicht mehr erinnern.

„Beim Spiel kann man einen Menschen in einer Stunde besser kennenlernen als im Gespräch in einem Jahr“ [7].

Schon die alten Griechen haben erkannt, dass der Mensch spielend Dinge einfacher lernt und aufnimmt. Gerade in einer Phase, die voll mit Informationen und Neuem ist, kann ein Spiel als Anker dienen, um alles aufzulockern.

Normalerweise gibt es Touren durch die Hochschule, in denen den neuen Studenten wichtige Orte vorgestellt werden. Aber bei diesen Touren ist wenig Interaktion vorhanden, man lässt sich als Student berieseln, am Ende bleibt meist nur wenig hängen. Laut der Lernpyramide bleibt beim reinen Zuhören meist nur 10–20% im Gedächtnis, durch ein interaktives Spiel wiederum können bis zu 75–90% im Gedächtnis bleiben.

Bei der Scavenger-Hunt geht es darum, Orte in der Hochschule spielend kennenzulernen. Durch Rätsel und Forschung neue Orte mit den Kommilitonen kennenlernen. Wenn dann im weiteren Verlauf des Studiums mal ein Ort gesucht wird, erinnert man sich vielleicht zurück an den Ort, welchen man mit seinen Kommilitonen durch das Lösen eines Rätsels gefunden hat.

1.2. Ziel der Arbeit

Das Ziel der Arbeit war es, das schon vorhandene Scavenger-Hunt-Spiel zu erweitern und zu Verbessern. Bei einer Schnitzeljagd müssen Spieler Rätsel, sogenannte Hints, lösen und bekommen dann bei richtigem Beantworten die nächste Aufgabe. Ziel war es, ein leicht bedienbares und benutzerfreundliches Spiel zu erstellen, mit welchem neue Studenten den Campus kennenlernen können. Es war aber auch ein großes Augenmerk darauf, die Erstellung einer Schnitzeljagd zu vereinfachen.

1.3. Vorgehen

Um den vollständigen Umfang dieser Erweiterung zu überblicken und zu verstehen, war es essenziell, die Dokumentation des Vorgängerprojekts sorgfältig zu analysieren. Diese Dokumentation lieferte wertvolle Einblicke in die bestehende Systemarchitektur, die getroffenen Designentscheidungen sowie bereits implementierte Funktionalitäten.

Basierend auf den gewonnenen Erkenntnissen aus der Analyse der Vorgängerdokumentation ergaben sich die folgenden Phasen des Entwicklungsprozesses:

1.3.1. Einarbeitungsphase

In dieser Phase wurde sich in vorhandenen Code, Architekturen, Tools und Frameworks eingearbeitet. Das vorhandene Spiel wurde ausführlich getestet und alle Ungereimtheiten wurden notiert und auf einem Kanban-Board Aufgaben zugewiesen. Außerdem wurden alle kommenden Features als Aufgaben definiert.

1.3.2. Entwurfsphase

Nach der Einarbeitungszeit wurden gefundene Probleme und neue Features auf dem Kanban-Board in verschiedene Kategorien der Dringlichkeit unterteilt. Um Features visuell mit Stakeholdern abklären zu können, wurden Wireframes und Prototypen der Features erstellt. Da es sich um eine Erweiterung handelt wurden nur da UML Diagramme erstellt wo diese benötigt werden.

1.3.3. Implementierungsphase

Die Erweiterungen sowie die Verbesserungen wurden anhand der zugewiesenen Dringlichkeit bearbeitet. Features, welche eine systemkritische Funktionalität darstellten, wurden deshalb als erstes bearbeitet. Die verschiedenen Features und Verbesserungen wurden so zerlegt, dass sich viele kleine Arbeiten ergaben. Diese wurden dann nach und nach implementiert und getestet. Auf diese Weise konnte sichergestellt werden, dass durch Änderungen keine bestehenden Funktionalitäten des Programms verloren gehen.

2. Grundlagen

Im folgenden Kapitel werden verwendete Frameworks und Tools beschrieben. Da das Projekt eine Erweiterung eines schon funktionstüchtigen Projektes war, wird im kommenden Kapitel nicht mehr auf die Auswahl der Systemarchitektur und des Softwaredesigns eingegangen. Grundlegende Bausteine des Projekts werden aber in kommenden Abschnitten trotzdem nochmal ausgeführt und erklärt.

2.1. C#

C# [6] ist eine moderne, leistungsstarke Programmiersprache, die speziell für die Entwicklung von Anwendungen im objektorientierten Stil konzipiert wurde. Es zeichnet sich durch seine klare Syntax, hohe Effizienz und große Anwendbarkeit aus, was es ideal für eine Reihe von Anwendungsdomänen macht. Ob lokale Applikationen, Webanwendungen, Spiele oder Cloud-Services: C# bietet eine haltbare Grundlage für professionell zu entwickelnde Softwareprodukte. Die Sprache wurde von Microsoft mit dem Ziel entwickelt, ein zentraler Bestandteil der .NET-Welt zu werden und es ermöglicht mit der engeren Integration des .NET-Frameworks eine einfache und kosteneffiziente Entwicklung von verteilten, gesicherten und plattformübergreifenden Anwendungen. Es speist sich aus einem Katalog hervorragender Bibliotheken sowie Tools und untermauert somit die Bemühungen der Entwickler, sich in die Lage zu versetzen, Softwareprodukte von heute mit Spitzenleistungen und einer robusten Lebenserwartung zu entwickeln. C# spielte in diesem Projekt eine tragende Rolle, als es sowohl für die grafische Benutzeroberfläche der Webanwendung als auch für interne Geschäftslogik in hohem Maße eingesetzt wurde. Dank C# gelang es, eine zügige Geschäftslogik, eine angenehme Benutzerschnittstelle und eine verlässliche und somit dauerhafte Architektur zu verwirklichen. Die objektorientierten Prinzipien durch C# wurden benutzt, um eine saubere und modulare Codebasis zu etablieren, durch die die zukünftige Wartung und Weiterentwicklung des Systems merklich einfach gemacht wurde.

2.2. WPF

WPF (Windows Presentation Foundation) [5] ist ein von Microsoft entwickeltes UI-Framework zur Erstellung moderner und leistungsfähiger Desktop-Anwendungen auf Basis von C#. Es bietet eine flexible Architektur, die eine klare Trennung von Benutzeroberfläche und Geschäftslogik ermöglicht, indem es XAML (Extensible Application Markup Language) für die UI-Definition nutzt. Dadurch können Entwickler und Designer unabhängig voneinander arbeiten, was die Wartbarkeit und Erweiterbarkeit der Anwendungen verbessert.

Ein wesentliches Merkmal von WPF ist das umfassende Datenbindungskonzept, das eine effiziente Synchronisation zwischen der Benutzeroberfläche und den zugrunde liegenden Datenmodellen erlaubt. Zudem bietet WPF eine leistungsstarke Unterstützung für Stile und Vorlagen, wodurch das Erscheinungsbild von Steuerelementen konsistent und flexibel angepasst werden kann. Durch die Verwendung von Ressourcen und dynamischen Layouts lassen sich skalierbare und adaptive UI-Designs erstellen.

Neben den klassischen UI-Elementen verfügt WPF über ein fortschrittliches Grafik-Rendering-Modell, das auf DirectX basiert. Dies ermöglicht hardwarebeschleunigte Darstellungen und die Implementierung anspruchsvoller visueller Effekte wie Transparenzen, Schatten und Animationen. Dadurch eignet sich WPF besonders für Anwendungen, die ein modernes und interaktives Design erfordern.

Ein weiterer Vorteil von WPF ist die Unterstützung für das Model-View-ViewModel (MVVM)-Pattern, das eine saubere Trennung zwischen UI, Logik und Daten gewährleistet. Dies erleichtert nicht nur das Testen, sondern auch die Wiederverwendbarkeit von Codekomponenten.

In diesem Projekt wurde WPF in Kombination mit C# genutzt, um eine effiziente, minimalistische und dennoch leistungsfähige grafische Benutzeroberfläche zu realisieren. Durch die Verwendung von Datenbindung und flexiblen Layouts konnte eine dynamische UI geschaffen werden, die sich an verschiedene Anwendungsfälle anpasst und eine intuitive Benutzererfahrung bietet.

2.3. Docker

Docker [1] ist eine leistungsstarke Software zur Containerisierung von Anwendungen. Sie ermöglicht es, Anwendungen in sogenannten Containern auszuführen, wodurch eine portable, konsistente und ressourcenschonende Umgebung geschaffen wird. Dies ähnelt dem Einsatz virtueller Maschinen oder dedizierter Server, jedoch mit einem deutlich geringeren Overhead.

Das Herzstück von Docker ist das Konzept der Images und Container. Ein Image ist eine schreibgeschützte Vorlage, die alle notwendigen Abhängigkeiten, Bibliotheken und Konfigurationsdateien enthält, um eine Anwendung auszuführen. Mithilfe von Dockerfiles – speziellen Skripten zur Definition eines Images – kann aus einer Anwendung oder ihren Komponenten ein entsprechendes Image erstellt werden.

Auf Basis dieser Images können dann Container erzeugt werden. Ein Container ist eine ausführbare Instanz eines Images, die isoliert von anderen Containern und Anwendungen auf dem Host-System läuft. Diese Isolation stellt sicher, dass verschiedene Anwendungen mit unterschiedlichen Abhängigkeiten gleichzeitig auf demselben System betrieben werden können, ohne sich gegenseitig zu beeinflussen.

Ein wesentlicher Vorteil von Docker besteht in seiner Plattformunabhängigkeit: Ein einmal erstelltes Image kann auf verschiedenen Betriebssystemen und Infrastrukturen ausgeführt werden – sei es auf lokalen Rechnern, in Rechenzentren oder in der Cloud. Dies erleichtert sowohl die Entwicklung als auch den Betrieb von Anwendungen erheblich, da sie unabhängig von der zugrunde liegenden Hardwareumgebung bereitgestellt und skaliert werden können.

Durch die Nutzung von Container-Orchestrierungstools wie Kubernetes lassen sich zudem mehrere Container verwalten, skalieren und automatisch bereitstellen, was Docker zu einer essenziellen Technologie in der modernen Softwareentwicklung und DevOps macht.

2.4. Svelte

Svelte [9] ist ein JavaScript-Framework zum Erstellen von Web-Apps. Der geschriebene Code einer Webanwendung wird bei Svelte schon während des Build-Prozesses in optimierten JavaScript-Code umgewandelt. Dieser Prozess hat wiederum den Vorteil, dass der Nutzer der Web-App bei Laufzeit keine bis wenige Wartezeiten hat und sich dadurch ein flüssiges Erlebnis ergibt.

In Svelte wird in HTML, CSS, TypeScript und JavaScript programmiert. In Type und JavaScript werden die allgemeinen Seiten erstellt. Um komplexere Funktionalitäten bereitzustellen, wird CSS verwendet.

Vorteile von Svelte sind

- Gute Performance dank Kompilierung zu Build-Zeit
- Intuitive und einfache Syntax dank reaktiver Variablen wie (\$:)
- Svelte benötigt keine externen Bibliotheken zur Laufzeit.

Nachteile von Svelte

- Wenig Dev- und Debugging- Tools wie in anderen Web-Anwendungs-Frameworks
- Nicht so verbreitet wie große Konkurrenten, dadurch ist die Community kleiner und es gibt nicht so viel Information, Hilfe und Beispiele wie bei anderen Web-Frameworks.

Allgemein ist Svelte somit für kleine bis mittel große Projekte durch die oben genannten Vor- und Nachteile geeignet. Svelte eignet sich auch bestens für Neueinsteiger in Sachen Web-Anwendungen und wurde deshalb für das Projekt Scavenger-Hunt ausgewählt.

Eine Svelte-Web-App besteht aus Pages und Layouts. Diese sogenannten `+page.svelte` befinden sich im `routes`-Ordner der Anwendung. Die Page, welche im Root liegt, ist die erste Seite, welche geladen wird. Eine Route kann beliebig viele Unterordner oder `+page.svelte` haben. Eine `+page.svelte` kann ein `+layout.svelte` besitzen, welches das Layout, also das allgemeine Aussehen der Seite, wie Hintergrund oder Standardkomponenten wie beispielsweise Footer, vorgibt. Wenn eine `+page.svelte` kein `+layout.svelte` besitzt, dann wird das Layout vom Elternordner als Standardlayout verwendet.

Komponenten werden verwendet, um den Seiten ihre Funktionalitäten zu geben. Es gibt schon einige weit entwickelte Komponentenbibliotheken. Die schon im Projekt etablierte Bibliothek `svelte-flowbite` [8] ist eine der Hauptbibliotheken für UI-Komponenten. Hier lassen sich Komponenten wie zum Beispiel Zeitstrahlen und Modals finden, aber auch einfachere Komponenten wie einfache Knöpfe oder Textfelder. Außerdem besitzt Svelte-Flowbite eine umfangreiche Bibliothek mit Images, welche verwendet werden können.

Um während der Laufzeit der Web-App Daten speichern zu können, werden sogenannte Stores verwendet. Stores sind wie globale Container, in welche Daten gespeichert werden können und auf welche die verschiedenen Komponenten dann zugreifen können.

2.5. Node

Node.js [2] ist eine leistungsstarke, serverseitige JavaScript-Laufzeitumgebung, die auf der V8-Engine von Google Chrome basiert. Sie ermöglicht die Entwicklung hochperformanter und skalierbarer Netzwerkanwendungen, indem sie eine nicht-blockierende, ereignisgesteuerte Architektur nutzt. Diese Architektur erlaubt es, mehrere Anfragen gleichzeitig zu verarbeiten, ohne dass einzelne Prozesse blockiert werden, was Node.js besonders effizient für Echtzeitanwendungen und datenintensive Dienste macht.

Neben der Bereitstellung eines Webserverns umfasst Node.js eine Vielzahl von Modulen und Tools, die für die Entwicklung serverseitiger Anwendungen benötigt werden. Die integrierten Module ermöglichen beispielsweise Dateioperationen, Netzwerkkommunikation, Datenbankverbindungen und Prozessverwaltung, ohne dass externe Bibliotheken erforderlich sind.

Ein zentrales Merkmal von Node.js ist sein asynchrones, ereignisgesteuertes I/O-Modell. Dadurch können Anfragen effizient verarbeitet werden, da der Server nicht auf die Beendigung einer einzelnen Operation warten muss, bevor er die nächste bearbeitet. Dies macht Node.js besonders geeignet für Anwendungen, die viele gleichzeitige Verbindungen benötigen, wie beispielsweise Webanwendungen, APIs, Chat-Anwendungen oder Streaming-Dienste.

Ein weiterer wichtiger Bestandteil von Node.js ist das Routing. Mithilfe von Frameworks wie Express.js können verschiedene Routen definiert und verwaltet werden, die von einer RESTful API genutzt werden können. Diese Routen ermöglichen den Zugriff auf unterschiedliche Endpunkte der Anwendung, um beispielsweise Daten abzurufen, zu erstellen oder zu aktualisieren.

Zusätzlich bietet Node.js eine enge Integration mit dem Paketmanager npm (Node Package Manager), welcher eine umfangreiche Sammlung von Bibliotheken und Modulen bereitstellt. Dies erleichtert die Entwicklung, da bestehende Lösungen genutzt und wiederverwendet werden können, ohne das Rad neu erfinden zu müssen.

Durch diese Eigenschaften ist Node.js eine beliebte Wahl für moderne Webanwendungen und Microservices-Architekturen. Die Kombination aus hoher Performance, Skalierbarkeit und der Möglichkeit, sowohl client- als auch serverseitig mit JavaScript zu arbeiten, macht es besonders attraktiv für Entwickler.

2.6. Entity Framework Core

Entity Framework Core Entity Framework Core (EFC) [4] ist ein leichtgewichtiges, erweiterbares und plattformübergreifendes objekt-relationales Mapping Object-Relational Mapping (ORM)-Framework von Microsoft. Es ermöglicht Entwicklern, mit Datenbanken unter Verwendung von .NET-Objekten zu arbeiten, ohne direkt SQL-Abfragen schreiben zu müssen. Da EFC Datenunabhängigkeit bietet, kann es auch problemlos schon in bestehenden Anwendungen eingesetzt werden. Hauptmerkmale von EFC sind

- Plattformübergreifend auf Windows , Linux oder MacOS Nutzbar.
- Dank Datenunabhängigkeit werden mehrere Datenbanken wie SQL Server, MySQL , SQLite usw. unterstützt.

- Migrationen also dass ändern von Zeilen und Spalten in der Datenbank kann einfach durch Migrationsscripte vorgenommen werden
- Keine SQL-Abfragen Objekte der Datenbank.

Objekte in der Datenbank werden von EFC unique Keys zugewiesen. Mit diesen Keys können dann intuitiv read, update oder delete auf das Objekt ausgeführt werden. Dadurch erhöht sich die Lesbarkeit und Wartbarkeit des Codes erheblich.

2.7. InstallForge

InstallForge[3] ist ein benutzerfreundliches und kostenloses Tool zur Erstellung von Installationspaketen für Windows-Anwendungen. Es ermöglicht Entwicklern, maßgeschneiderte Setup-Dateien zu erstellen, die den Installationsprozess ihrer Software auf Zielsystemen automatisieren. Die Software bietet eine intuitive grafische Benutzeroberfläche, die es auch Anfängern im Bereich der Softwareentwicklung ermöglicht, Installationsprogramme zu erstellen, ohne tiefgehende Programmierkenntnisse zu benötigen. Mit Funktionen wie dem Hinzufügen von Dateien, dem Erstellen von Verknüpfungen, der Definition von Zielordnern und der Anpassung von Installationsdialogen ermöglicht InstallForge eine hohe Flexibilität. Zudem unterstützt es erweiterte Funktionen wie die Durchführung von benutzerdefinierten Skripten und die Silent Installation. InstallForge ist kostenlos, Open Source und bietet eine einfache Lösung, um Installationspakete zu erstellen, die auf verschiedenen Windows-Versionen problemlos funktionieren.

3. Problemanalyse

Das Projekt Scavenger-Hunt ist geplant, um Erstsemester-Studenten spielend die verschiedenen Orte der Hochschule zu zeigen. Deshalb war der Fokus während des gesamten Projekts auch auf der einfachen und intuitiven Bedienbarkeit des Spiels.

