



Lost Island

Lost Island

Autor: Sebastian Kull, Nepomuk Lehmann

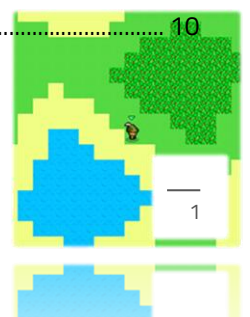
Betreuer: Dimitri Waber

Datum: 13.07.2022

Bbc
Berufs
bildungs
center

Inhalt

1	Abstract	2
2	Anleitung	2
2.1	Backend:	2
2.2	Frontend	3
3	Aufbau	3
3.1	User Stories	3
3.2	Architektur und Technologien	3
3.3	Planung	3
3.3.1	Scrum	3
4	Backend	4
4.1	Datenbank	4
4.1.1	Struktur	4
4.1.2	CRUD Operationen	4
4.2	Socket	5
4.2.1	AddEventListener:	5
4.2.2	Join:	5
4.2.3	onDataRecieved:	5
4.2.4	onProjectileRecieved:	5
4.2.5	onConnected & onDisconnected:	5
4.2.6	isEventListenerForMapAdded:	5
4.3	Perlin Noise Map Generator	6
4.4	UML Diagramm	6
5	Frontend	7
5.1	Zielpublikum	7
5.1.1	Beschreibung	7
5.1.2	Konsequenzen bei der Umsetzung	7
5.2	Mock Up	7
5.3	Sitemap	8
5.4	UML Diagramm	8
5.5	Wichtigste Components	8
5.5.1	Game.js	8
5.5.2	Modals	9
6	Testing	10
6.1	Testumgebung	10
6.2	User Acceptance Testfälle	10



7	Testprotokoll	13
8	Fazit	14
8.1	Sebastian Kull	14
8.2	Nepomuk Lehmann.....	14
9	Impressum	15

1 Abstract

In unserer Webapplikation kann der Benutzer ein Multiplayer Spiel, in seinem Browser, spielen.

Das Spiel besteht aus einer 2D-Welt die frei aus der Vogelperspektive frei erkundbar ist. Die Welt wird mit Perlin-Noise generiert und wird jedes Mal neu berechnet, sobald ein Benutzer die Seite aufruft. Jedes Mal, wenn ein neuer Benutzer auf die Webseite geht, taucht in dem Spiel ein neuer Spieler auf. Das Gegenteil geschieht wenn ein Benutzer das Spiel verlässt. Die Spieler können sich via Sprachchat austauschen. Jedoch hört man nur die nahen gelegenen Spieler. Das Volumen wird der Distanz zwischen den Spieler angepasst. Die Spieler können sich per Nachrichten in einem Chat austauschen.

Daten werden Mittels Socket und http Request's in einer Datenbankgespeichert und ermöglichen den Austausch von Koordinaten und Spielerdaten sowie Audiosignal. Das Backend wird mit Java erstellt. Das Frontend und das Spiel werden in JavaScript geschrieben. Als Framework benutzen wir Springboot und React.

2 Anleitung

Für das Starten dieses Projektes werden folgende Umgebungen benötigt:

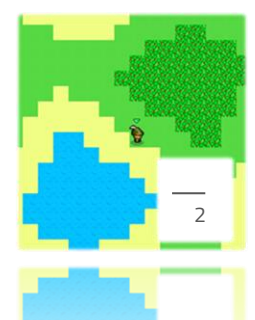
- Node v16.14.2
- SDK 17

2.1 Backend:

- Schritt 1: Öffnen sie das Backend in einer DIE
- Schritt 2: Wählen sie den Port und die IP-Adresse für den Socket Server und der API aus [1]
- Schritt 3: führen sie das «Skript database/create_database.sql» aus
- Schritt 4: builden sie das Projekt
- Schritt 5: Starten sie die Applikation

[1]

- Socket: java/ch/bbcag/lostislandbackend/LostIslandBackendApplication.java
- API : src/main/resources/application.properties



2.2 Frontend

- Schritt 1: Öffnen sie das Frontend in einer IDE
Schritt 2: Wählen sie den Port und die IP-Adresse für die Socket Verbindung und API aus [2]
Schritt 3: Führen sie den Befehl «npm install» oder «npm i» aus
Schritt 4: Führen sie den Befehl «npm next build» aus
Schritt 5: Führen sie den Befehl «npm start» aus

Entwicklungsmodus Modus (Anstatt Schritt 4 und 5)

- Schritt 3: Führen sie den Befehl «npm run dev» aus

[2]

