BattleSprout

Web-Anwendungsentwicklung Sommersemester 2023

Petrov Ilia   
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

Klinger Fabian  
line 2: *dept. name of organization*  
*(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: Kasseckert Tassilo  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID Schmidt Fabian  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

Stricker Natalie  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

Kietzer Rebecca  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

Barbee Montell  
*Medieninformatik*   
m.barbee@oth-aw.de

*Abstract*— Beschreibung der Software-Architektur für das Projekt BattleSprout von Team Rot. Dabei wird ¨ zunächst allgemein ein Überblick über das System gegeben, Architekturentscheidungen erklärt und auf Probleme bei der Realisierung eingegangen. Anschließend werden noch zukünftige Ausbaumöglichkeiten der Anwendung aufgelistet.

# Einleitung

## Mission Statement

Bei BattleSprout handelt es sich um eine Web-Anwendung, die es Spielern erlaubt, eine Variation des bekannten Brettspiels Battleships zu spielen.

Der Fokus liegt hier auf der Kombination von Wettbewerb und Kooperation. Es geht darum ein Strategiespiel zu erstellen, bei dem der Kern auf Zusammenarbeit liegt, jedoch trotzdem das Wettbewerbselement erhalten bleibt.

Dadurch soll beobachtet werden, ob sich die Spielermentalität ändert, wenn das Ziel ist, dem anderen Spieler zu helfen und nicht wie bei Battleships die gegnerische Flotte zerstören zu müssen.

## Spielregeln und Spielmechanik

Spieler müssen in BattleSprout strategisch vorgehen, um den Garten des Herausforderers zuerst erblühen zu lassen.

Vor dem Beginn des eigentlichen Spiels müssen die Spieler ihre Pflanzen auf einem Raster platzieren. Jeder Spieler besitzt ein eigenes Raster, auf dem er Pflanzen platzieren kann. Dazu kommen Wasserbomben, die abwechselnd von den Spielern auf die Pflanzen des Herausforderers geworfen werden müssen. Trifft man dabei eine Pflanze, so blüht diese auf. Der Spieler, der zuerst alle Pflanzen des anderen erblühen lässt, gewinnt das Spiel.

## Kontextabgrenzung

Die Darstellung zeigt das System als Blackbox und zeigt die Fremdkomponente auf und Interaktion mit dem BenutzerEin Bild, das Kreis, Screenshot, Diagramm, Astronomie enthält.

Automatisch generierte Beschreibung

## Architekturziele

Das System soll folgende Architekturziele berücksichtigen:

* Zuverlässigkeit
  + Der Nutzer sollte eine Reibungslose Spielerfahrung haben
  + Die Verarbeitung der Spielaktionen und die Aktualisierung der Spielzustände müssen zuverlässig sein, um Spielausfälle zu vermeiden
* Benutzbarkeit
  + Die Anwendung soll für eine breite Gruppe von Nutzer intuitiv navigierbar sein
* Echtzeitkommunikation
  + Schnelle Aktualisierung des Spielstatus und Darstellung der Spieleaktionen

# Lösungsstrategie

## Systemüberblick

BattleSprout verwendet den MERN-Stack mit Typescript als Programmiersprache um die Vorteile der statischen Typisierung zu nutzen. Zusätzlich wird Socket.IO integriert, um eine Echtzeitkommunikation zwischen Client und Server zu ermöglichen.

Ein Bild, das Text, Screenshot, Schrift, Grafikdesign enthält.

Automatisch generierte Beschreibung

## Architektur Frontend

Im Frontend wird React mit Programmiersprache Typescript verwendet. Durch die wiederverwendbaren Komponenten bei der Entwicklung mit React kann man verschieden Elemente wie das Spielbrett, die Freundesliste oder den Login eigenständig behandeln. Dies ist vorteilhaft bei der Entwicklung mit mehreren Personen, da man so recht unabhängig voneinander arbeiten kann.

Bei Spielen ist es wichtig, den derzeitigen Zustand eines Spiels im Auge zu behalten und bei Änderungen zu aktualisieren und die effiziente Zustandsverwaltung mit React funktioniert hier sehr gut.

Typescript ermöglicht es Funktionen und Schnittstellen klar zu definieren und somit mögliche Verwirrungen bei der Eingabe und Ausgabe der Typen frühzeitig zu verhindern.

Die Hauptelemente der Architektur im Frontend teilen sich folgendermaßen auf:

### Assets: Hier befinden sich die nötigen Bilder und Grafiken zur ansprechenderen Darstellung der Anwendung. Darunter fallen zum Beispiel das Hintergrundbild der Gesamtanwendung und verschiedene Pflanzenarten, um das visuelle Erlebnis der Nutzer zu verbessern.