3.1. Bestandsaufnahme

Das Projekt wurde als Erweiterungsprojekt des bereits vorhandenen Scavenger-Hunt-Spiels geplant. Da die grundlegende Architektur bereits gegeben und auch gut implementiert war, lag der Fokus bei der Projektarbeit darauf, eventuell vergessene Features zu implementieren und den Ablauf des Spiels flüssiger zu gestalten. Über den Hunt Editor ist es möglich, Schnitzeljagden zu erstellen. Eine Schnitzeljagd setzt sich hierbei aus einem oder mehreren Assignments zusammen. Jedes Assignment besitzt einen Hint (Text oder Bild) und eine Solution (Text, QR-Code oder Location). Erstellte Schnitzeljagden werden in der Datenbank abgespeichert. Im Editor können die gespeicherten Schnitzeljagden dann wieder geöffnet, bearbeitet und gespeichert werden. Über den manuell eingetragenen Link einer Schnitzeljagd kann man sich zu der Schnitzeljagd über die Hunt-Participation registrieren. Durch die Registrierung wird der Nutzer mit der Hunt verknüpft und kann sich dann über die Hunt-Web-Game-Anwendung anmelden. Auf die Web-App Hunt-Web-Game gelangt man, indem man einen weiteren Link eingibt. Nachdem man sich angemeldet hat, befindet man sich in der Game-Ansicht. Hier sieht man alle Schnitzeljagden, bei denen man sich angemeldet hat, als Kartenansicht. Man kann dann auf den „Play“-Button einer Schnitzeljagd klicken und das Spiel starten. Es werden dann nacheinander die Assignments durchgegangen. Beim richtigen Beantworten eines Hints wird der Hint des nächsten Assignments geladen. Bei Falscheingabe wird ein Hinweis zum jeweiligen Hint gegeben. Wenn das letzte Assignment gelöst wird, erhält der Nutzer eine Glückwunsch-Nachricht.

3.1.1. Anforderungen

Nach ausgiebiger Recherche, Tests mit der vorhandenen Anwendung und Absprache mit Stakeholdern haben sich dann folgende Features, Erweiterungen oder Verbesserungen herauskristallisiert:

Must Have	Should have
<ul style="list-style-type: none"> - EXE für das Starten des Servers - Webapp für die Spieler - Optimierung der Performance - Einladung als Link zum beitreten - Erweiterung / Verbesserung der User Experience 	<ul style="list-style-type: none"> - Erweiterung der "Hints" mit Video und Ton - Testen des Projektes - Hunt-Editor überarbeiten (Einklappen von Location, Nummerierung der Hints, etc...)
Could have	Nice to have
<ul style="list-style-type: none"> - Kombinieren von "Hint-Typen" miteinander - Erstellen einer „Vorzeige“ Schnitzeljagd für die Ersties - Erweiterung der Hunt mit „Abschluss Text“ 	<ul style="list-style-type: none"> - Geocaching - AR

Abbildung 3.1.: Ziele der Projektarbeit

Die Folgenden drei Überpunkte können aus den Anforderungen Abgeleitet werden.

Benutzerfreundlichkeit

Wie aus den Anforderungen entnehmbar war, lag der große Fokus auf der Benutzerfreundlichkeit der Anwendung. Die Anwendung wird auch von angehenden Studenten verwendet und dient sozusagen auch als Aushängeschild dafür, was als Student an der Hochschule Aalen alles erreicht werden kann. Einige Punkte in der 3.1 sind nur kurz per Überbegriff beschrieben. Die genauen Schritte werden im Kapitel 4 genauer beschrieben. Vor allem der Anmeldeprozess sollte einfacher gestaltet werden, da man sich über einen Link zu einer Hunt registrieren muss und dann über einen zweiten Link an der Hunt angemeldet werden kann. Dieser Prozess sollte vereinfacht werden, damit ein Nutzer der Anwendung nur einen Link eingeben oder einscannen muss. Außerdem sollte beim Spielen für den Nutzer noch zusätzliche Informationen bereitstehen, um verschiedene Hints zu lösen. Der Fokus der Benutzerfreundlichkeit lag nicht nur auf den Endverbrauchern, sondern auch auf den Personen, die mithilfe des Hunt-Editors 4.2.1 Hunts erstellen. Um diesen Prozess zu vereinfachen, war der Plan, eine kleine grafische Benutzeroberfläche (GUI) zu erstellen, mit der man sich einfach und intuitiv durch das Projekt klicken kann, Hunts erstellen und testen kann.

Sicherheit

Auch zum Thema Sicherheit wurde sich bei der Erweiterung nochmal Gedanken gemacht. Auch im vorhergehenden Projekt wurde sich schon Gedanken gemacht, wie man durch kryptographisches Hashen der Benutzerdaten die Sicherheit verbessern könnte. Bei der Erweiterung wurde versucht einige Sicherheitsaspekte zu verbessern aber der Fokus lag auch bei der Erweiterung mehr auf der Funktionalität und Erweiterung der Features. Einige Sicherheitsverbesserungen wurden dennoch umgesetzt wie man aus 4.3 sehen kann.

Skalierbarkeit

Die Architektur wurde schon so gewählt dass es möglich sein sollte mehrere Schnittzeigten gleichzeitig zu erstellen und zu Spielen. Durch das hinzufügen von neuen Hint-Typen wie Video oder Audio wurde diese Skalierbarkeit auf die Probe gestellt und es haben sich auch hier einige Verbesserungsmöglichkeiten gezeigt.

4. Erweiterungsentwurf

4.1. Blockansicht

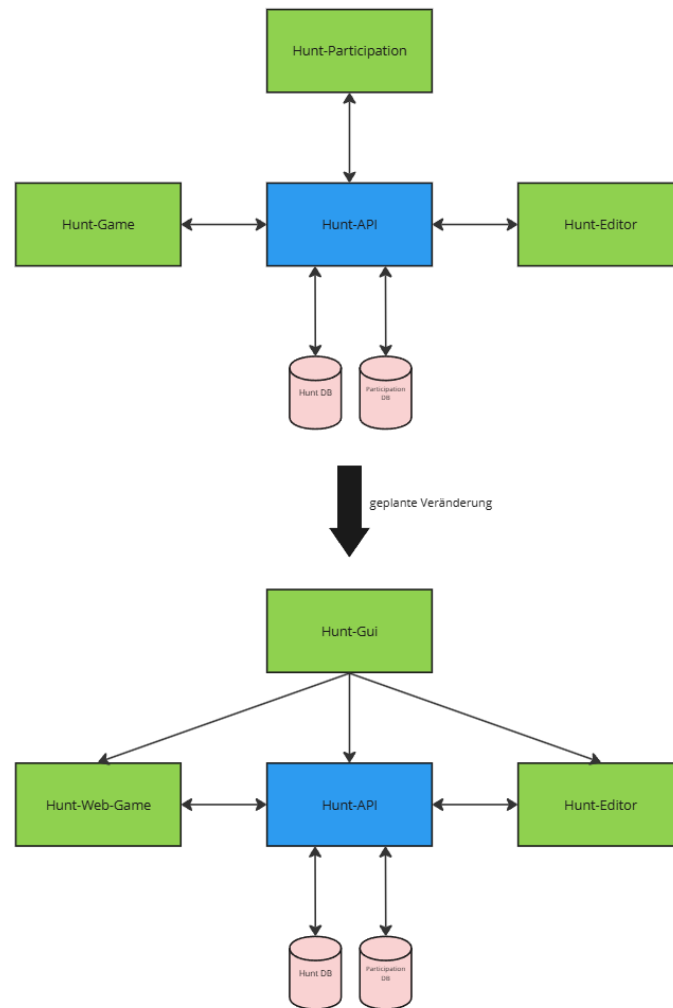


Abbildung 4.1.: Kontext Diagramm der Anwendung

Das in 4.3 gezeigte Kontext Diagramm zeigt das Zusammenspiel der einzelnen Komponenten des Scavanger-Hunt Spiels. Die oben gezeigte Darstellung zeigt, wie das System vor der Erweiterung aufgebaut war. Das Schaubild darunter beschreibt die geplanten Veränderungen. Die grün markierten Felder sind die jeweiligen Anwendungen mit User Interface (UI). Die Schnittstellen zwischen Hunt-API (blau), Hunt-Editor und Hunt-Web-Game werden durch REST-API erreicht. Die Hunt-API dient als Backend und somit als das Herz der Anwendung. Die in Rosa markierten Zylinder stellen die Datenbankbindung dar. Da im Backend und bei den Datenbanken nur kleinere Veränderungen vorgenommen wurden, werden die Veränderungen in den Sektionen 4.3 und 4.2.1 genauer beschrieben. Für genaueren Einblick in die Hunt-API und die Datenbank wird die [vorausgegangene Dokumentation](#) als Referenz empfohlen.

4.2. Hunt-GUI

Die Idee einer zentralen grafischen Oberfläche (GUI) wurde entwickelt, um den gesamten Startprozess der Schnitzeljagd-Anwendung zu vereinfachen und zu optimieren. Ziel ist es, den Nutzern ein benutzerfreundliches Erlebnis zu bieten, das eine schnelle und unkomplizierte Nutzung der Anwendung ermöglicht. Durch die Bereitstellung einer GUI, die sämtliche notwendigen Startkomponenten automatisiert, wird die Notwendigkeit reduziert, dass Nutzer manuell mehrere Prozesse starten oder konfigurieren müssen. Dies sorgt nicht nur für eine erhebliche Zeiterparnis, sondern auch für eine höhere Benutzerfreundlichkeit, da die Anwendung ohne größere Einarbeitungszeit direkt verwendet werden kann.

Die GUI ist so konzipiert, dass sie beim Starten automatisch Docker und dessen Container öffnet, was den Einsatz von Containern für das Backend und andere notwendige Komponenten erleichtert. Die Benutzer müssen sich nicht mit komplexen Terminalbefehlen oder technischen Details auseinandersetzen, sondern können einfach per Knopfdruck entweder den Editor oder das Spiel starten. Diese Automatisierung steigert die Effizienz und macht die Anwendung zugänglicher für Nutzer, die mit der zugrunde liegenden Technologie wie Docker nicht vertraut sind.

Ein weiteres wichtiges Merkmal der GUI ist die Unterstützung von zwei unterschiedlichen Modi: einem Dark-Mode und einem White-Mode. Dies gibt den Nutzern die Möglichkeit, die Anwendung an ihre individuellen Vorlieben anzupassen, was sowohl das visuelle Erlebnis als auch die Benutzerfreundlichkeit fördert.

Die Gestaltung der GUI wurde bewusst minimalistisch gehalten. Die Oberfläche ist klar strukturiert, mit nur den wichtigsten Steuerungselementen, die der Nutzer benötigt, um die Anwendung zu bedienen. Zu den wesentlichen Elementen gehören die Schaltflächen zum Starten des Editors oder Spiels, sowie die Möglichkeit, zwischen dem Dark- und White-Mode zu wechseln. Diese Reduktion auf das Wesentliche sorgt

dafür, dass die Anwendung auch für neue Nutzer schnell verständlich und intuitiv ist.

Für den visuellen Bezug zur Hochschule Aalen wurde ein ansprechendes Hintergrundbild integriert, das den Campus zeigt, sowie ein Hochschul-Icon, das das Branding und die Zugehörigkeit zur Hochschule unterstreicht. Diese Designentscheidungen tragen dazu bei, dass sich die Nutzer mit der Anwendung verbunden fühlen und gleichzeitig eine klare visuelle Identität für das Projekt geschaffen wird.

Insgesamt ermöglicht die GUI nicht nur eine einfache Bedienung der verschiedenen Komponenten der Schnitzeljagd, sondern sorgt auch für eine angenehme und benutzerfreundliche Interaktion. Sie stellt sicher, dass die Anwendung sowohl für erfahrene Nutzer als auch für Anfänger zugänglich bleibt und optimiert die Benutzererfahrung durch durchdachte Designentscheidungen und benutzerorientierte Funktionen.

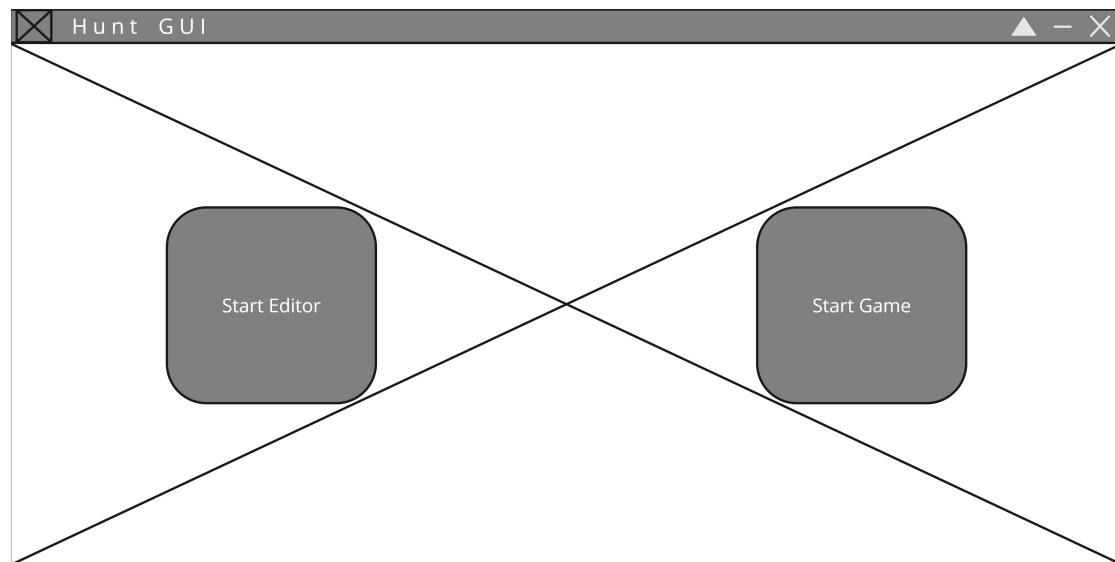


Abbildung 4.2.: Dark Mode der GUI

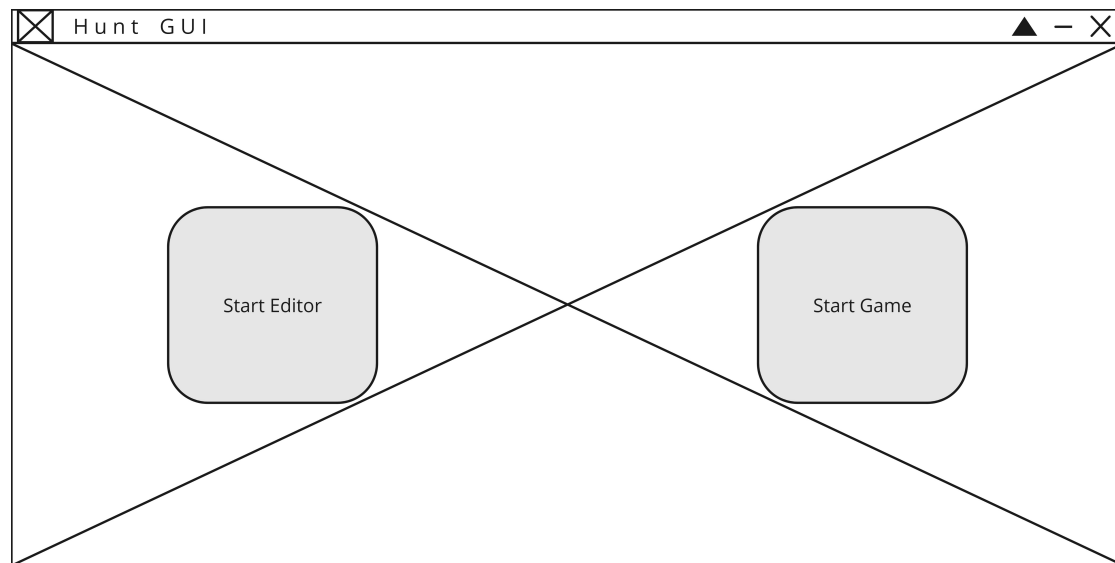


Abbildung 4.3.: White Mode der GUI

4.2.1. Hunt-Editor

Der Hunt-Editor wird verwendet, um Schnitzeljagden zu erstellen und zu verwalten. Auch im Editor wurden einige Veränderungen und Anpassungen vorgenommen. Hier war vor allem das Augenmerk darauf, die zusätzlichen Hint-Typen zu implementieren. Außerdem wurden viele kleine Verbesserungen vorgenommen, damit die Bedienung einfacher und intuitiver wird.

Wireframes wurden genutzt, um Anforderungen wie das Nutzerhandling oder die Interaktivität der Erweiterung zu überprüfen und weitere Schritte zu planen.