Socket: pages/_app.js

API: lib/api.js

3 Aufbau

3.1 User Stories

- Als Benutzer möchte ich mich registrieren können, damit mein eigenes Profil erstellt wird.
- Als Benutzer möchte ich mich einloggen können, damit ich Zugriff auf das Spiel habe
- Als Benutzer möchte ich mich registrieren können, damit mein eigenes Profil erstellt wird.
- Als Benutzer möchte ich mich einloggen können, damit ich Zugriff auf das Spiel habe
- Als Benutzer möchte ich wissen welcher Spieler mir gehört.
- Als Benutzer möchte ich Informationen über das aktuelle Geschehen sehen.
- Als Benutzer möchte ich mich im Mittelpunkt haben und eine Map erkunden können.
- Als Benutzer möchte ich Nachrichten von anderen Spielern auf der Map sehen.

3.2 Architektur und Technologien

- Backend: Java Spring Boot, IDE: IntelliJ IDEA Ultimate, Datenbank: MySQL
- Frontend: ReactJS/NextJS, IDE: Webstorm

3.3 Planung

3.3.1 Scrum

3.3.1.1 Scrum Master

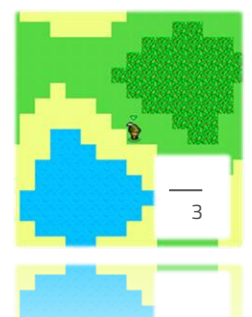
Sebastian Kull

3.3.1.2 Daily Scrum

Jeden Morgen, sobald das Team vollständig ist

3.3.1.3 Definition of Done

- Akzeptanzkriterium erfüllt
- Auf Git commit mit sinnvoller Message
- Dokumentation nachgeführt



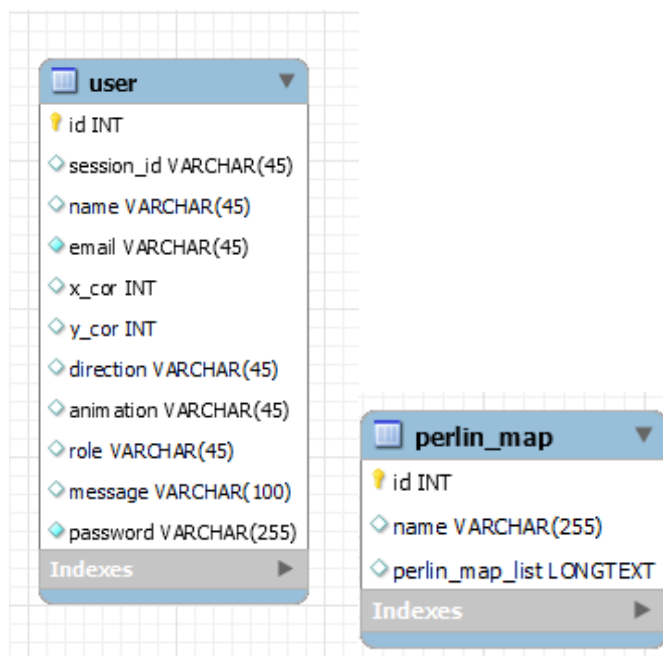
4 Backend

4.1 Datenbank

Damit das Erstellen der Datenbank unkompliziert und schnell realisierbar ist, wird im Backendprojekt im Database Ordner ein entsprechendes SQL-File «CreateDatabase.sql» abgelegt, welches ausgeführt werden kann. Die benötigten Tabellen werden automatisch bei Starten der Applikation erzeugt.

4.1.1 Struktur

Die verwendete Datenbank besteht aus zwei Tabellen: User und Perlin Map



user
id INT
session_id VARCHAR(45)
name VARCHAR(45)
email VARCHAR(45)
x_cor INT
y_cor INT
direction VARCHAR(45)
animation VARCHAR(45)
role VARCHAR(45)
message VARCHAR(100)
password VARCHAR(255)
Indexes

perlin_map
id INT
name VARCHAR(255)
perlin_map_list LONGTEXT
Indexes

4.1.2 CRUD Operationen

Users können durch Registrierungen erstellt werden. Nach erfolgreichem Registrieren kann man sich einloggen und erhält ein Session Token. Mit diesem Token hat man die Möglichkeit folgende geschützte Operationen betätigen:

Der Benutzer erhält eine Liste mit den existierenden Maps.