### CreateGame: In der CreateGame-Komponente können Benutzer ein neues Spiel erstellen. Dazu können sie Einstellungen wie zum Beispiel die Größe des Spielbretts wählen und das Spiel nach ihren Vorlieben anpassen.

### FriendList: Die FriendList-Komponente ist im Layout der Seite eingebaut und zu jederzeit für den User über ein Burgermenü in der linken oberen Seite der Anwendung Zugreifbar. Hier können Freunde hinzugefügt, entfernt und zu einem Spiel einem Spiel eingeladen werden.

### GameField: Das GameField stellt den Hauptbereich der Anwendung dar. Es enthält das Spielbrett, auf dem die Pflanzen platziert werden. Das Spielbrett des gegnerischen Spielers, auf das die Wasserbomben geworfen werden, um die Pflanzen des Gegners zum Blühen zu bringen. Die Spielbretter werden mithilfe eines Gitters realisiert, die aus einzelnen Zellen bestehen. Die Zellen tragen jeweils ihre Position und eine Variable um zu prüfen, ob sie schon bewässert worden sind. Für die Vorbereitungsphase ist außerdem eine Komponente definiert, die dem Spieler die benötigte Zahl der Pflanzen anzeigt, die sie noch platzieren müssen, bevor sie ein Spiel starten können.

### Login und Signup: Diese Login- und Signup-Komponenten erlauben es dem Nutzer, sich in die Anwendung einzuloggen oder ein neues Konto zu erstellen. Es ist wichtig anzumerken, dass nur eingeloggte und autorisierte Nutzer Zugriff auf alle Funktionen haben. Um auf die Hauptfunktionen der Anwendung, wie z.B. das Erstellen eines Spiels, die Freundesliste, das Spielbrett oder die Rangliste zugreifen zu können, muss der Nutzer vorher erfogleich eingeloggt sein.

### Ranking: Die Ranking-Komponente zeigt die Rangliste der Spieler mit den meisten Siegen insgesamt an. Benutzer können sehen, wie sie im Vergleich zu anderen Personen stehen.

## Architektur Backend

Für die Speicherung von Benutzerdaten, der Rangliste und der Freundesliste wurde MongoDB gewählt. MongoDB bietet uns die Flexibilität, Datenstrukturen anzupassen und zu erweitern, was in der Spieleentwicklung besonders wichtig ist. Da sich die Anforderungen im Laufe der Entwicklung ändern können, müssen Datenstrukturen an neue Gegebenheiten anpassbar sein. MongoDB erfüllt diese Anforderung souverän und ermöglicht eine effiziente Datenverwaltung.

Das Node.js basierte Express bietet eine einfache Möglichkeit API-Routen zu implementieren und in Verbindung mit axios, einem HTTP-Client, kann man die Kommunikation zwischen Backend and Frontend vervollständigen. Der Express Server reagiert auf die Anfragen, die mit axios gestellt werden. Während eines Spiels wird für Echtzeitkommunikation Socket.io verwendet. Das Ziel ist es, die Latenz während eines Spiels gering zu halten, um das Spielerlebniss zu verbessern. Socket.io liefert mit der Möglichkeit, Echtzeit bidirektionalen Datenaustausch umsetzen zu können, die perfekte Grundlage, um dieses Ziel zu erreichen.

Das Backend lässt sich in die Folgenden Abschnitte einteilen:

### Socket: Enthält die Implementierung der Echtzeitkommunikation zwischen dem Server und dem Client während eines Spiels. Hier werden die Validierung und die Handshakes durchgeführt, um sicherzustellen, dass nur autorisierte Clients auf die jeweiligen Socket-Verbindungen zugreifen können. Des Weiteren werden hier verschiedene Events definiert, die während eines Spiels auftreten können. Bei „handleSetSplash“ beispielsweise werden der Spieler, der die Aktion ausgeführt hat und die entsprechenden Koordinaten, die dieser gewählt hat in eine Funktion übergeben, die die Logik der Aktion implementiert.

### Models: Der Model Ordner beinhaltet unsere definierten Datenstrukturen wie zum Beispiel einen Nutzer, der eine e-mail, ein Passwort und Freunde besitzt.

### Routes: Enthält die API-Routen der Anwendung für die Kommunikation außerhalb eines Spiels. Dort sind Routen für zum Beispiel die Verwaltung von Freunden, der Registrierten Nutzer und der Rangliste enthalten.

# Probleme bei der Entwicklung

# Zukuntsausblick

Derzeitig bietet BattleSprout eine Möglichkeit eine Variante des Spiels „Battleships“ zu spielen.