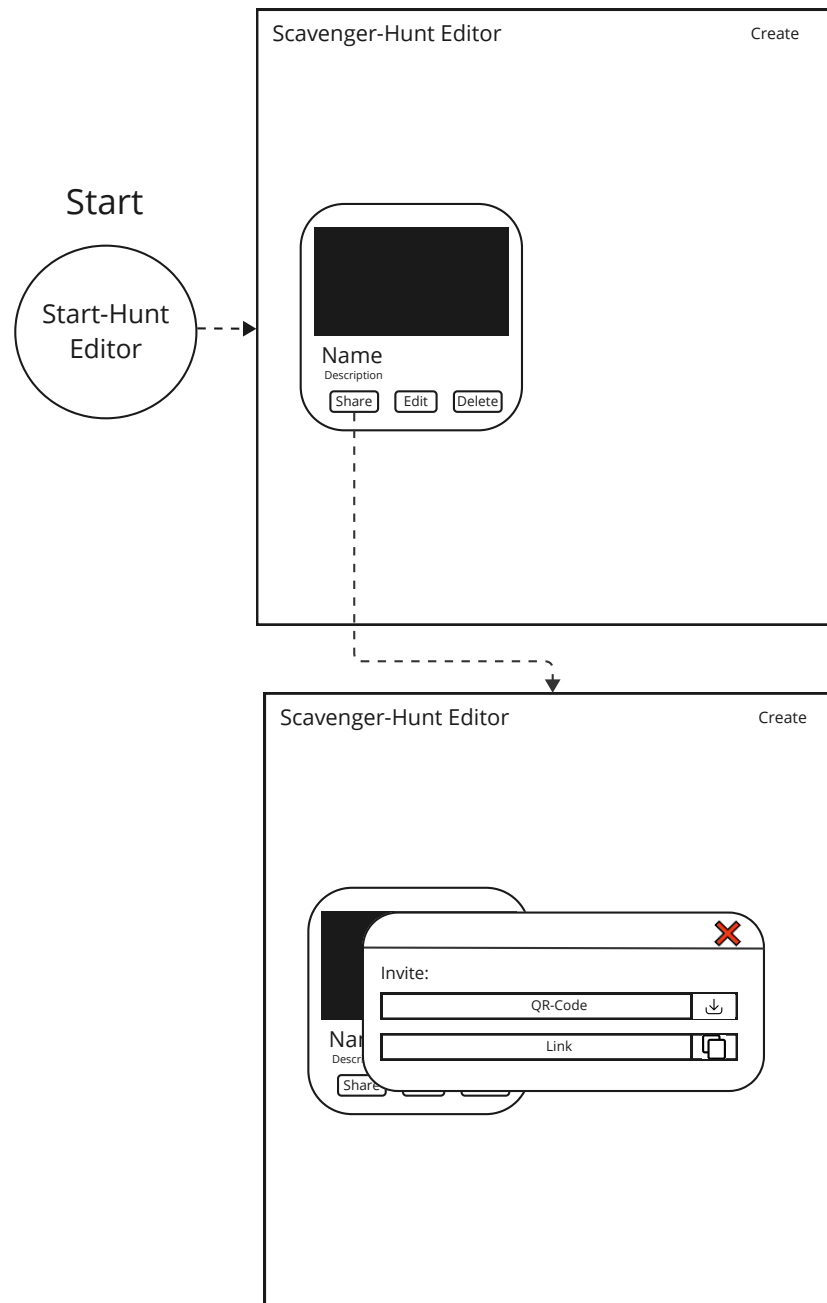


Abbildung 4.4.: Wireframe zur Share Funktion

Dieses Wireframe zeigt die geplante Share-Funktion. Die gestrichelten Linien sollen die Übergänge zu den verschiedenen Zuständen darstellen. Der Share Button sollte für den Ersteller einer Schnitzeljagd ein einfacher Weg sein, seine Schnitzeljagd zu teilen. Da die Schnitzeljagden ja für Erstsemester Studenten geplant ist und sich diese noch nicht so gut mit den Studenten Tools wie Studmail auskennen war die Überlegung, dass man den Link zum Beitreten als QR Code darstellen kann. Diesen können die Studenten dann ganz einfach einscannen, ohne sich mit den Tools auskennen zu müssen. Der andere Weg ist den Link in die Ablage zu kopieren. Von dort kann man ihn dann einfach in eine Rundmail einfügen und so die Studenten zum Spielen einladen.

Geplant waren auch für den Editor noch weitere Features, welche kein Wireframe benötigt haben. Zum einen wurde geplant, dass weitere Hint-Typen in Form von Videos und Audio hinzukommen. Im Laufe der Arbeit hat sich dann bemerkbar gemacht, dass nur ein Bild oder ein Video als Hint-Typ eventuell nicht aussagekräftig genug ist. Deshalb wurde sich die Überlegung gemacht, dass man bei Bildern und Videos noch einen zusätzlichen Text mit angeben kann, welcher dann das Rätsel noch etwas genauer erklärt.

Um die Benutzung des Editors noch intuitiver zu gestalten, war geplant, noch einen „Zurück“-Knopf einzufügen, mit welchem man während der Erstellung einer Schnitzeljagd zwischen den verschiedenen Ansichten (Schnitzeljagd-Beschreibung, Add Assignment und Overview) hin- und herwechseln kann. Außerdem wurde geplant, über ein Assignment die Assignment-Nummer sowie Hint-Type und Solution-Type zu schreiben, damit der Nutzer einen schnelleren Überblick über die verschiedenen Assignments seiner Schnitzeljagd bekommt.

4.3. Hunt-Web-Game

Da der Prozess des Registrierens und Anmeldens zu einer Schnitzeljagd nicht sehr intuitiv war, wurden sich Gedanken gemacht, wie man diesen Ablauf leichter gestalten kann. Hierzu wurden wieder einige Wireframes erstellt, um diese Überlegungen mit Stakeholdern abzuklären. Aus den Überlegungen resultierenden die Wireframes C.1, 4.5 und 4.6. Der geplante Normalablauf für Nutzer sollte per QR Code oder Link im Web Game Screen 1 C.1 starten. Nachdem der Button "Participate!" betätigt wird, soll der Nutzer die Möglichkeit haben, einen Benutzernamen und ein Passwort einzugeben.

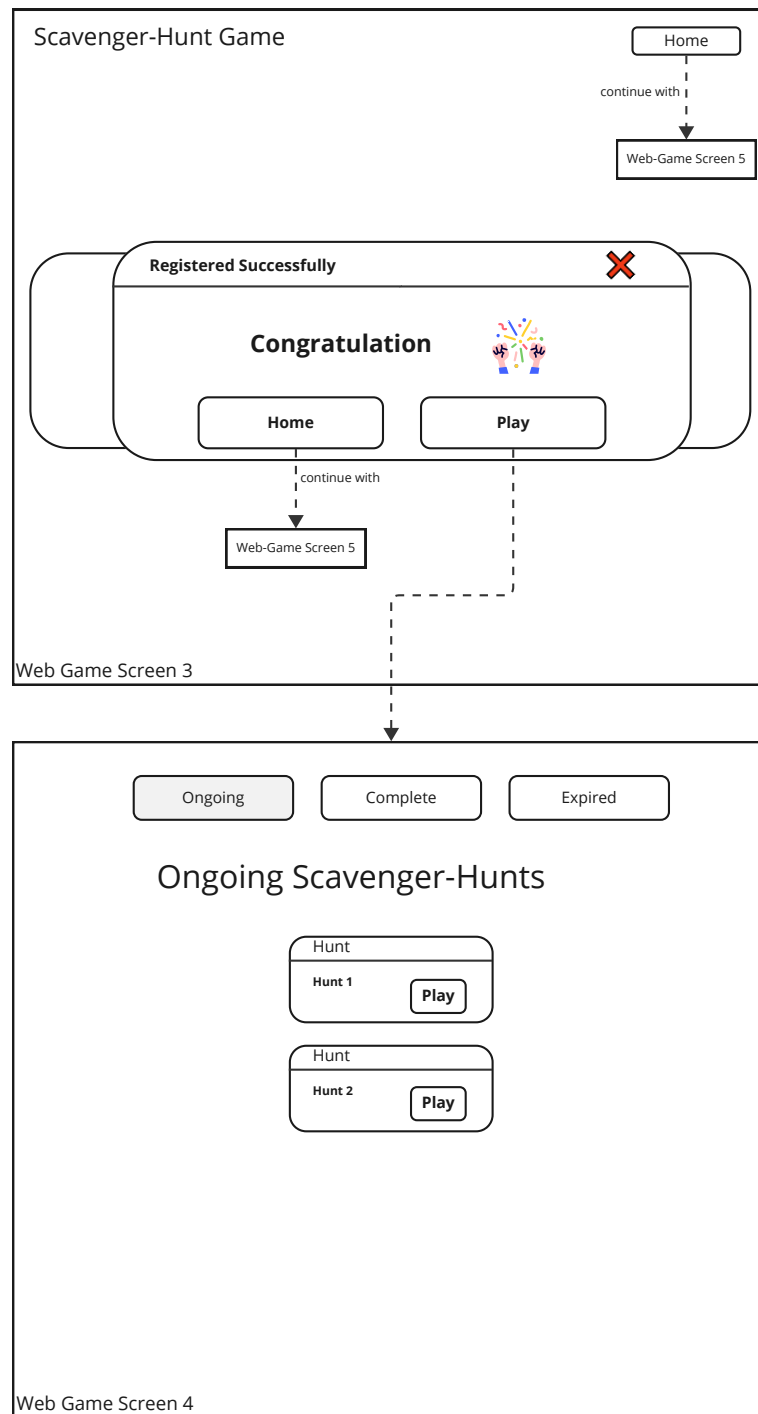


Abbildung 4.5.: Hunt Web Game Wireframe vom Login Prozess

Dann falls er noch nicht registriert war und "Register"betätigt sollte er Feedback 4.5 ähnlich wie in Web Game Screen 3 bekommen. Falls der Nutzer sich für die Hunt registriert hat und sich nur einloggen möchte kann er dies durch den im Web Game Screen 2 befindenden "Login"button machen.

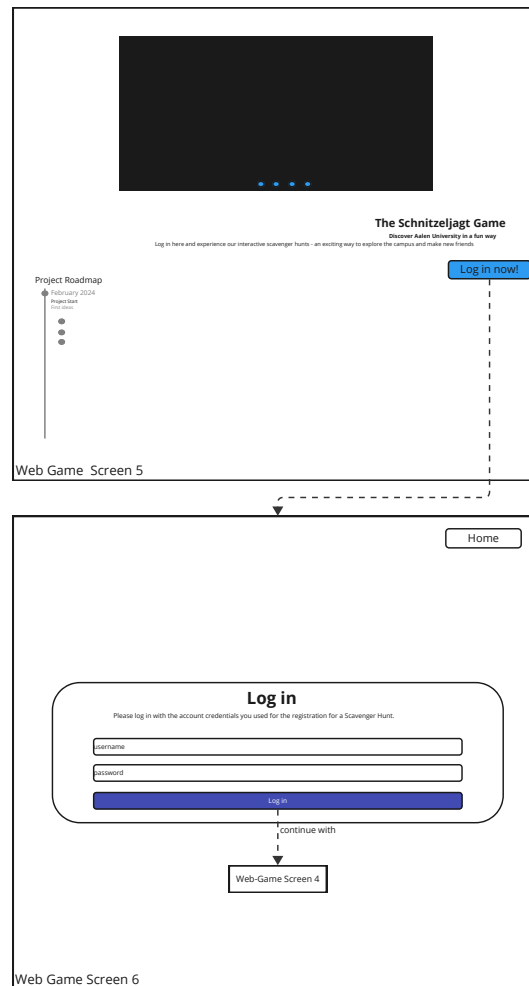


Abbildung 4.6.: Hunt Web Game Wireframe vom Login Prozess

Über die "Home"Button im rechten Eck kommt man auf den Web Game Screen 5 4.6. Von hier aus kann man sich wieder einloggen und alte schon registrierte Hunts spielen.

4.4. Spielablauf

Der Spielablauf wurde auf zwei Aktivitätsdiagramme aufgeteilt das erste Diagramm 4.7 beschreibt den Ablauf des Registrieren und Anmelden beim Hunt Web Game.

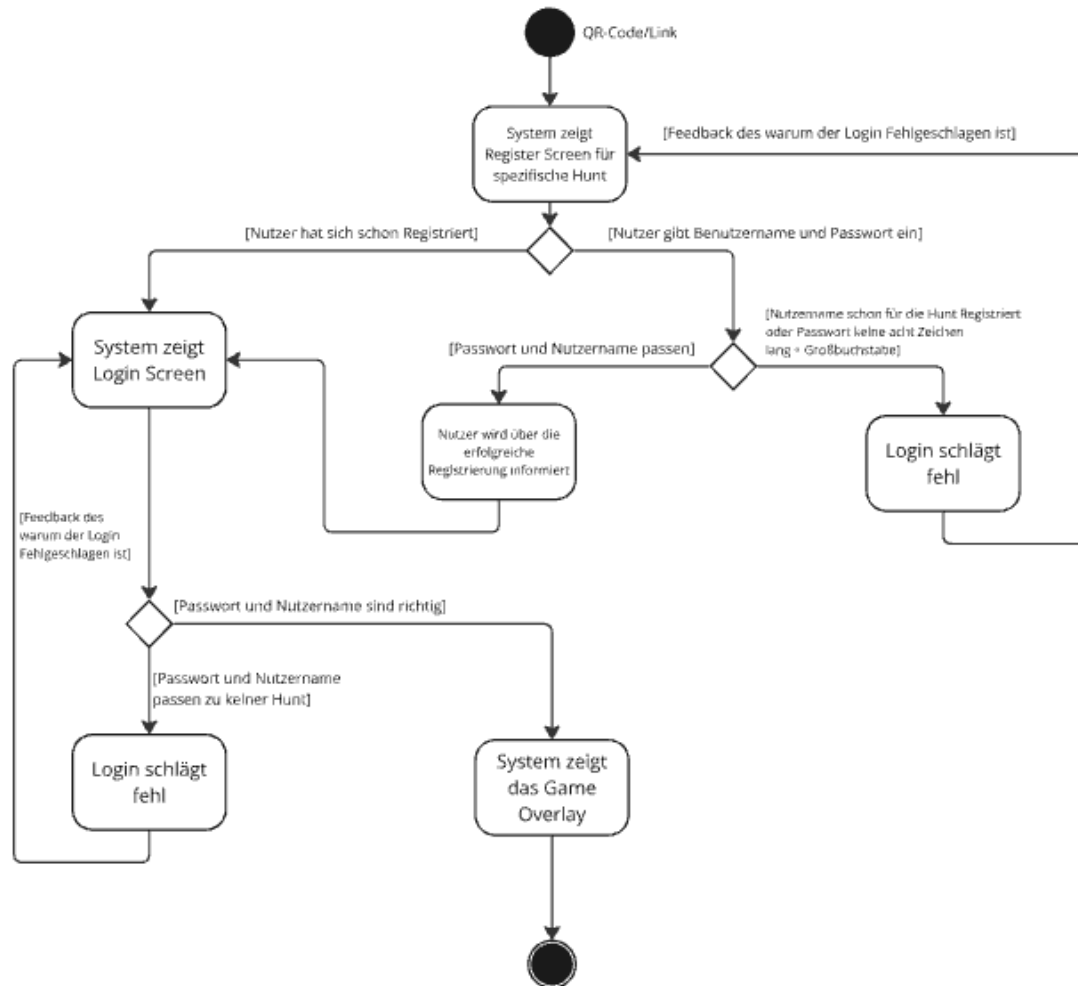


Abbildung 4.7.: Aktivitätsdiagramm des Registrier Prozesses

Der Startpunkt ist hierfür das Einscannen eines QR-Codes oder das eingeben eines Links. Der Nutzer wird dann auf eine Seite geleitet auf welche er die Überschrift und Beschreibung der Hunt sieht außerdem gibt es ein Feld zum eingeben des Nutzernamen und ein Feld zum eingeben des Passworts. Unter alle dem kann der Nutzer falls er sich schon zu einem früheren Zeitpunkt registriert hat auch direkt mit seinen alten Nutzerdaten anmelden. Wenn der Nutzer sich erfolgreich

Registriert hat wird dies per Pop up bestätigt. Falls der Nutzer ein schon für die Hunt Registrierten Nutzer oder ein Passwort wählt welches zu kurz ist und zu wenig Groß/Kleinbuchstaben besitzt wird er mit einer Feedbacknachricht auf den Fehlschlag hingewiesen. Wenn der Nutzer sich dann erfolgreich registriert hat kann er sich einloggen. Dafür muss er nochmal seine Account Daten eingeben. Hier wird nochmal geschaut ob der Nutzer für eine Hunt registriert und ob dass Passwort zum eingegebenen Nutzer stimmt. Bei Falscheingabe oder falls kein Nutzer mit der Kombination besteht wird der Nutzer darauf hingewiesen. Wenn dann alles richtig eingegeben wurde kommt der Nutzer ins Game Overlay. Hier werden alle Hunts angezeigt.

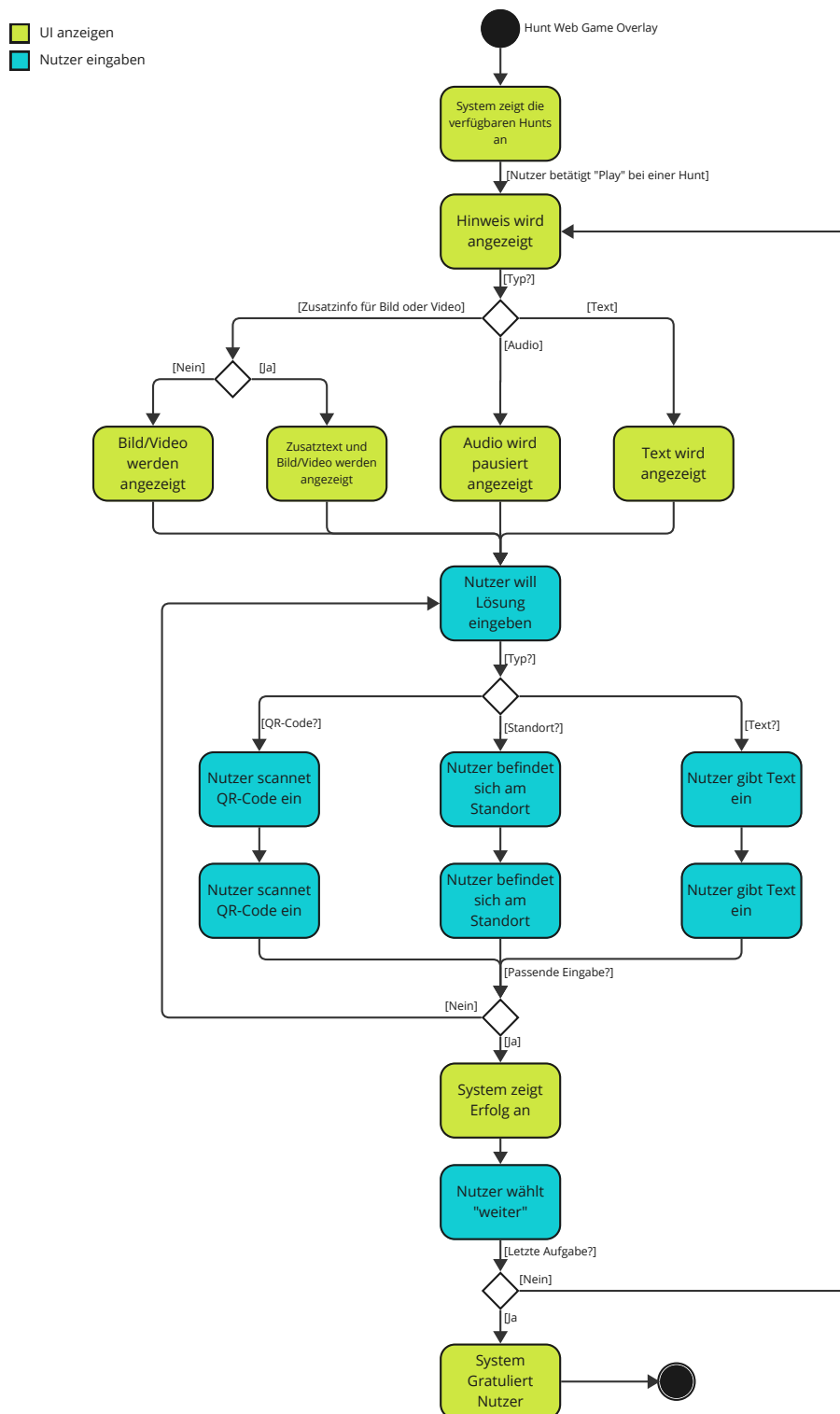


Abbildung 4.8.: Aktivitätsdiagramm des Spiels

Wenn der Nutzer nun auf eine aktuelle Hunt zum Spielen klickt fängt das Spiel an. Das System zeigt dem Nutzer den ersten Hinweis an. Es gibt vier verschiedene Arten von Hinweisen.