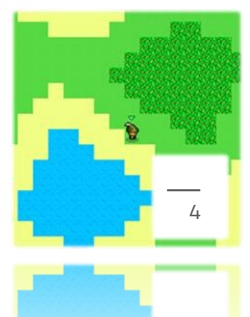
Der Benutzer kann eine Spielwelt erstellen.

Folgende Änderungen sind nicht Spiel relevant. Diese Funktionen sind vorhanden damit die Benutzer Daten gespeichert werden und beim nächsten Einloggen erhältlich sind:

Der Benutzer kann seinen Namen ändern.

Der Benutzer kann seine Nachricht überschreiben.

Der Benutzer kann seine Rolle ändern.



4.2 Socket

Die Echtzeit Übertragung geschieht via Socket. Socket ist eine bidirektionale Verbindung. Man kann mit einer Verbindung Daten senden aber auch empfangen. In diesem Projekt ist im Backend ein Socket Server vorhanden. Mit diesem können sich die Clients verbinden und Daten austauschen. Um die Daten Sprachenunabhängig zu machen, werden diese vor Versand in JSON umgewandelt.

Jeder Benutzer sendet seine Spieler Daten an einen Server. Der Server verarbeitet die Daten, erstellt ein Datenpaket mit all den erhaltenen Daten und schickt sie dann an jeden Spieler weiter. Für jede Karte die aktuell gespielt wird, wird ein eigener Eventlistener erstellt. So ist es möglich verschiedene Spielräume zu haben. Die erhaltenen Daten werden auf eine Klasse gemappt.

Der Socket Server besteht aus folgenden Funktionen:

4.2.1 AddEventListener:

Erstellen eines neuen Spielraumes. Jedem Event Name wird zusätzlich `/id` hinzugefügt. Wobei «id» für die Map Id steht welche in diesem Raum geladen ist.

4.2.2 Join:

Diese Funktion ist zuständig dem Benutzer die neue Session Id zu senden. Zusätzlich werden ihm alle Spieler des Raumes gesendet.

4.2.3 onDataReceived:

Hier werden alle Spieler Daten aufgenommen und dem passenden Raum weitergeleitet.

4.2.4 onProjectileReceived:

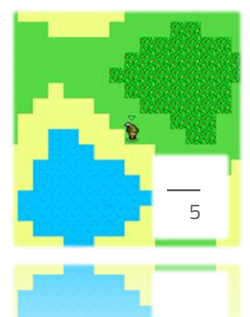
Hier werden alle Daten der Projektile gesammelt und dem passenden Raum weitergeleitet.

4.2.5 onConnected & onDisconnected:

onConnected ermöglicht das Loggen der Benutzer die neu dazugekommen sind. onDisconnected speichert die aktuellen Daten des Benutzers in der Datenbank ab. Zusätzlich wird dieser Spieler aus seinem aktuellen Raum entfernt.

4.2.6 isEventListenerForMapAdded:

Diese Klasse wird benutzt, um zu überprüfen, ob ein Eventlistener schon erstellt worden ist oder nicht. Dies wird überprüft, wenn ein Spieler die Daten einer Spielkarte abrufen.



4.3 Perlin Noise Map Generator

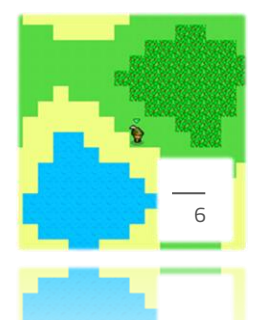
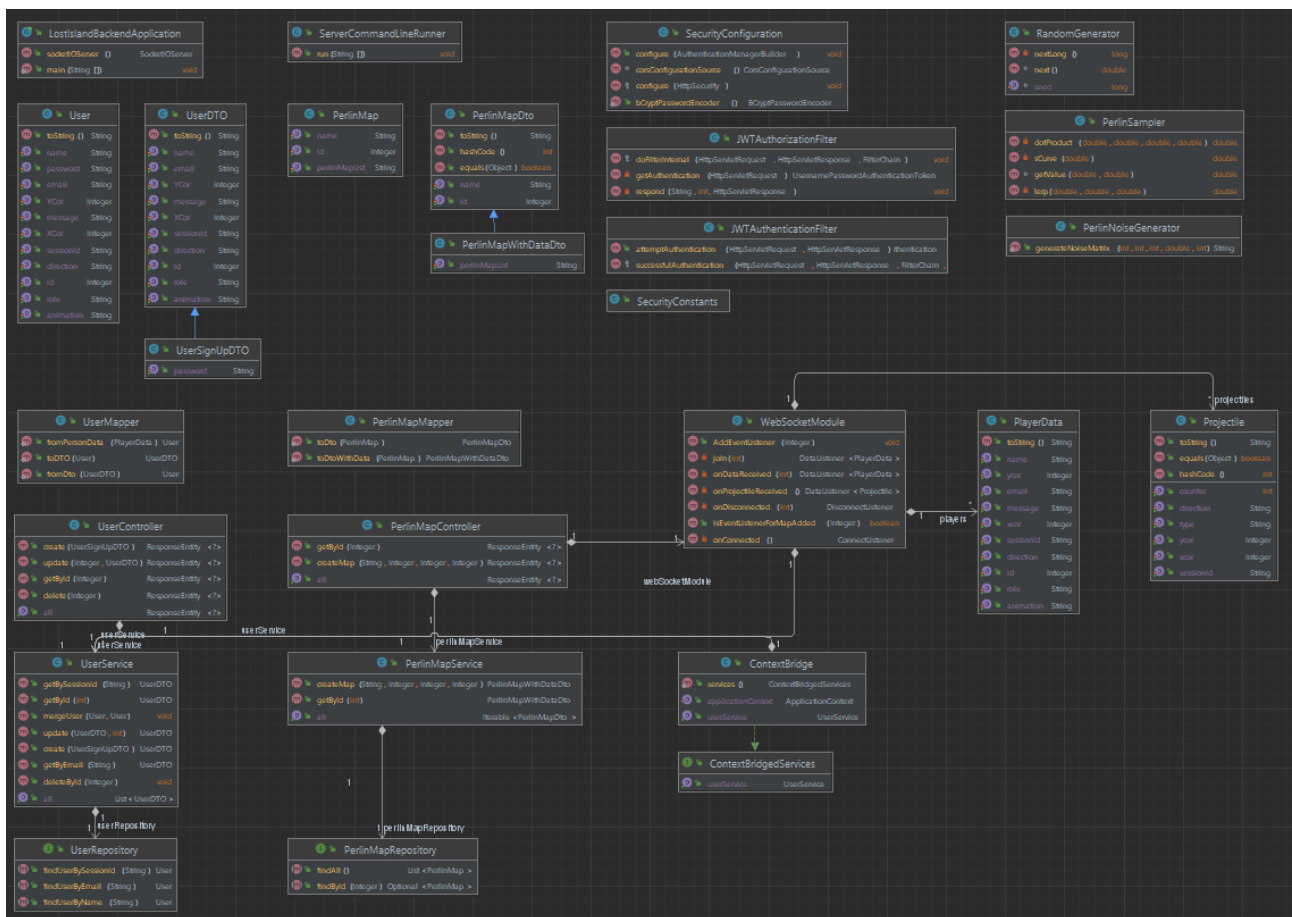
Wir haben uns entschieden, dass die Spiel Welt Prozedural generiert werden soll. Wenn ein Benutzer eine neue Welt erstellen will, soll im Backend eine Funktion aufgerufen werden, die diese generiert. Um diese Welt organisch wirken zu lassen wird in diesem Projekt mit Perlin-Noise gearbeitet.

Um eine Welt im Frontend darstellen zu lassen benötigt es ein 2-Dimensionales Array. Die erste Dimension steht für die Y-Achse und die zweite für die X-Achse. Pro Wert wird eine Zahl zwischen -1 und 1 erzeugt. Im Abbild auf der rechten Seite sieht man das Ergebnis einer solchen Berechnung. Kleinere Zahlen werden dunkel eingefärbt und grosse weiss.



Da wir in diesem Projekt nur 4 Stufen benötigen werden die Zahlen skaliert und gerundet, sodass diese 0, 1, 2 oder 3 ergeben. Jede Zahl steht für Wasser, Sand, Gras oder Wald.