- **Bild:** Dem Nutzer wird ein Bild angezeigt. Zusätzlich kann hier optional noch ein Erklärungstext stehen welcher das Bild weiter erklärt.
- **Video:** Dem Nutzer wird ein Video angezeigt. Das Video befindet sich im pausierten und stummgeschalteten Zustand. Zusätzlich kann hier optional noch ein Erklärungstext stehen welcher das Video weiter erklärt.
- **Audio:** Dem Nutzer wird eine Audio angezeigt. Die Audio befindet sich im pausierten Zustand.
- **Text:** Dem Nutzer wird ein Text angezeigt.

Wenn der Nutzer dann überlegt hat kann dieser eine Lösung eingeben. Es gibt drei verschiedene Lösungsarten.

- **Text:** Der Nutzer kann seine Lösung in ein Textfeld eintragen. Bei Fehleingabe wird ermittelt um wieviele Zeichen der Nutzer daneben liegt und dann wird dieser darauf hingewiesen.
- **QR-Code:** Wenn QR-Code als Lösung ausgewählt ist wird die Kamera des Gerätes aktiviert um den QR-Code einzuscannen.
- **Standort:** Es wird überprüft ob der Nutzer sich am Ort der gesucht wird befindet. Sollte der Nutzer noch zu weit entfernt sein wird er mit einer Meteranzahl auf den Unterschied zu seiner Position hingewiesen.

Bei richtiger Eingabe und falls noch Aufgaben aus der Schnitzeljagd ungelöst sind wird der nächste Hinweis angezeigt. Dieser Vorgang wiederholt sich so lange bis keine Aufgaben mehr übrig sind.

5. Implementierung

5.0.1. Aufbau der Hunt-GUI

Für die grafische Benutzeroberfläche (GUI) wurde C# in Kombination mit WPF verwendet. WPF ermöglicht eine einfache und schnelle Gestaltung der GUI. Die Struktur von WPF basiert auf einem MainWindow, das aus einer C#-Klasse und einer XAML-Klasse besteht. Während die XAML-Klasse für das Design bzw das Layout der Oberfläche verantwortlich ist, übernimmt die C#-Klasse die entsprechenden Backend-Funktionen.

Zusätzlich zum MainWindow besitzt ein WPF-Projekt standardmäßig auch eine App-Klasse, die ähnlich aufgebaut ist. Der Unterschied zum MainWindow besteht darin, dass diese Klasse ausschließlich für das Design der Elemente in der GUI verantwortlich ist. Hier können z. B. Stile für Knöpfe, Slider oder benutzerdefinierte Elemente definiert werden, die dann überall im System verwendet werden können.

Um eine GUI tatsächlich umsetzen zu können, ist empfohlen ein vordefiniertes Framework wie Model-View-Controller (MVC[10]) oder Model-View-ViewModel (MVVM[11]) zu Verwenden. Diese bieten meist einfache Lösungen für viele Herausforderungen, die beim Aufbau einer GUI beachtet werden müssen. In diesem Projekt wurde eine angepasste Variante des MVVM-Frameworks verwendet. Der Aufbau von MVVM ist sehr einfach und basiert auf drei Aspekten:

- **Das Model:** Repräsentiert die Daten oder die Business-Logik des Projekts.
- **Die View:** Repräsentiert die einzelnen Ansichten (Views).
- **Das ViewModel:** Fungiert als Brücke zwischen Model und View, verarbeitet die User-Interface-Logik (UI-Logik) und bindet Daten an die Ansicht.

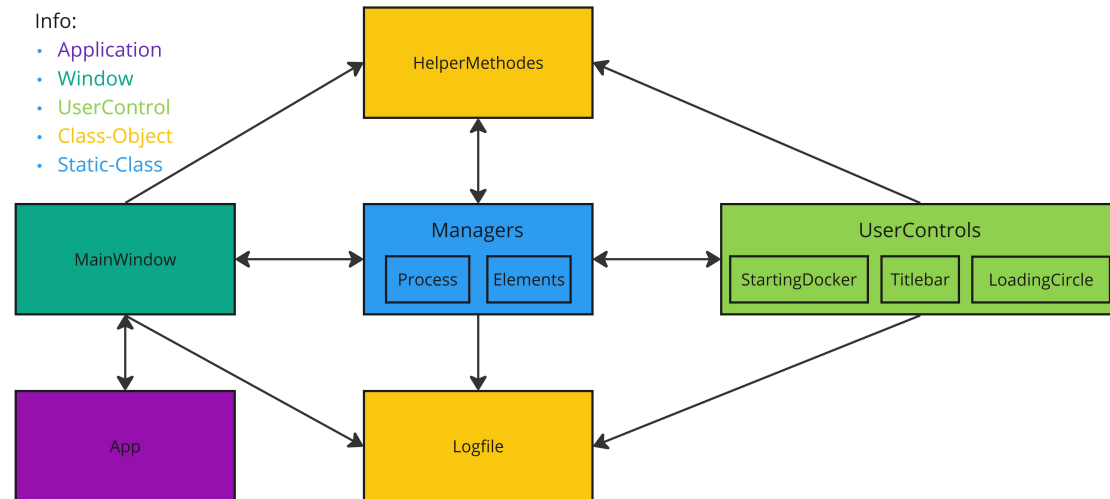


Abbildung 5.1.: Relationsdiagramm GUI

In 5.1 ist der Aufbau unserer GUI und ihre Relationen beschrieben. Dabei entspricht das Model den HelperMethodes sowie der MainWindow.cs, die View besteht aus MainWindow.xaml und den UserControls, und das ViewModel wird durch die Manager repräsentiert. Zusätzlich gibt es noch die Klasse Logfile, die für das Logging verantwortlich ist, sowie die App-Klasse, die wie zuvor beschrieben, den Stil der GUI verwaltet.

Der hier verwendete Aufbau ist nur eine Variante des MVVM-Frameworks, da die Trennung von Model und View nicht vollständig umgesetzt ist. Unsere MainWindow-Klasse entspricht sowohl dem Model als auch der View, wodurch die Interaktion nicht über das ViewModel erfolgt, sondern direkt zwischen Model und View.

Es wäre möglich, MVVM vollständig umzusetzen, was in diesem Fall jedoch nicht erforderlich war, da die GUI weder umfangreich noch komplex sein sollte. Dennoch bietet diese Umsetzung alle notwendigen Vorteile von MVVM, um eine stabile und erweiterbare GUI zu gewährleisten.

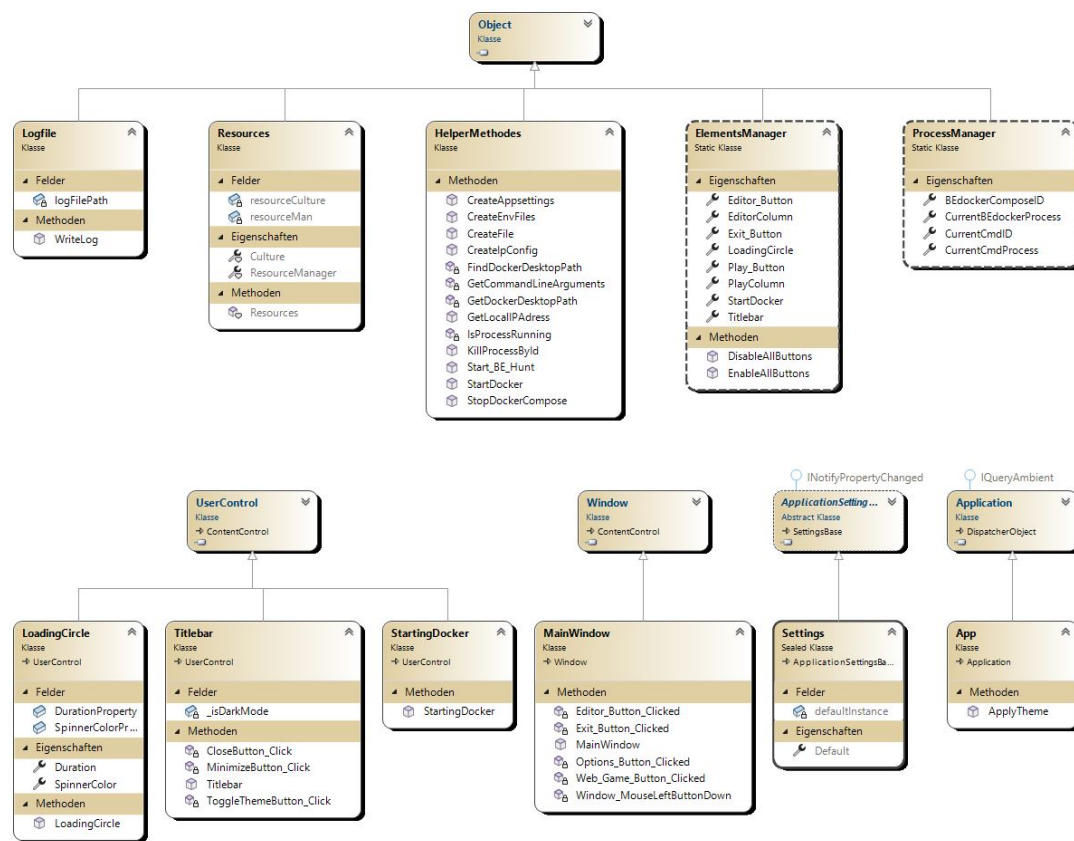


Abbildung 5.2.: Klassendiagramm GUI

Wie in 5.1 beschrieben, zeigt das Klassendiagramm eine angepasste Variante des MVVM-Patterns, wobei die einzelnen Bestandteile nicht strikt getrennt sind, sondern auf die Anforderungen der Anwendung abgestimmt wurden. Das Model wird hier durch die `HelperMethodes`-Klasse sowie die `MainWindow.cs`-Datei repräsentiert. `HelperMethodes` enthält zentrale Methoden zur Verwaltung von Konfigurationsdateien, zum Starten und Stoppen von Docker sowie zur Prozesskontrolle. `MainWindow.cs` übernimmt sowohl Aufgaben der Geschäftslogik als auch die Steuerung der Benutzeroberfläche, wodurch die Trennung zwischen Model und View in dieser Implementierung nicht vollständig eingehalten wird.

Die View besteht aus `MainWindow.xaml` und den verschiedenen `UserControls`, die spezifische UI-Elemente enthalten. `UserControl` dient als Basisklasse für Elemente wie `LoadingCircle`, der für Ladeanimationen zuständig ist, `Titlebar`, die Schaltflächen für Minimieren, Schließen und das Umschalten des Dark-/White-Modus bereitstellt, und `StartingDocker`, das die Steuerung des Docker-Starts ermöglicht. Diese Komponenten ermöglichen eine modulare und wiederverwendbare Gestaltung der Benutzeroberfläche.

Das ViewModel wird durch verschiedene Manager-Klassen realisiert. ElementsManager übernimmt die Verwaltung von UI-Elementen und steuert deren Aktivierung oder Deaktivierung. ProcessManager verwaltet die laufenden Prozesse und speichert wichtige Informationen zu den aktiven Docker- und CMD-Prozessen. Diese Manager fungieren als Vermittler zwischen der View und der Logik, wodurch eine strukturierte Organisation der Anwendung sichergestellt wird.

Zusätzlich zur MVVM-Architektur gibt es die Logfile-Klasse, die für das Logging verantwortlich ist und eine zentrale Methode WriteLog bereitstellt, um Anwendungsereignisse zu protokollieren. Die App-Klasse ist für die Verwaltung des UI-Stils zuständig und definiert globale Designrichtlinien für die gesamte Anwendung.

Insgesamt zeigt das Diagramm eine flexible und pragmatische Umsetzung des MVVM-Patterns, bei der gewisse Trennungen zwischen Model, View und ViewModel zugunsten einer einfacheren Architektur bewusst aufgehoben wurden. Diese Struktur ermöglicht eine effiziente Entwicklung und Wartung der Anwendung, während gleichzeitig die Vorteile des MVVM-Designs weitgehend erhalten bleiben.

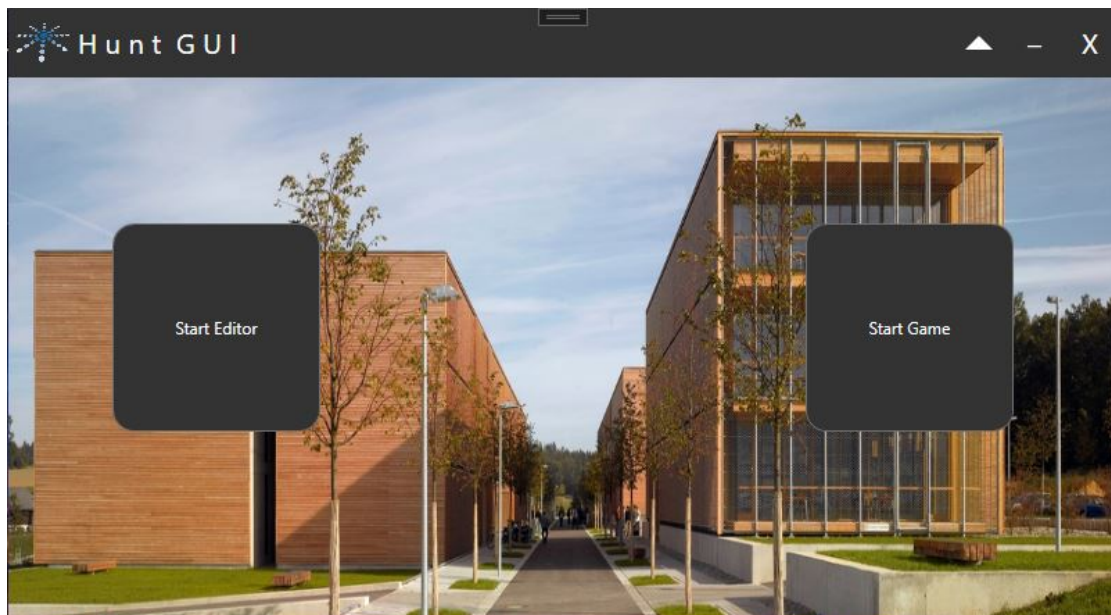


Abbildung 5.3.: Fertige GUI (Black)

Der Aufbau der GUI ist bewusst minimalistisch gehalten (siehe 5.3) und verfügt über lediglich zwei Knöpfe: „Start Editor“ und „Start Game“. Zusätzlich gibt es eine Titelleiste mit der Option, über das Dreieck-Symbol zwischen dem Black- und White-Mode zu wechseln. Das Fenster kann über das Minus-Symbol minimiert und über das X-Symbol geschlossen werden. Im Hintergrund ist ein Bild des Hochschulcampus Burren zu sehen.

Der Ablauf beim Starten der GUI ist folgender: Nach dem Doppelklick auf die Hunt-GUI-Verknüpfung wird zunächst ein Ladebildschirm angezeigt, während Docker gestartet wird. Sobald Docker bereit ist, erscheint die in Abb. 5.3 dargestellte Oberfläche.

Der Benutzer hat dann die Möglichkeit, entweder auf „Start Editor“ oder „Start Game“ zu klicken. Dies öffnet zwei CMD-Fenster (siehe C.2): Eines davon ist für docker-compose zuständig und startet die Backend-Docker-Container, während das andere den Frontend-Editor oder das Frontend-Game startet.

Beim ersten Starten von entweder dem Frontend-Editor oder dem Frontend-Game (nach erneutem Öffnen der GUI) wird das docker-compose-CMD-Fenster geöffnet. Danach bleibt dieses CMD-Fenster aktiv, bis die GUI vollständig über den X-Knopf geschlossen wird. Dies ermöglicht einen schnellen Wechsel zwischen Editor und Game, ohne dass die Backend-Docker-Container bei jedem Wechsel neu gestartet werden müssen.

Um anzuzeigen, welches Element gerade aktiv ist, der Editor oder das Game, wird ein Ladebalken über dem jeweiligen aktiven Element eingeblendet. Zudem gibt es einen „Exit“-Knopf, mit dem das aktuell laufende Element beendet werden kann. Da immer nur eines der beiden Elemente gleichzeitig ausgeführt werden kann, muss das aktive Element zuerst durch Klicken des „Exit“-Knopfs geschlossen werden, bevor der andere Knopf betätigt werden kann.

Die GUI übernimmt neben der Benutzeroberfläche auch einige Setup- und Hintergrundfunktionen. Dazu gehört das Erstellen der .env Datei, in der sensible Daten wie Passwörter gespeichert werden. Zudem mussten alle ipconfig.json Dateien generiert werden, die die IP-Adresse des verwendeten Systems enthalten. Ebenso sollten appsettings.json-Dateien erstellt werden, die weitere sensitive Einstellungen enthalten und ausschließlich lokal verfügbar sein sollen.