4.4 UML Diagramm



5 Frontend

5.1 Zielpublikum

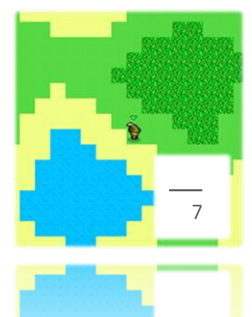
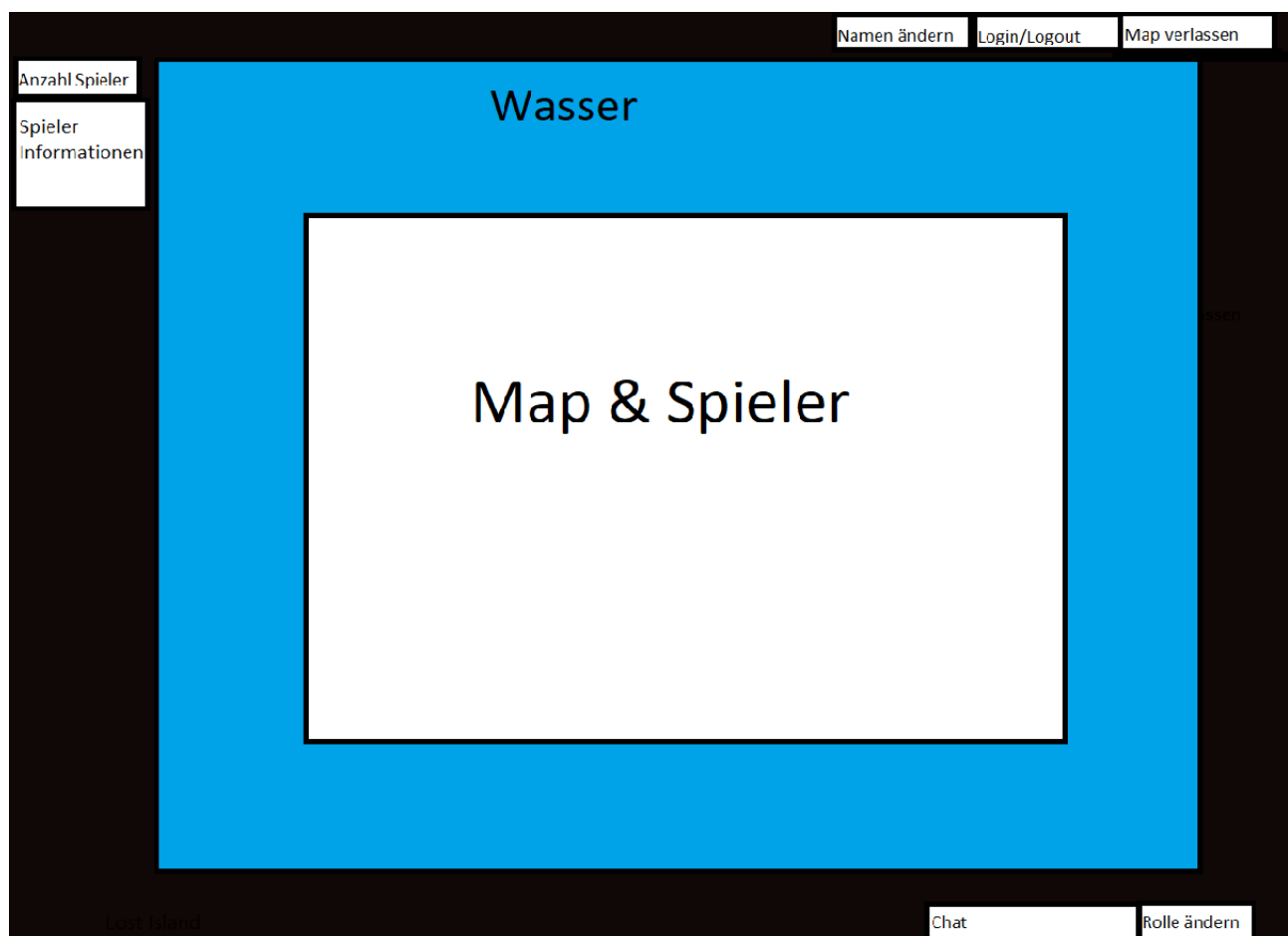
5.1.1 Beschreibung

Dieses Spiel ist eher für junge Menschen geeignet die gerne neue Leute kennen lernen möchten. Durch die Chatfunktion wird dies ermöglicht.