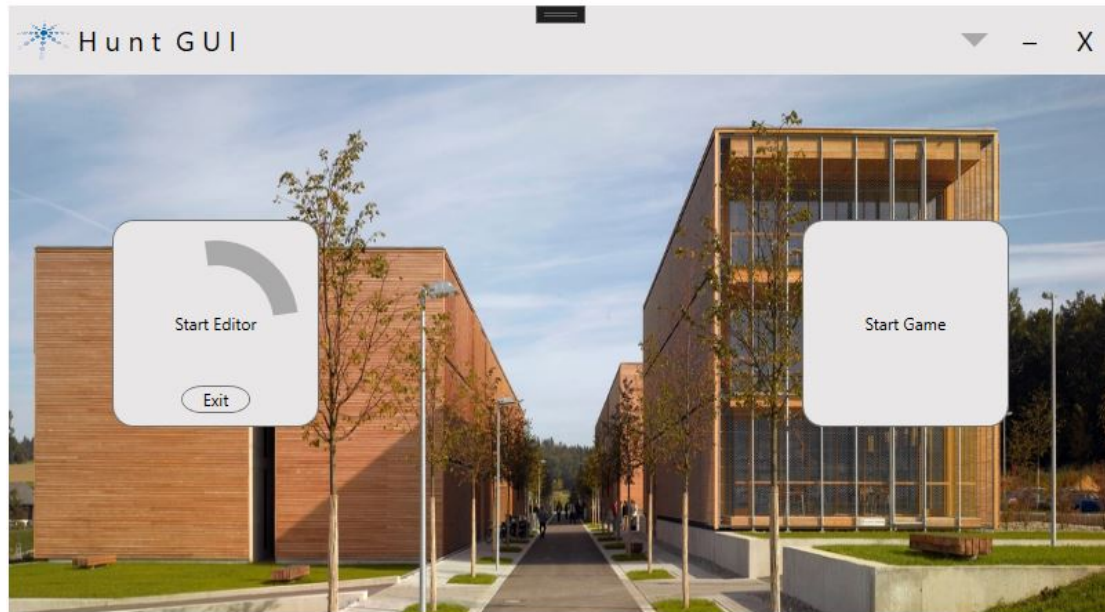


Abbildung 5.4.: Starten des Editors

5.0.2. Aufbau des Hunt-Editor

Im Editor wurden einige Features, die in 4.2.1 beschrieben wurden, implementiert. Es sind jedoch noch einige weitere kleinere Änderungen hinzugekommen, die unter dem Punkt UI-Verbesserung zusammengefasst wurden.

Eine Anforderung war es, im Editor als Hint-Typ zusätzlich Video und Audio hinzuzufügen. Für diese Funktionalität musste im Editor nicht viel verändert werden. Die Logik, um das Video/Audio in einen String zu wandeln, war bereits implementiert und wurde auch hier wiederverwendet. Beim Implementieren sind jedoch einige Fehler aufgefallen. Zum einen wurde festgestellt, dass ein Bild oder ein Video allein möglicherweise nicht aussagekräftig genug ist und daher ein zusätzlicher Erklärungstext erforderlich ist. Der zweite Fehler war, dass es ein Übertragungslimit von etwa 30 MB bei den REST-API-Aufrufen gab. Auf die Lösung dieses Fehlers wird in 5.0.4 genauer eingegangen. Die Lösung des ersten Fehlers war das Hinzufügen eines zusätzlichen Textfeldes für den Bild- oder Video-Hint-Typ.

Abbildung 5.5.: Hinzufügen eines Assignments mit Hint-Typ Video

Wie in 5.5 erkennbar ist, wurde ein extra Textfeld angelegt, in welches man die Zusatzinfo zum Hinweis schreiben kann. Auf dem Bild erkennt man auch, dass ein 'Previous'-Button eingefügt wurde, mit dem man zwischen den einzelnen Erstellungsschritten einer Hunt hin- und herwechseln kann. Hier war es vor allem wichtig, dies konsistent zu gestalten.

ID	HINT TYPE	HINT	ADDITIONAL HINT DATA	SOLUTION TYPE	SOLUTION
1	Video		Usefull Text	Text	dummy
2	Image		undefined	Text	dummy

Abbildung 5.6.: Übersicht einer erstellten Hunt mit allen Assignments im Überblick

Um das konsistente Wechseln zwischen bspw. 5.5 und 5.6 zu ermöglichen, musste die Logik so geändert werden, dass die aktuelle Hunt in den Hunt-Store gespeichert wird, sobald der 'Next'-Button betätigt wurde. Durch diese Änderungen wurde die Hunt korrekt im Store gespeichert, jedoch wurden beim Wechseln nur die eingegebenen Texte in den entsprechenden Feldern angezeigt. Beim Wechseln zu 5.5 wurden Bilder und Videos nicht in dem von Svelte-Flowbite importierten Modul zum Hochladen einer Datei angezeigt. Das Video war als String im Store vorhanden, wurde aber beim Wechsel nicht in ein Video oder Bild umgewandelt und dadurch von der Upload-Komponente nicht angezeigt. Dieser Fehler wurde behoben, indem eine eigene File-Upload-Komponente implementiert wurde. Diese prüft, ob eine Datei im Store vorhanden ist, und zeigt dann nicht die Flowbite-Komponente an, sondern

einen Text, der auf die Anwesenheit der Datei hinweist.

Die Zuweisung der in 5.5 erkennbaren 'Assignment Nr' führte ebenfalls zu einigen Problemen. Wenn eine Hunt in die Datenbank gespeichert wird, werden, da EFC 2.6 verwendet wird, die IDs als Primary Keys für Hint, Solution, Assignment und Hunt zugewiesen. Vor der Änderung wurden im 5.6 Overview einer Hunt diese IDs angezeigt. Bei der Erstellung einer Hunt gab es keine Probleme, da die IDs hier von null an gesetzt wurden. Beim Editieren hingegen traten Probleme auf, da die IDs von EFC übernommen wurden. Dadurch ergab sich beim Editieren einer Hunt das Problem, dass beispielsweise beim Löschen eines Assignments die dadurch frei gewordene ID nicht wieder belegt wurde. EFC verwendet beim Speichern eines neuen Assignments die nächste verfügbare ID, wodurch die Nummerierung der Assignments nicht mehr korrekt war. Dies führte zu einem weiteren Problem: Wenn eine Hunt mit beispielsweise drei Assignments erstellt wurde und anschließend eine neue Hunt angelegt wurde, setzte sich die ID der Assignments der neuen Hunt mit vier fort. Das Problem wurde behoben, indem beim Erstellen und Editieren einer Hunt die IDs der Assignments immer vor der Bearbeitung festgelegt werden. Da EFC die vom Frontend übergebenen IDs nicht berücksichtigt, treten hierbei keine weiteren Fehler auf.

Wenn ein Assignment das erste oder das letzte in der Liste der Assignments einer Hunt ist, werden die entsprechenden Buttons zum Verschieben deaktiviert. Dank der Änderung der Assignment-IDs ist es möglich, ein Assignment zu verschieben oder zu löschen, ohne dass die 'Assignment Nr' falsch ist.

Weitere kleine Änderungen betreffen die Texte am oberen Rand eines Assignments 5.5 sowie den 'Home'-Button 5.7, der nach dem erfolgreichen Erstellen einer Hunt angezeigt wird.

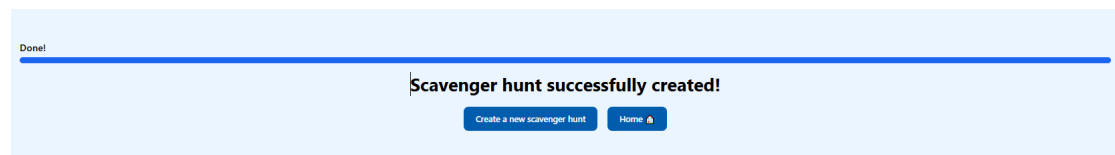


Abbildung 5.7.: Erfolgreiches Erstellen einer Hunt

Der Prozess zum Teilen einer Schnitzeljagd war noch nicht vorhanden, weshalb, wie in 5.8 erkennbar, ein 'Share'-Button eingefügt wurde.

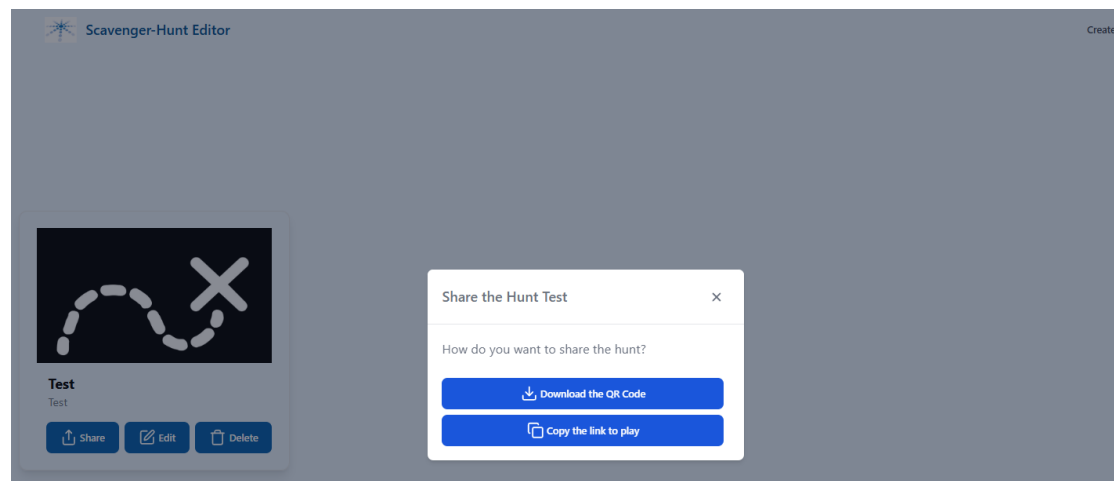


Abbildung 5.8.: Teilen des Links oder QR-Codes einer Schnitzeljagd

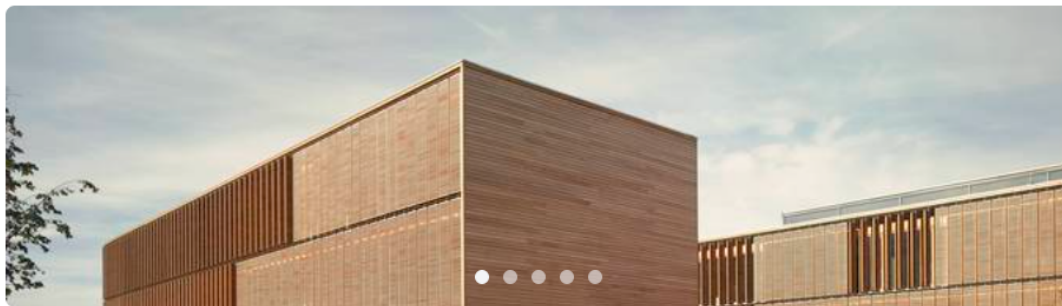
Mithilfe dieses Buttons kann ein QR-Code heruntergeladen werden. Dieser wird aus der IP-Adresse des Netzwerks, dem Port, über den Spieler auf das Spiel zugreifen können, und der Route zur jeweiligen Jagd erstellt. Dieser String wird dann mit dem bereits vorhandenen QR-Code-Generator umgewandelt und heruntergeladen. Wenn man den 'Copy link to play'-Button auswählt, wird der String in die Zwischenablage kopiert und kann dann beispielsweise in eine E-Mail eingefügt werden. Um die IP-Adresse herauszufinden, wird, wenn das Programm über die GUI gestartet wird, die IP-Adresse in einer 'ipconfig.json' gespeichert. Mit einem REST-API-Aufruf kann diese IP-Adresse dann im Editor abgefragt werden.

Die Erstellung des Links stellte sich ebenfalls als Schwierigkeit heraus. Da auf dem Rechner ein Port geöffnet werden muss, stellt dies natürlich eine große Sicherheitslücke dar. Deshalb wurde in Richtung mDNS geforscht. Die mDNS-Adresse ist wie ein Alias, der anstelle einer IP-Adresse verwendet werden kann. Beispielsweise kann die IP-Adresse 192.0.0.1 mit dem Alias Scavhunt.local ersetzt werden. Durch mDNS kann die IP-Adresse dann auch über den Alias angesprochen werden. Ein Systemadministrator kann im Router für bestimmte IP-Adressen solche mDNS-Aliase vergeben.

Eine andere Möglichkeit ist es, den Benutzernamen des Rechners, auf dem das Backend läuft, zu verwenden. Dieser Benutzername kann auch geändert werden. Hier wurde jedoch keine zufriedenstellende Lösung gefunden, weshalb sich der Link weiterhin aus IP:Port/route zusammensetzt.

5.0.3. Aufbau des Hunt-Web-Games

Die Hunt-Web-Game-Ordnerstruktur war zuvor nicht vorhanden. Der Stand war, wie in 3.1 erklärt, dass sich Spieler über einen Link zu einer Hunt registrieren mussten. Anschließend musste ein weiterer Link eingegeben werden, um die registrierte Hunt zu spielen. Dieser Prozess wurde, wie in 4.3 beschrieben, als sehr umständlich empfunden und daher optimiert. Außerdem mussten die im Editor 5.0.2 beschriebenen Änderungen, wie zusätzliche Hint-Typen und zusätzliche Textfelder mit Informationen zu Bildern und Videos, auch im tatsächlichen Spiel angezeigt werden.



The Schnitzeljagd Game

Discover Aalen University in a fun way

Log in here and experience our interactive scavenger hunts - an exciting way to explore the campus and make new friends

Log in now! →]

Project Timeline

February 2024

Project Start

First ideas, wireframing and workflows.

March 2024

Architecting

Architectural design, technological exploration and first insight.

June 2024

Implementation

Start of the implementation.

August 2024

Submission

Abbildung 5.9.: Home-Seite der Hunt-Web-Game-Anwendung

Die Seite 5.9 ist das Home- bzw. Wurzelverzeichnis der Anwendung. Diese Seite wurde aus dem alten Web-Game übernommen und erweitert. Die Punkte im Zeitstrahl wurden um die aktuellen Meilensteine ergänzt, und auch das '+layout.svelte' wurde angepasst.

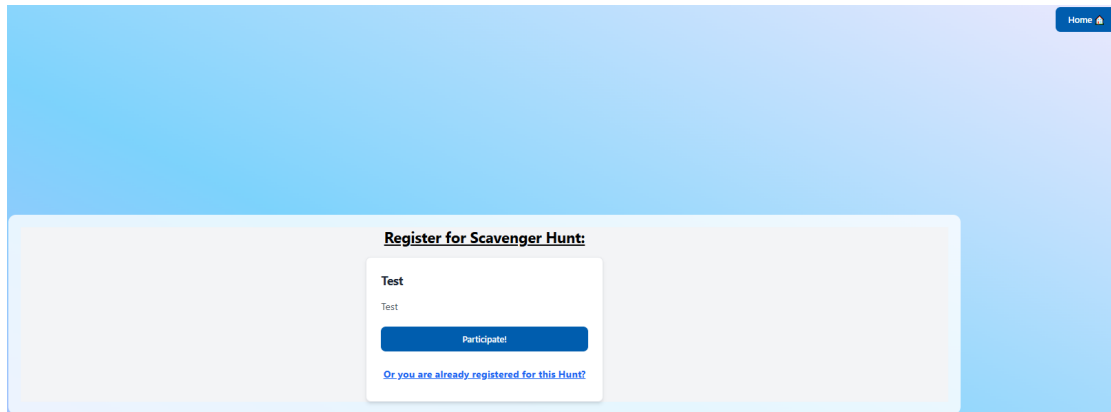


Abbildung 5.10.: Screen zur Registrierung einer Hunt

Wenn ein Link oder QR-Code einer Hunt eingegeben wird, erscheint der Screen 5.10. Im '+layout.svelte' wurden die Farben angepasst sowie ein 'Home'-Button platziert, der zum Home-Screen 5.9 führt. Auf dem 'Registrierungs'-Screen kann man den Namen und die Beschreibung der Hunt sehen sowie sich per Klick auf 'Participate!' für die Hunt registrieren oder sich per Klick auf 'Or you are already registered for this Hunt?' mit einem bestehenden Benutzerkonto anmelden.

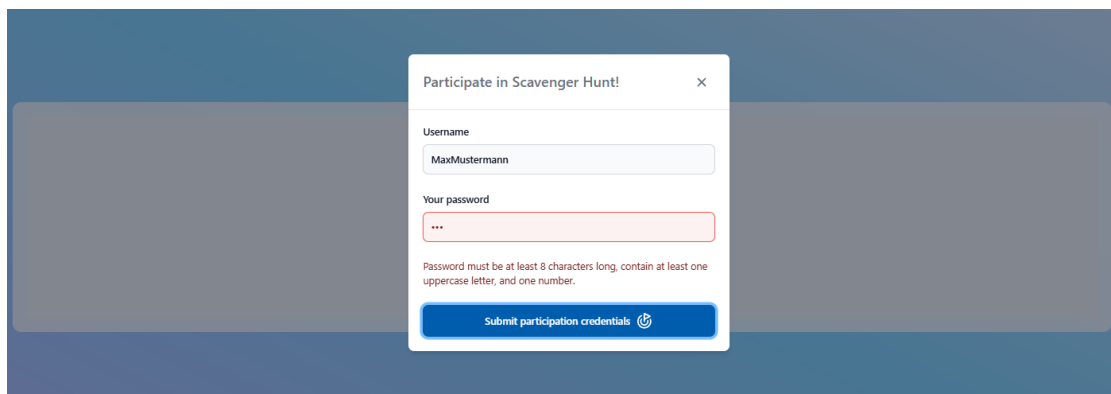


Abbildung 5.11.: Registrier-Pop-up und Eingabe eines unsicheren Passworts

Wenn auf den 'Participate!'-Button geklickt wird, öffnet sich ein Pop-up-Fenster 5.11. Hier kann ein Benutzername und ein Passwort gewählt werden, um sich für die Hunt zu registrieren. Wie auf dem Bild erkennbar, wurde ein Feature zur Auswahl

eines sicheren Passworts implementiert. Das Passwort muss mindestens 8 Zeichen lang sein und einen Großbuchstaben sowie eine Zahl enthalten.

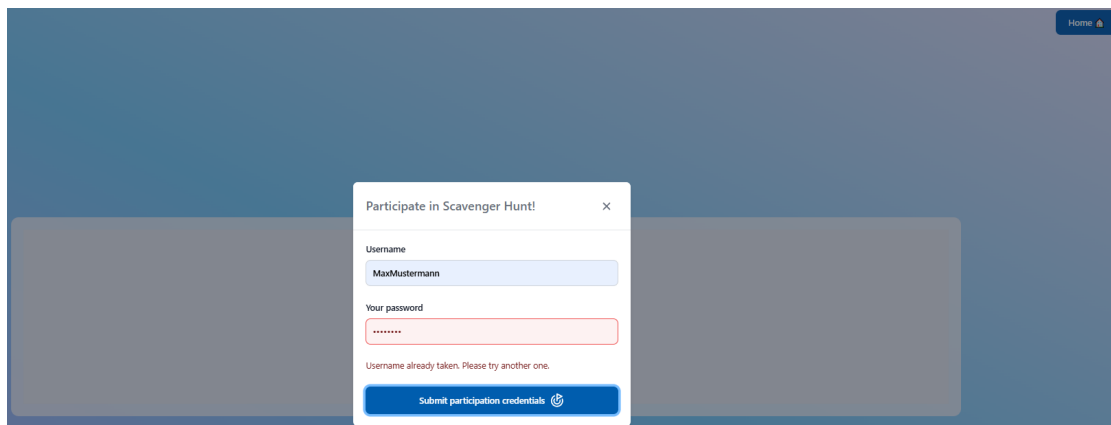


Abbildung 5.12.: Registrierung zu einer Hunt mit schon verwendetem Nutzernamen

Wie in Screen 5.12 ersichtlich, wird auch geprüft, ob der gewählte Benutzername bereits im System existiert. Falls der Benutzername schon belegt ist, wird der Nutzer mit einer Meldung darauf hingewiesen.

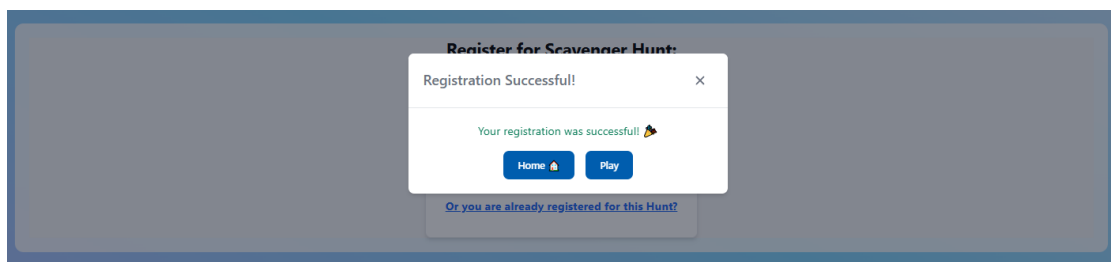


Abbildung 5.13.: Erfolgreiches Anmelden zu einer Schnitzeljagd

Es wurde zusätzliches Feedback beim erfolgreichen Anmelden eines Nutzers eingefügt 5.13. Wenn hier auf 'Play', auf dem Home-Screen 5.9 oder auf 'Or you are already registered for this Hunt?' 5.10 geklickt wird, öffnet sich der Login-Screen 5.14.

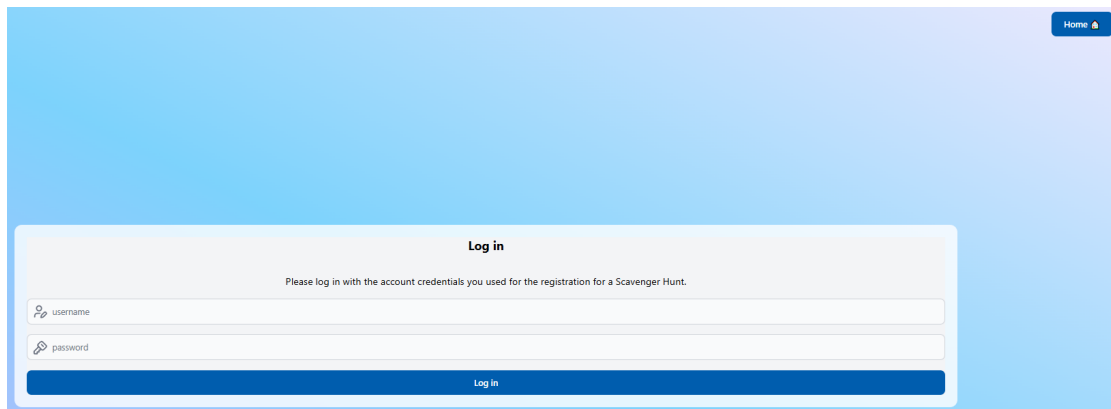


Abbildung 5.14.: Login Screen

Hier kann dann der Nutzernamen und das Passwort eingegeben werden, und anschließend erscheint der Game-Overview-Screen 5.15.

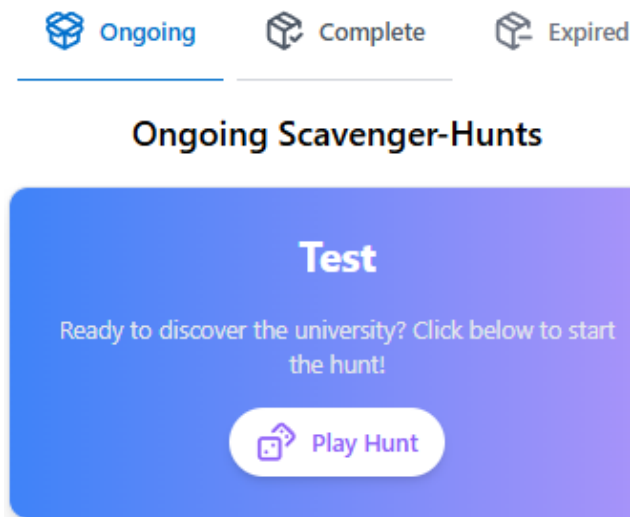


Abbildung 5.15.: Übersicht aller vom Nutzer registrierten Hunts

Hier werden dann alle Hunts aufgelistet, für die sich der Nutzer mit demselben Nutzernamen und Passwort registriert hat. Es gibt drei verschiedene Tabs, die aus dem alten Web-Game übernommen wurden: 'Ongoing'-Hunts, also aktuell noch nicht beendete Hunts, 'Complete'-Hunts, die der Nutzer bereits erfolgreich abgeschlossen hat, und 'Expired'-Hunts, für die sich der Nutzer registriert hat, die aber ihre Time-to-Live überschritten haben. Das Overlay für die Hunt-Cards wurde etwas überarbeitet, sodass es durch Farben ansprechender wirkt.

Wenn auf 'Play Hunt' gedrückt wird, startet die Hunt und das erste Assignment

wird angezeigt.

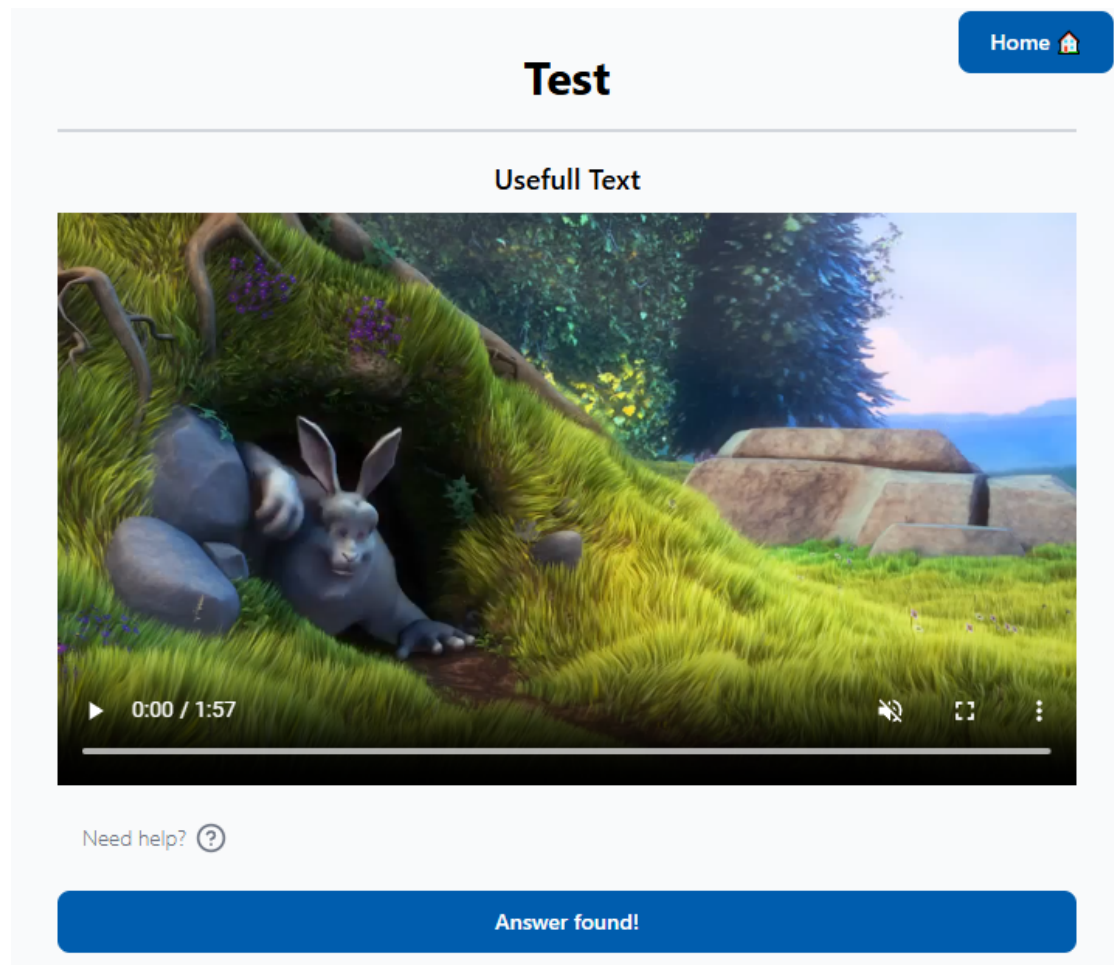


Abbildung 5.16.: Ansicht des Video-Hint-Typs mit zusätzlichen Informationen

Wie auf dem Bild 5.16 zu erkennen ist, wird das Video pausiert und stumm geschaltet angezeigt. Über dem Video befindet sich die Zusatzinformation zum Video. Wenn ein Video oder Bild keine Zusatzinformation besitzt, sieht der Screen wie in Bild 5.17 aus.




Abbildung 5.17.: Ansicht des Bild Hint Typ ohne zusätzlichen Informationen

Wenn auf den 'Answer found!'-Button gedrückt wird, öffnet sich die Solution-Sektion 5.18. Da die Formatierung auf Mobilgeräten teilweise verzerrt sein kann, wurde die Funktionalität implementiert, dass der Screen automatisch zum nächsten Button scrollt, wenn der 'Answer found!'-Button und der 'Submit'-Button gedrückt werden.

Need help? ?

Answer found!

Submit a solution

 Submit Solution

Submit your answer when ready.

clever answer

Submit

You are correct!

Next hint!

Abbildung 5.18.: Solution-Sektion mit richtiger Eingabe

Wenn das letzte Assignment gelöst wird, erscheint der Cheer-Screen 5.19. Auf diesem Screen wurde zusätzlich ein 'Back to Home'-Button eingefügt, mit dem man wieder zur Hunt-Übersicht 5.15 gelangt.

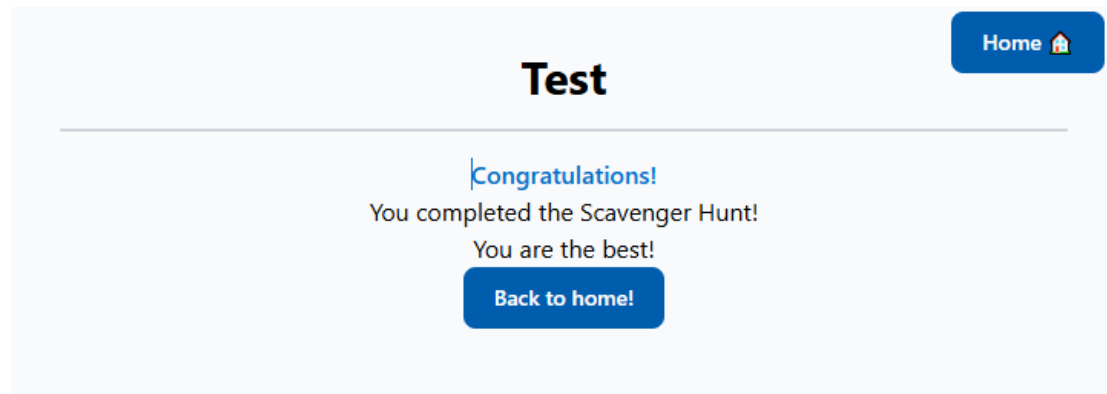


Abbildung 5.19.: Cheer-Screen

Vor der Änderung wurde die dann absolvierte Hunt nicht im 'Complete'-Tab angezeigt. Diese Änderung wurde aber auch implementiert, dass eine absolvierte Hunt sich dann im 'Complete'-Tab 5.20 befindet.

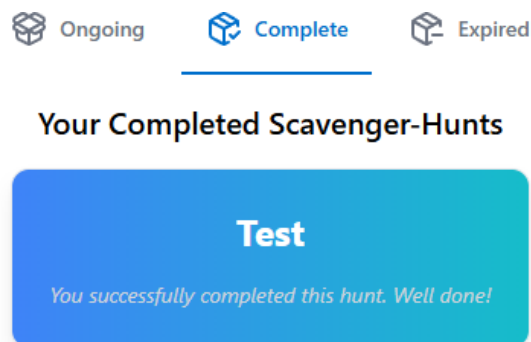


Abbildung 5.20.: Completed Hunt

Auch hier wurde die Farbe und die Nachricht angepasst. Vor der Änderung gab es auch hier noch einen 'Play'-Button, welcher keine Funktion erfüllte. Dieser wurde durch einen Text ersetzt. Zeitlich ausgelaufene Hunts werden im 'Expired'-Tab mit einem nochmal anderen Farbschema angezeigt.

5.0.4. Hunt-Be-API

Damit die in den Absätzen 5.0.2 und 5.0.3 beschriebenen Features funktionieren, mussten im Backend noch einige Dinge angepasst und implementiert werden.

Zum einen wurde der in 5.0.2 erklärte limitierte Upload von Hunts behoben. Dafür wurde in der Konfigurationsdatei, welche den Node-Server aufsetzt, die **client_max_body_size** auf 100 MB erhöht. Damit ist es möglich, Hunts mit vielen Assignments und größeren Dateien zu erstellen.

Außerdem musste für die Funktionalität der zusätzlichen Texte für Bild und Video in der Datenbank eine Zeile hinzugefügt werden. Mit dem Befehl `dotnet ef migrations add AddadditionalDataToHint` kann die zusätzliche Zeile in der Datenbank erstellt werden. Bei Eingabe dieses Befehls wird eine neue, automatisch generierte Migrationsdatei erstellt. Diese Migrationsdatei wird beim Erstellen und Aufsetzen der Docker-Container dazu verwendet, die Datenbank zu generieren.

Es wurden noch einige kleinere Methoden im Backend geschrieben. Eine davon wird zum Befüllen der ipconfig Datei im Ordner Hund-API und Participant-API verwendet. Dazu wird beim Starten der GUI die verfügbaren IP-Adressen aus dem Netz ausgelesen, dann werden diese angepingt. Das heißt es wird geprüft, ob zu diesen Adressen eine Verbindung aufgebaut werden kann und anschließend wird eine erreichbare Adresse in die ipconfig-File gespeichert.

Außerdem wurde eine kleine fetch Methode geschrieben, um aus dem Frontend diese IP-Adresse aus der ipconfig Datei auszulesen. Der erste Ansatz dieser Funktion war, dass während der Laufzeit IP-Adressen abgefragt werden. Da die Anwendung aber in Containern läuft, gab es dabei einige Probleme und es wurde auf die ipconfig beim Erstellen der GUI zurückgegriffen.

6. Inbetriebnahme

Die Inbetriebnahme des Projekts war komplex, da viele Komponenten aufeinandertreffen und miteinander interagieren. Da es sich um eine Erweiterung eines Vorgängerprojekts handelt, konnten wir auf die bestehende Grundlage zurückgreifen und eine solide Basis für die Inbetriebnahme schaffen.

Dies bedeutete, dass nicht alle Komponenten vollständig neu getestet werden mussten, da viele Funktionen aus dem Vorprojekt unverändert übernommen wurden. Besonders auf der Backend-Seite wurden in vielen Bereichen lediglich einzelne Methoden hinzugefügt, ohne das gesamte System zu überarbeiten. Dies hatte sowohl Vor- als auch Nachteile:

- **Vorteil:** Bestehende Module, die nicht verändert wurden, mussten nicht separat erneut getestet werden.
- **Nachteil:** Neue Module oder geänderte Logiken erforderten umfassende Tests. Dabei wurden sowohl die logische Korrektheit und Funktionalität einzelner Methoden und Module geprüft als auch Integrationstests durchgeführt, um sicherzustellen, dass die neuen Komponenten korrekt mit dem restlichen System interagieren.

Gänzlich neue Module, wie die Hunt-GUI, wurden ausgiebig getestet, wobei sich die GUI als wenig komplex erwies. Da sie nur grundlegende Elemente enthält und eine geringe Nutzerinteraktion erfordert, gestalteten sich die Tests relativ einfach. Die größte Herausforderung bestand in der Logik zum Öffnen und Schließen der einzelnen Backend-CMD-Konsolen sowie in dem Ausgiebigen testen des Ladekreises, der umfangreicher gestaltet wurde. Ausführlich getestet wurde der Editor und die Web-App auch im Bezug auf das Testen auf Mobil Geräten auf welchen die Web-App hauptsächlich ausgeführt wird.