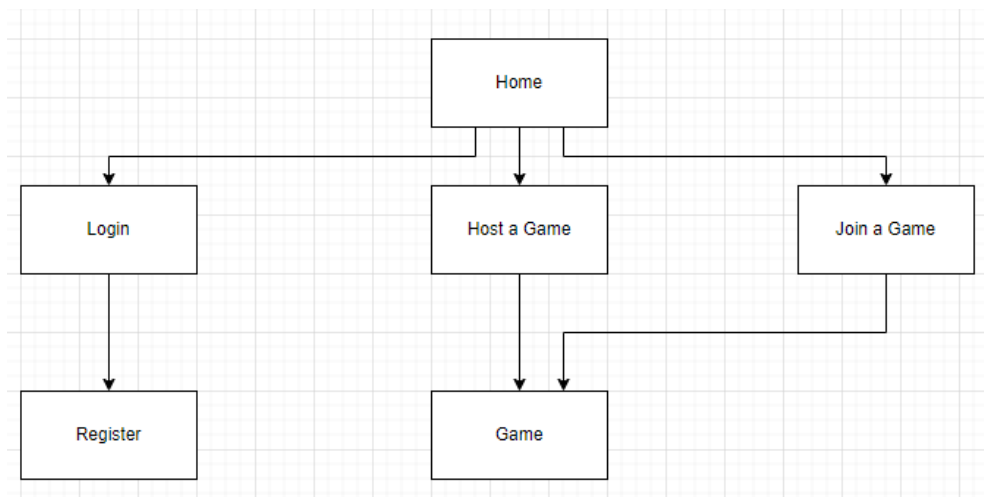
5.1.2 Konsequenzen bei der Umsetzung

Die Grafiken und Farben sind so gewählt, dass es junge Leute anspricht.

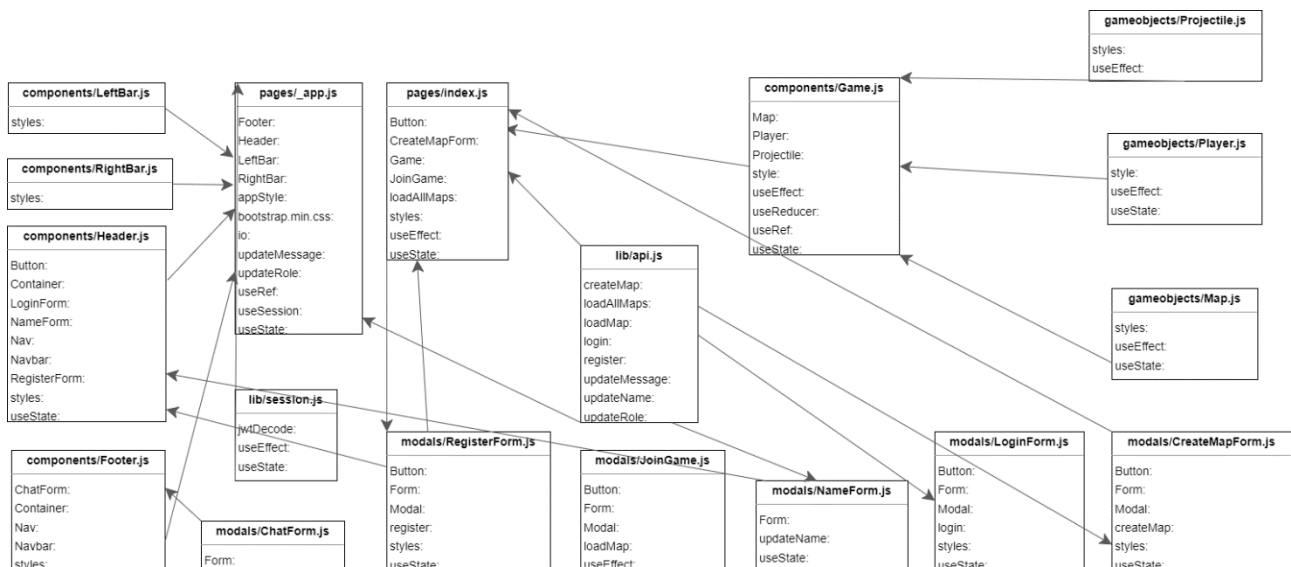
5.2 Mock Up



5.3 Sitemap



5.4 UML Diagramm



5.5 Wichtigste Components

5.5.1 Game.js

In dieser Komponente ist der wichtigste Teil der Game Logik implementiert.

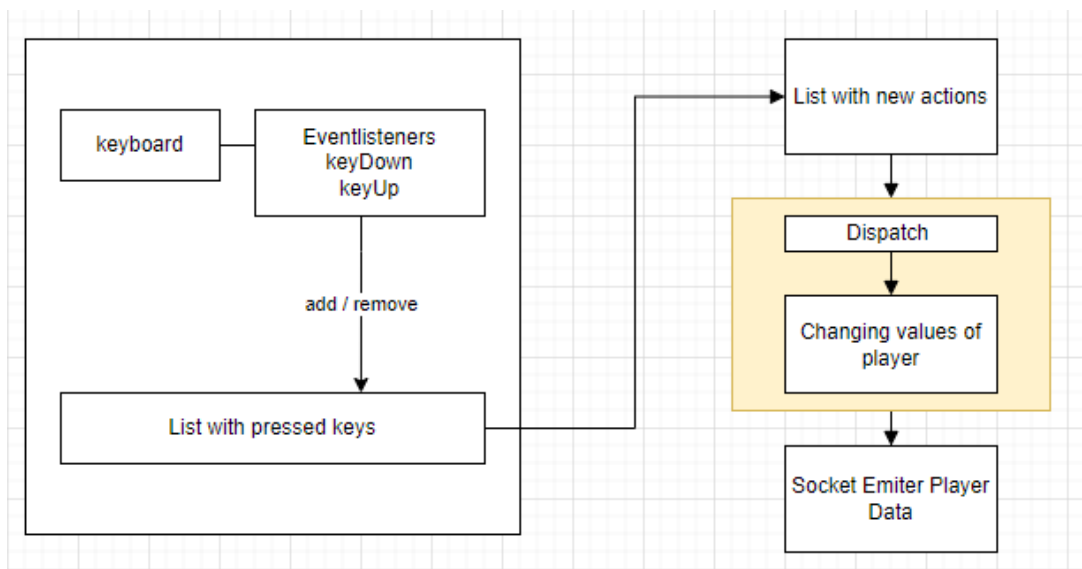
5.5.1.1 Verarbeitung von Keyboard Events und Versendung der Daten via Socket:

Um zu verhindern, dass ein Key Event mehrfach ausgeführt werden kann, wird eine Key Code (Der Wert einer Taste) Liste erstellt, die den jeweiligen Wert nur der Liste hinzufügt, wenn sie noch nicht vorhanden ist.

Die Daten des eigenen Spielers werden mit einem Reducer gehandelt (In der Skizze farbig eingefärbt). Ein Reducer führt eine Funktion basierend auf den Aktions-Typ aus und gibt den veränderten Status des Players als Rückgabe Wert zurück. Das Verändern des Spieler Objektes löst ein re-render aus. In der folgenden Skizze wird abgebildet wie und wann ein dispatch (aufruf der Reducer Funktion) aufgerufen



wird. Das Aufrufen der Funktion wird zeitlich, mit einem Intervall, festgelegt. Jede Veränderung des Spielers wird an den Server mit der jeweiligen Spiel Raum ID gesendet.



5.5.1.2 Verarbeitung der erhaltenen Daten und Rendering:

Die Daten, die benötigt werden zum Rendern werden über den Receiver erhalten. Auch diese des eigenen Spielers. Für das Rendern erhält der Receiver eine Liste mit Objekten. Mit einer Iteration durch die Liste wird Spieler für Spieler verarbeitet und gerendert. Ein Objekt sieht folgender Masse aus:

```
{id=3, sessionId='4ae9e61c-acf8-46b4-b703-e51d6a7dbe56', name='user3', xcor=0, ycor=0, direction='right', animation='idle', role='Archer-Green', message=""}
```

Anhand dieser Objekte kann herausgefiltert werden, ob der eingeloggte Spieler dieselbe Session ID besitzt und weiss welche Daten zum eigenen Spieler gehören.

5.5.2 Modals

Im Ordner «Modals» sind alle Komponente, die zuständig sind, Eingaben des Benutzers aufzunehmen und weiter zu verarbeiten. Das Verarbeiten und Versenden dieser Formular Daten wurden in der tiefst möglichen Schicht herausgelagert, um diese auch für andere Komponenten verfügbar zu machen.

5.5.2.1 RegisterForm / LoginForm

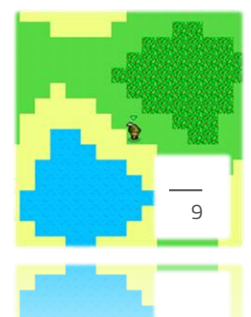
RegisterForm ist ein Formular, mit dem man einen neuen Benutzer erstellen kann. Mit E-Mail und Passwort kann man sich danach mit der LoginForm Komponente einloggen

5.5.2.2 ChatForm / NameForm:

Senden von Nachrichten und Bearbeitung des eigenen Namens. Die Daten werden per Patch request an die Datenbank gesendet, aber auch über Socket, als Spieler Datenpaket, übermittelt. Die Änderung des Wertes des Spieler Attributes Message löst über den Socket bei allen Spielern ein Rendering der Nachricht im Spiel (als Form einer Sprechblase) aus.

5.5.2.3 CreateMapForm

Bei dem Erstellen einer neuen Spiel Welt muss der Benutzer Werte eingeben. Diese Werte werden an das Backend gesendet. Im Backend wird eine neue Map erstellt und and den Benutzer retourniert.