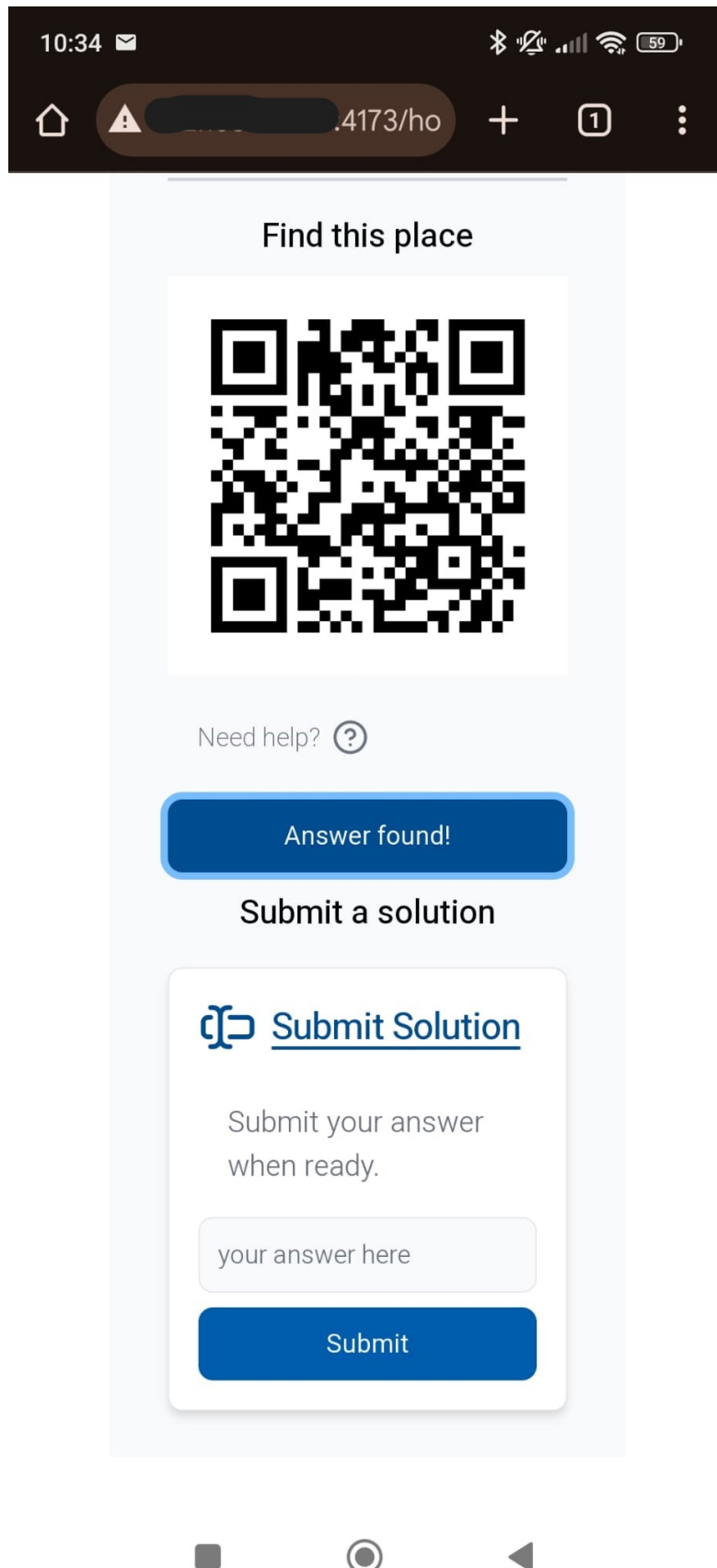


Abbildung 6.1.: Web-Game auf Ansicht auf dem Handy

Das Bild 6.1 ist die Ansicht auf einem Handy(Poco P20). Hier wurde vor allem darauf geachtet dass die Formatierung der verschiedenen Texte, Buttons und Modals nicht übereinander clippen. Außerdem wurde auch getestet ob die neu implementierten Features einwandfrei Funktionieren.

Der Editor ist als Desktopanwendung designet. Trotzdem wurde getestet ob und wie der Editor auf einem kleineren Bildschirm dargestellt wird um mögliche integration für Mobil Geräte für den Editor zu ermöglichen. Getestet und Inbetrieb genommen wurde der Editor indem Verschiedene Hunts erstellt wurden. Es wurden zum Beispiel Hunts mit allen Hint und Solution Typen erstellt, extrem große und extrem viele verschiedene Files hochgeladen und extrem lange Texte eingegeben.

Der Installer wurde auf verschiedenen Geräten in verschiedenen Netzwerken getestet. Es wurde auf verschiedensten Geräten Hunts erstellt und gespielt.

Was natürlich noch Fehlt ist der Stresstest also wie das System auf größere Gruppen von Benutzern reagiert. Diese Testgruppen könnten dann auch nochmal Verbesserungsvorschläge liefern und man könnte eventuell schwer belastete Komponenten verbessern.

Zusätzlich wurde ein Dokument zum erstellen einer Beispielschnitzeljagt für Erstsemesterstudenten der Elektrotechnik und Informatik Fakultät erstellt. Diesem Dokument beigelegt sind Dateien welche zum erstellen verwendet werden sollen.

Es wurde zusätzlich noch ein Benutzerhandbuch und eine Spielanleitung geschrieben um die Nutzung der Anwendung zu vereinfachen. Zusätzlich wurde noch ein Installationsguide geschrieben welcher ausführlich erklärt wie der Installer zu verwenden ist und welche Dinge bei der Installation zu beachten sind.

Der gesamte Source Code wurde mithilfe von AI mit Kommentaren versehen. Diese Kommentare wurden manuel auf fehler und richtigkeit geprüft und verbessert.

6.0.1. InstallForge

Für die Erstellung eines Installers für unser Projekt habe wir das Tool InstallForge verwendet. InstallForge ist ein benutzerfreundliches Programm zur Erstellung von Installationspaketen für Windows-Anwendungen. Der erste Schritt war, InstallForge herunterzuladen und zu installieren, was schnell und unkompliziert war.

Anschließend wurde ein neues Projekt in InstallForge erstellt, in dem die grundlegenden Installationsparameter wie Zielverzeichnis und Programminformationen definiert wurden. Das Tool ermöglichte es, alle relevanten Dateien und Ordner, die für die Ausführung des Projekts notwendig sind, hinzuzufügen, einschließlich der ausführbaren Dateien und Konfigurationsdateien.

Ein wesentlicher Bestandteil der Installation war das Erstellen eines Ordners Hunt im Verzeichnis ProgramData. Dieser Ordner enthält zwei Unterordner: logfile, der die Datei Logfile.txt für die Speicherung von Log-Daten bereithält, und HSAA_Projektarbeit, in dem das Git-Repository für unser Projekt „Scavenger Hunt“ abgelegt ist. Diese Struktur ermöglicht es, sowohl die Log-Daten zu überwachen als auch die Versionskontrolle für das Projekt bereitzustellen.

Zusätzlich wurde eine Verknüpfung auf dem Desktop erstellt, die den Nutzern schnellen Zugriff auf die Hauptanwendung ermöglicht. Diese Verknüpfung verweist auf die ausführbare Datei des Projekts, sodass der Nutzer das Projekt direkt nach der Installation starten kann.

Besonders nützlich war die Möglichkeit, benutzerdefinierte Aktionen hinzuzufügen, wie etwa das Erstellen der Verknüpfung auf dem Desktop und das Hinzufügen von spezifischen Ordnerstrukturen. Darüber hinaus konnte die Installation so konfiguriert werden, dass alle relevanten Dateien korrekt an die richtige Stelle kopiert und die notwendigen Komponenten bereitgestellt werden.

Insgesamt hat InstallForge die Erstellung eines Installers für unser Projekt erheblich vereinfacht und bot uns eine schnelle, effektive Lösung, unser Produkt für die Endnutzer bereitzustellen.

6.0.2. Systemtest

Für eine umfassende Systemprüfung wurde das Spiel auf insgesamt acht unterschiedlichen Geräten getestet, um eine möglichst hohe Kompatibilität und Stabilität sicherzustellen. Dabei kamen zwei Laptops, drei Desktop-PCs, zwei Smartphones und ein Tablet (iOS) zum Einsatz. Neben der allgemeinen Funktionalität auf verschiedenen Hardwarekonfigurationen wurde auch die Netzwerkkompatibilität untersucht. Dafür erfolgten Tests in vier unterschiedlichen Netzwerken, darunter drei Heimnetzwerke sowie ein öffentliches Netzwerk der Hochschule Aalen, um die Stabilität und Zuverlässigkeit der Verbindung unter verschiedenen Bedingungen zu überprüfen.

Ein zentraler Bestandteil des Systemtests war die Evaluierung des Installationsprozesses mithilfe des mit InstallForge erstellten Installers. Dabei wurde geprüft, ob die Software auf allen Geräten fehlerfrei installiert und ausgeführt werden kann. Zusätzlich wurde die Benutzerfreundlichkeit der grafischen Oberfläche (GUI) getestet, um eine intuitive Bedienung sicherzustellen. Ein weiterer wichtiger Aspekt war die Funktionsprüfung des integrierten Editors, mit dem neue Schnitzeljagden erstellt werden können. Abschließend wurde das Spiel selbst auf PCs, Smartphones und Tablets ausführlich getestet, um sicherzustellen, dass es plattformübergreifend stabil läuft, eine gute Performance bietet und eine benutzerfreundliche Spielerfahrung ermöglicht.

Während der Tests traten zahlreiche Fehler auf, die sowohl den Installationsprozess als auch die allgemeine Funktionalität des Spiels beeinträchtigten. Dazu zählten Probleme bei der Erkennung der Netzwerkkonfiguration, unerwartete Abstürze auf bestimmten Geräten und Darstellungsfehler in der Benutzeroberfläche. Viele dieser Probleme konnten während der Testphase behoben werden, jedoch blieben einige bekannte Fehler bestehen, die in zukünftigen Versionen optimiert werden sollten. Diese wurden größtenteils in Form von Verbesserungsvorschlägen dokumentiert und erhielten daher eine geringere Priorität, da sie nicht unmittelbar die Kernfunktionalität des Spiels beeinträchtigten.

7. Zusammenfassung und Ausblick

7.1. Erreichte Ergebnisse

Die Ziele der Erweiterung des Projekts Scavenger-Hunt sind in Abb. 3.1 dargestellt. Vorab kann festgehalten werden, dass alle wesentlichen Ergebnisse erreicht wurden. Das Vorgängerprojekt war zwar voll funktionsfähig, jedoch nicht optimal auf die Benutzererfahrung (User Experience) ausgelegt. Die Hauptzielsetzung dieser Erweiterung bestand daher in der Verbesserung der Nutzerfreundlichkeit. Im Rahmen dessen haben wir unter anderem folgende Optimierungen vorgenommen:

- **Hunt-GUI:** Es wurde eine GUI mit WPF entwickelt, die den Start des Projekts erheblich vereinfacht. Anstatt die einzelnen Module wie Docker, BE-Hunt-API, FE-Hunt-Editor, FE-Hunt-Participation und FE-Hunt-Game manuell über Konsolenbefehle zu starten, übernimmt die GUI nun das automatische Starten dieser Komponenten.
- **Zusammenführung:** Die Module FE-Hunt-Participation und FE-Hunt-Game wurden in einem Modul zusammengeführt: FE-Hunt-Web-Game, das nun beide Funktionen vereint. Zuvor konnte entweder FE-Hunt-Participation oder FE-Hunt-Game gleichzeitig ausgeführt werden, was das Beitreten und Spielen umständlich machte. Jetzt können beide Funktionen gleichzeitig genutzt werden, sodass Spieler dem Spiel beitreten und direkt losspielen können, ohne auf alle anderen Spieler warten zu müssen, bis diese ebenfalls dem Spiel beigetreten sind.
- **Editor:** Der Editor wurde um zahlreiche 'Quality of Life'-Erweiterungen ergänzt, darunter neue Hinweise wie Video und Ton sowie die Möglichkeit, diese zu kombinieren. Außerdem wurden Funktionen wie das Kopieren des Links oder das Erstellen eines QR-Codes zum Beitreten der Hunt hinzugefügt. Zudem wurden viele kleinere Anpassungen und Bugs behoben, die zu einem besseren Gesamterlebnis beitragen.
- **BE-Hunt:** Das Backend wurde vor allem durch einige kleinere Methoden und Funktionen ergänzt. Darunter zum Beispiel eine RESTful API Methode zum Übertragen der IP. Außerdem wurde in der Hunt-Datenbank noch eine zusätzliche Zeile eingefügt und die Übertraggröße einer Hunt auf 100 MB erhöht.

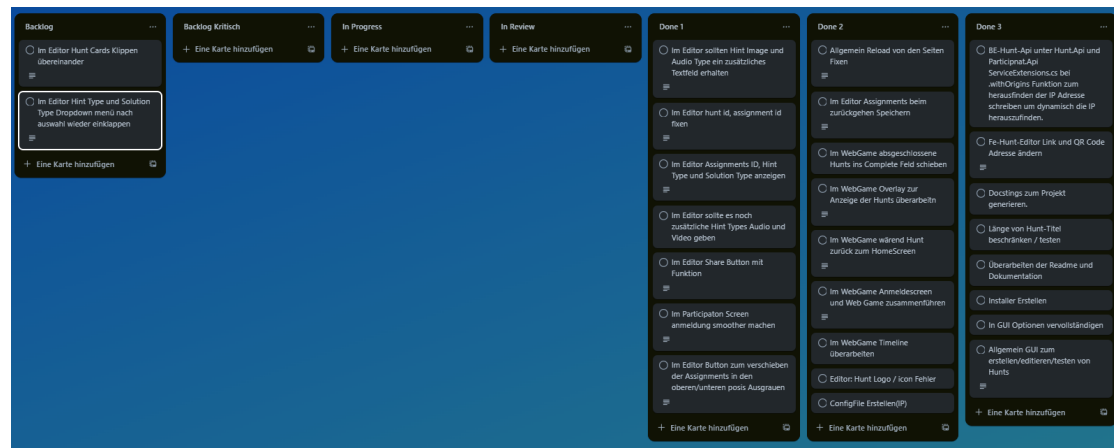


Abbildung 7.1.: Trello Board

Die oben aufgelisteten Punkte stellen nur einen Bruchteil dessen dar, was im Rahmen des Projekts erreicht wurde. Neben den erwähnten Verbesserungen wurden zahlreiche weitere Optimierungen und Erweiterungen vorgenommen, die die Funktionalität und Benutzererfahrung deutlich verbessert haben. Abschließend lässt sich sagen, dass die erwarteten Features vollständig umgesetzt und die ursprünglich gesteckten Ziele erfolgreich erreicht wurden. Das Projekt hat nicht nur die Funktionalität des Vorgängers beibehalten, sondern auch maßgeblich zur Verbesserung der Benutzerfreundlichkeit beigetragen. Es bietet nun eine stabilere, benutzerfreundlichere und flexiblere Lösung für alle Beteiligten.

7.2. Ausblick

Das Projekt ist definitiv noch nicht vollständig ausgefeilt, und es gibt natürlich noch einiges, was geändert und verbessert werden kann. Die folgenden Punkte sind einige Ideen, die während der Arbeit aufgekommen sind.

- Erweiterung und Einbindung von Augmented Reality: Man könnte auf die Kamera des Mobilgeräts zugreifen, dadurch virtuelle Objekte darstellen und die Hunt noch interaktiver gestalten.
- Verbesserung der Sicherheit, insbesondere der Verwaltung von Nutzerdaten: Denkbar wäre eine Verschlüsselung der Passwörter.
- Überarbeitung des Links zum Beitreten einer Hunt: Derzeit werden die IP-Adresse und ein Port angezeigt, was problematisch ist, da dadurch ein Port im Netzwerk offengelegt wird, der für böswillige Angriffe ausgenutzt werden könnte. Eine bessere Lösung wäre die Verwendung eines Aliases oder mDNS anstelle der IP-Adresse.
- Weitere Hint-Typen: Beispielsweise der oben genannte AR-Typ oder komplexere Hinweis-Typen wie Rätsel oder kleine Minispiele.
- Wie im Trello-Board (7.1) ersichtlich, gibt es noch zwei kleinere, unkritische Bugs. Der erste Bug tritt im Editor auf: Wenn mehrere Hunts erstellt wurden und der Entwicklermodus (F12) aktiviert oder das Browserfenster auf einem kleinen Bildschirm geöffnet wird, überlappen sich die Hunt-Karten. Aufgrund von Zeitmangel und der Tatsache, dass dieser Fehler für die Benutzer bzw. Spieler der Anwendung nicht sichtbar ist, hatte er eine niedrige Priorität. Der zweite Punkt ist weniger ein Bug als ein Verbesserungsvorschlag: Wenn im Editor ein "Hintöder SSolutionTyp ausgewählt wird, sollte sich das Dropdown-Menü automatisch wieder einklappen.

Mit diesen Änderungen sollte das Scavenger Hunt Game zu einem noch runderen Erlebnis für die Erstsemesterstudierenden werden und ihnen umso mehr im Gedächtnis bleiben.

7.3. Eigene Meinung und Learnings

Felix Kurz: Ich hatte sehr viel Spaß mit dem Projekt. Am Anfang gab es sehr viel Neues, das gelernt werden musste, vor allem weil wir noch nie etwas mit Docker, RabbitMQ, REST-API und Svelte bzw. JavaScript gearbeitet hatten. Nachdem wir uns jedoch eingearbeitet und den grundlegenden Aufbau verstanden hatten, machte es wirklich Spaß, neue Features zu erstellen und alte zu überarbeiten. Außerdem

habe ich viel mit den oben genannten Tools gearbeitet was mir in meiner kommenden Karriere definitiv nicht schaden wird. Unsere Vorgänger haben uns glücklicherweise eine Einführung und eine kurze Erklärung zum Projekt gegeben, was sehr hilfreich für den Einstieg war.

Trotzdem war es oftmals auch frustrierend da dinge in welche viel Zeit geflossen ist auf den ersten Blick nicht so ersichtlich ist wie ich mir das gewünscht hätte. Beispielsweise wurde viel Zeit in das Erstellen des Links gesteckt und am Ende hat es dann doch nicht so geklappt wie geplant war.

Die Zusammenarbeit im Team hat super geklappt, da Felix immer erreichbar war und man sich dadurch auch spontan absprechen konnte. Um unsere Ziele und Aufgaben nicht aus den Augen zu verlieren, haben wir ein Trello-Board 7.1 verwendet, was auch sehr hilfreich war, um zu sehen, woran der andere gerade arbeitet.

Außerdem gefällt mir, dass das Projekt nicht einfach nur irgendein Projekt ist, das nach der Bearbeitung in einem Ordner verstaubt, sondern etwas ist, womit hoffentlich bei den Erstsemesterstudierenden ein guter Eindruck hinterlassen wird.

Felix Biedenbacher: Die Gruppenarbeit stellte eine spannende Herausforderung dar, insbesondere in Bezug auf das Einarbeiten in neue Technologien, das Verständnis von fremdem Code und die Integration neuen Codes in bestehende Strukturen. Ich habe im Verlauf des Projekts viel gelernt, insbesondere bei der Arbeit mit Docker sowie Svelte und JavaScript. Diese Erfahrungen sind für mein zukünftiges Berufsleben äußerst wertvoll und werden mir helfen, meine technischen Fähigkeiten weiter auszubauen.

Mein Gruppenmitglied, Felix Kurz, war stets motiviert und jederzeit erreichbar, was die Zusammenarbeit sehr angenehm und produktiv gestaltete. Meine Hauptaufgabe im Projekt war die Entwicklung der GUI. Ich hatte großen Spaß an der Arbeit und freue mich auf eine zukünftige Zusammenarbeit mit Felix Kurz.

Literatur

- [1] Docker. *Docker: Accelerated, Containerized Application Development*. Accessed: 2025-02-18. 2025. URL: <https://www.docker.com/>.
- [2] Node.js Foundation. *Introduction to Node.js*. Accessed: 2025-02-17. 2025. URL: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>.
- [3] InstallForge. *InstallForge - Create Professional Setup Packages*. Abgerufen am 24. Februar 2025. 2024. URL: <https://installforge.net/>.
- [4] Microsoft. *Entity Framework Core Documentation*. Accessed: 2025-02-17. 2025. URL: <https://learn.microsoft.com/en-us/ef/core/>.
- [5] Microsoft Docs. *Einführung in WPF in Visual Studio*. Zugriff am 23. Februar 2025. n.d. URL: <https://learn.microsoft.com/de-de/dotnet/desktop/wpf/getting-started/introduction-to-wpf-in-vs?view=netframeworkdesktop-4.8>.
- [6] Microsoft Docs. *Tour durch C#*. Zugriff am 23. Februar 2025. n.d. URL: <https://learn.microsoft.com/de-de/dotnet/csharp/tour-of-csharp/>.
- [7] Plato. *Game and Knowledge*. Beim Spiel kann man einen Menschen in einer Stunde besser kennenlernen als im Gespräch in einem Jahr.
- [8] Flowbite Team. *Flowbite Svelte - UI-Komponentenbibliothek*. <https://flowbite-svelte.com/>. Online; zuletzt abgerufen am 16. Februar 2025. 2025.
- [9] Wikipedia-Autoren. *Svelte (Framework)*. [https://de.wikipedia.org/wiki/Svelte_\(Framework\)](https://de.wikipedia.org/wiki/Svelte_(Framework)). Online; zuletzt abgerufen am 16. Februar 2025. 2025.
- [10] Wikipedia-Benutzer. *Model View Controller*. Abgerufen am 20. Februar 2025. 2024. URL: https://de.wikipedia.org/wiki/Model_View_Controller.
- [11] Wikipedia-Benutzer. *Model View ViewModel*. Abgerufen am 20. Februar 2025. 2024. URL: https://de.wikipedia.org/wiki/Model_View_ViewModel.

A. Architektonische Entscheidungen

Da das Projekt eine Erweiterung eines Vorgängerprojektes war, hatten wir wenig Einfluss auf die Architektur dieses Projektes.

A.1. Architektur Hunt-GUI

Die Entscheidung für die Technologie der grafischen Benutzeroberfläche (GUI) wurde nach einer sorgfältigen Abwägung verschiedener Optionen getroffen. Nach einer Analyse der Anforderungen und möglicher Alternativen entschieden wir uns für Windows Presentation Foundation (WPF). Diese Wahl basierte auf mehreren Faktoren.

Zunächst bietet WPF eine einfache und flexible Lösung zur Erstellung von Desktop-Anwendungen unter Windows. Durch die Trennung von Logik (C#) und Darstellung (XAML) ermöglicht WPF eine saubere, modulare Entwicklung, was insbesondere bei zukünftigen Erweiterungen oder Anpassungen von Vorteil ist. Zudem erlaubt es eine komfortable Implementierung moderner UI-Konzepte, darunter Datenbindung, MVVM-Architektur (Model-View-ViewModel) und eine umfassende Unterstützung für Styles und Templates.

Ein weiterer ausschlaggebender Punkt war die bereits vorhandene Erfahrung innerhalb unseres Teams. Da einige Mitglieder bereits mit WPF gearbeitet hatten, konnte die Einarbeitungszeit minimal gehalten werden. Dies ermöglichte es uns, von Anfang an produktiv zu arbeiten, anstatt uns erst in eine neue Technologie einarbeiten zu müssen.

Abschließend lässt sich sagen, dass die Entscheidung für WPF in diesem Projekt keine tiefgreifenden Auswirkungen hatte, da die GUI als eigenständiges Modul konzipiert wurde. Das bedeutet, dass sie unabhängig von anderen Systemkomponenten entwickelt wurde und keine komplexe Integration mit dem Backend erforderte. Die Kommunikation zwischen GUI und Backend erfolgt eindimensional, was bedeutet, dass das Backend nie auf die GUI zugreift. Dadurch wäre es in der Zukunft problemlos möglich, WPF durch eine andere Technologie, beispielsweise eine Web-basierte Lösung mit Electron oder Blazor zu ersetzen.

Zusammenfassend stellte sich WPF als eine sinnvolle, pragmatische und zukunftssi-

chere Wahl heraus. Es ermöglichte eine schnelle und unkomplizierte Entwicklung, eine flexible Gestaltung der Benutzeroberfläche sowie eine einfache Wartung und Erweiterung für die Zukunft.

A.2. Architektur Hunt-Web-Game

Kontext: Der Login-Vorgang sollte einfacher gestaltet werden. Dafür gab es verschiedene Ansätze, die in Betracht gezogen wurden. Bereits vorhanden waren zwei verschiedene Login-Modals: eines zur Registrierung für eine Hunt und ein weiteres zur Anmeldung.

Entscheidungen: Es wurde entschieden, ob und wie der Registrierungs- bzw. Anmeldeprozess verändert werden muss und ob die vorhandene Ordnerstruktur sowie bestehende Anwendungen beibehalten oder überarbeitet werden sollten.

Auswahl: Letztendlich fiel die Entscheidung auf eine neue Ordnerstruktur mit vorhandenen Routen. Dies brachte folgende Vorteile mit sich:

- Klarere Struktur, da es nur einen Einstiegspunkt für die Nutzer der Hunt gibt.
- Einfachere Handhabung, da Nutzer nicht mehr mehrere Links eingeben müssen.

Nachteile dieser Entscheidung waren:

- Zusammenführen einiger Komponenten, wodurch erstellte Seiten, die nicht mehr benötigt wurden, entfernt wurden.

Konsequenzen: Die Konsequenzen dieser Entscheidung waren die Zusammenführung und das Wegfallen der Ordner/Anwendungen Hunt-Game und Hunt-Participation. Die Routen und Seiten der Ordner wurden zusammengeführt. Eine Seite, auf der man die Anzahl der aktuellen Spieler sehen konnte, wurde entfernt. Diese Statistik könnte jedoch später noch auf der neuen Hauptseite eingefügt werden.

B. Nutzung von Generativer KI-Systeme

Dieses Projekt wurde im Rahmen einer Projektarbeit an der Hochschule Aalen mit Hilfe von generativen KI-Systemen erstellt. Generative KI-Systeme kamen bei der Dokumentation für Rechtschreibung und Satzbau zu Hilfe, wie auch bei der Generation der Docstringe in den Source Code Files. Diese bilden die einzigen Vorkommnisse von generativen KI-Systemen. Alle anderen erschaffenen Lösungen in diesem Projekt wurden vollständig und eigenständig umgesetzt.

C. Anhang A

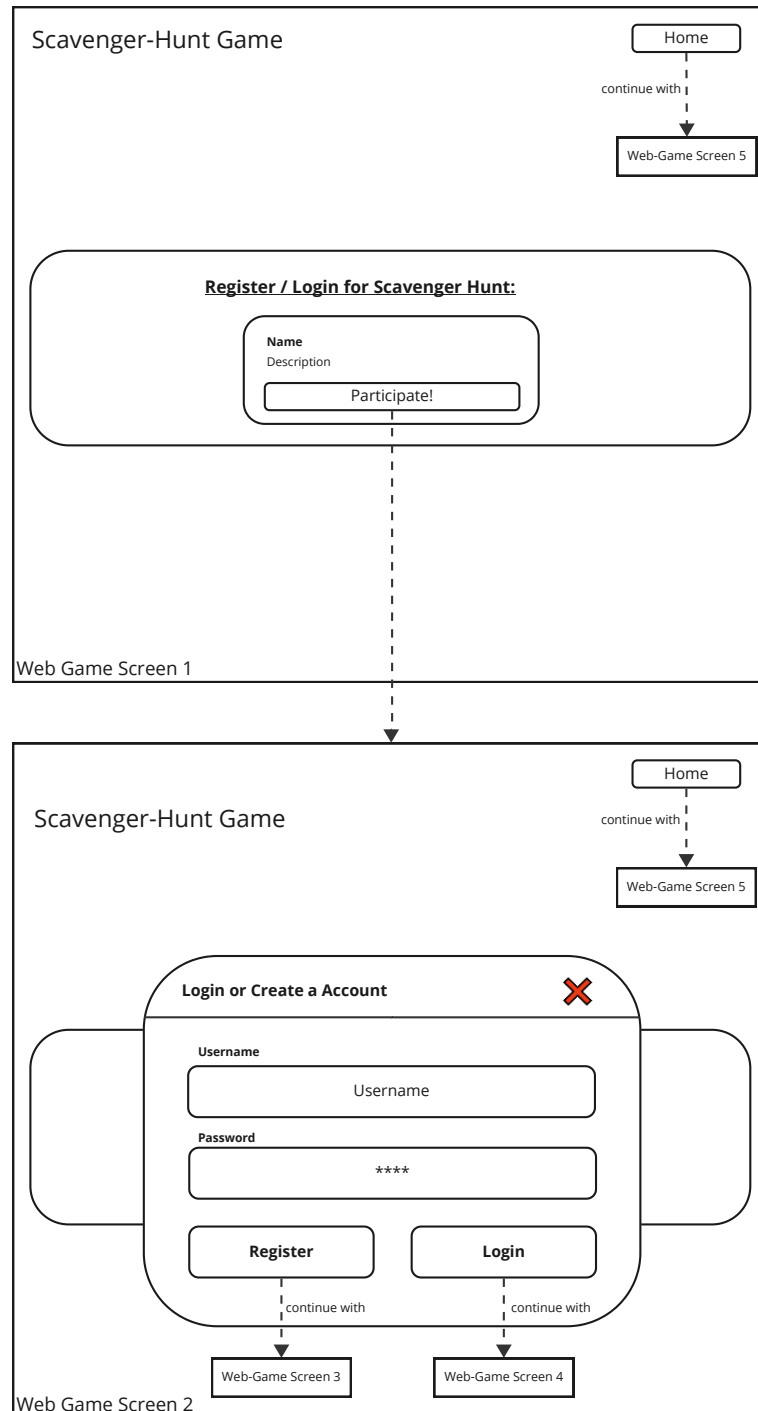
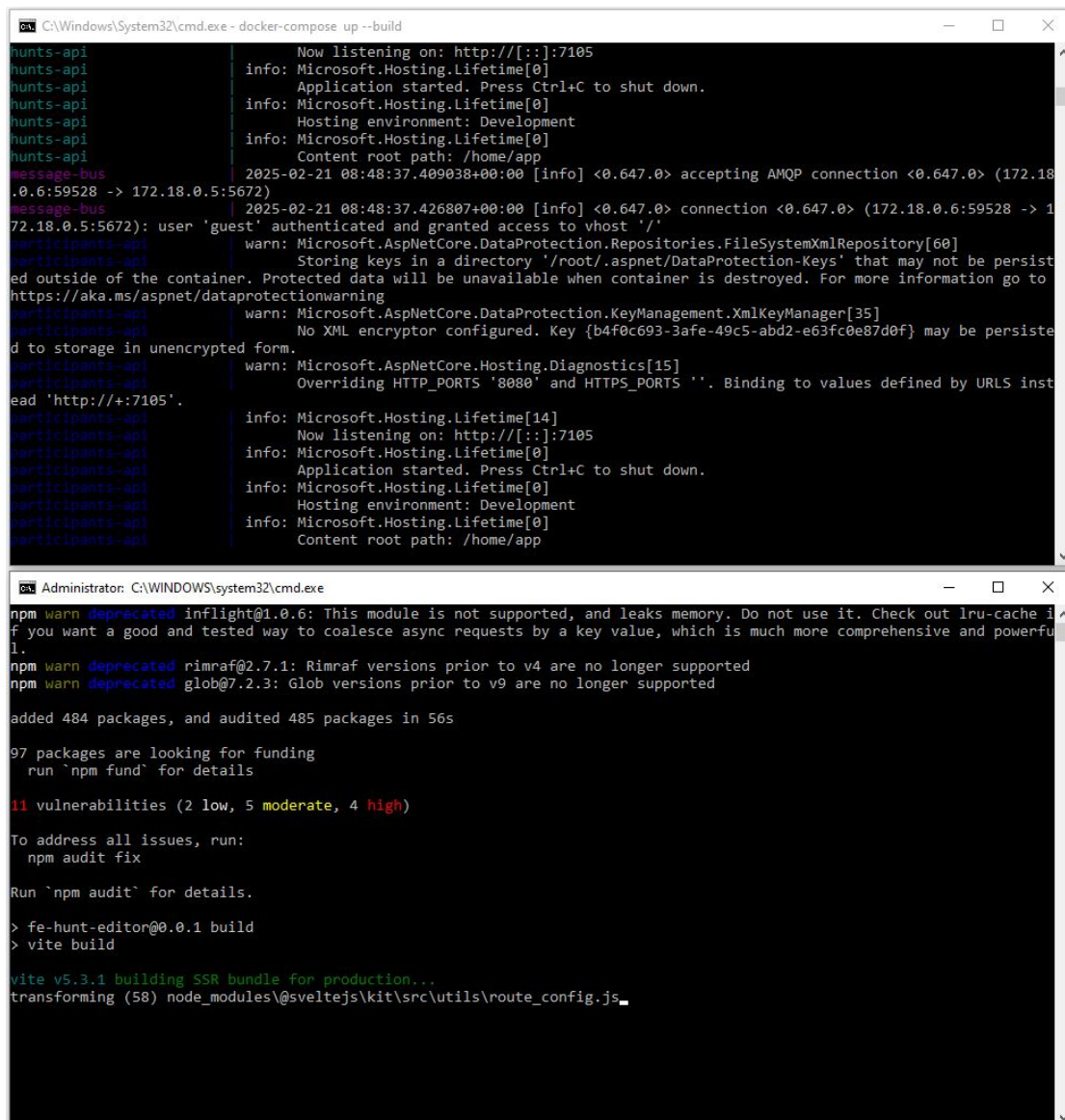


Abbildung C.1.: Hunt Web Game Wireframe vom Login Prozess



The image shows two terminal windows. The top window is titled 'C:\Windows\System32\cmd.exe - docker-compose up --build' and displays the output of a Docker Compose command. It shows logs for 'hunts-api' and 'message-bus' containers, including startup messages and AMQP connection details. The bottom window is titled 'Administrator: C:\WINDOWS\system32\cmd.exe' and shows the output of an npm audit command, displaying warnings about deprecated modules and a list of vulnerabilities.

```

C:\Windows\System32\cmd.exe - docker-compose up --build
hunts-api      Now listening on: http://[::]:7105
hunts-api      info: Microsoft.Hosting.Lifetime[0]
hunts-api      Application started. Press Ctrl+C to shut down.
hunts-api      info: Microsoft.Hosting.Lifetime[0]
hunts-api      Hosting environment: Development
hunts-api      info: Microsoft.Hosting.Lifetime[0]
hunts-api      Content root path: /home/app
message-bus     2025-02-21 08:48:37.409038+00:00 [info] <0.647.0> accepting AMQP connection <0.647.0> (172.18
.0.6:59528 -> 172.18.0.5:5672)
message-bus     2025-02-21 08:48:37.426807+00:00 [info] <0.647.0> connection <0.647.0> (172.18.0.6:59528 -> 1
72.18.0.5:5672): user 'guest' authenticated and granted access to vhost '/'
participants-api warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
participants-api Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persist
ed outside of the container. Protected data will be unavailable when container is destroyed. For more information go to
https://aka.ms/aspnet/dataprotectionwarning
participants-api warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
participants-api No XML encryptor configured. Key {b4f0c693-3afe-49c5-abd2-e63fc0e87d0f} may be persiste
d to storage in unencrypted form.
participants-api warn: Microsoft.AspNetCore.Hosting.Diagnostics[15]
participants-api Overriding HTTP_PORTS '8080' and HTTPS_PORTS ''. Binding to values defined by URLs inst
ead 'http://+:7105'.
participants-api info: Microsoft.Hosting.Lifetime[14]
participants-api Now listening on: http://[::]:7105
participants-api info: Microsoft.Hosting.Lifetime[0]
participants-api Application started. Press Ctrl+C to shut down.
participants-api info: Microsoft.Hosting.Lifetime[0]
participants-api Hosting environment: Development
participants-api info: Microsoft.Hosting.Lifetime[0]
participants-api Content root path: /home/app

Administrator: C:\WINDOWS\system32\cmd.exe
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache i
f you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerfu
l.
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
added 484 packages, and audited 485 packages in 56s
97 packages are looking for funding
  run `npm fund` for details
11 vulnerabilities (2 low, 5 moderate, 4 high)
To address all issues, run:
  npm audit fix
Run `npm audit` for details.
> fe-hunt-editor@0.0.1 build
> vite build
vite v5.3.1 building SSR bundle for production...
transforming (58) node_modules/@sveltejs/kit/src/utils/route_config.js_

```

Abbildung C.2.: CMD Konsolen für docker-compose und Editor