5.5.2.4 JoinGame

Beim Aufruf dieser Komponente wird ein GET Request ausgeführt welches alle Spiel Welten der Datenbank erhält. Diese werden im Browser als Liste gerendert. Der Benutzer kann dann einer Welt beitreten.

5.5.2.5 Player, Map, Projectile

Diese Komponenten werden im Browser Bildlich angezeigt. Sie passen sich den mit gegebenen Werten an. Wichtigste Rolle ist das Verschieben der Grafiken auf der X und Y-Achse.

6 Testing

6.1 Testumgebung

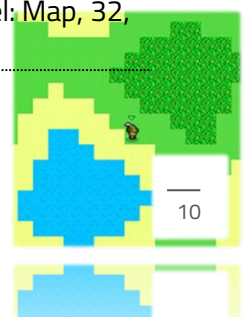
- Windows 10
- Chrome
- Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz 3.19 GHz
- Display: 1920 X 1080

6.2 User Acceptance Testfälle

Abschnitt	Inhalt
ID	T-01
Vorbedingungen	Backend und Frontend ist gestartet.
Ablauf	Klick oben rechts auf Login. Klick auf «Register for free». Gib in jedes Feld ein valides Attribut ein und klicke dann auf Register. (Merke dir die Angaben für T-02)
Erwartetes Resultat	Das Pop-up schliesst sich nach dem Registrieren.

Abschnitt	Inhalt
ID	T-02
Vorbedingungen	T-01 war erfolgreich.
Ablauf	Klick oben rechts auf Login. Gib bei E-Mail und Password das gleiche ein wie bei T-01. Klicke anschliessend auf login.
Erwartetes Resultat	Das Pop-up schliesst sich nach dem Login. Oben rechts steht nun Logout. Im Spielbereich erscheinen nun folgende Optionen. «Join a Game» und «Host a Game»

Abschnitt	Inhalt
ID	T-03
Vorbedingungen	T-02
Ablauf	Klick auf Host a Game. Gib hier wieder gültige Attribute ein. Beispiel: Map, 32, 1, 8. Klicke auf Create.



Erwartetes Resultat Es erscheint ein Spieler auf einer Map.

Abschnitt	Inhalt
-----------	--------

ID	T-04
----	------

Vorbedingungen	T-03
----------------	------

Ablauf	Klick unten rechts auf Change Role.
--------	-------------------------------------

Erwartetes Resultat	Jedes Mal, wenn auf den Button gedrückt wird, ändert sich das Aussehen des Charakters. Der Charakter wird erst aktualisiert, wenn der Spieler sich bewegt hat. (Pfeiltasten pressen)
---------------------	--

Abschnitt	Inhalt
-----------	--------

ID	T-05
----	------

Vorbedingungen	T-03
----------------	------

Ablauf	Klick unten rechts auf «Chat with your friends». Gib irgendeinen Text ein. Drücke Enter.
--------	---

Erwartetes Resultat	Eine TextBox erscheint neben dem Spieler, welche nach zehn Sekunden wieder verschwindet. Der Text wird erst angezeigt, wenn der Spieler sich bewegt hat. (Pfeiltasten pressen)
---------------------	--

Abschnitt	Inhalt
-----------	--------

ID	T-06
----	------

Vorbedingungen	T-03
----------------	------

Ablauf	Klick oben rechts auf Change your name. Gib irgendeinen Namen ein. Drücke Enter.
--------	--

Erwartetes Resultat	Oben links kann man den neuen Namen sehen. Der Name wird erst aktualisiert, wenn der Spieler sich bewegt hat. (Pfeiltasten pressen)
---------------------	---

Abschnitt	Inhalt
-----------	--------

ID	T-07
----	------

Vorbedingungen	T-03
----------------	------

Ablauf	Klick oben rechts auf Logout.
--------	-------------------------------

Erwartetes Resultat	Der Startscreen mit dem Floss wird angezeigt. Oben steht nun wieder Login
---------------------	---

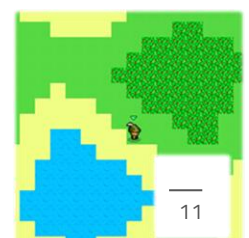
Abschnitt	Inhalt
-----------	--------

ID	T-08
----	------

Vorbedingungen	T-03
----------------	------

Ablauf	Halte R gedrückt.
--------	-------------------

Erwartetes Resultat	Der Charakter wirft sich auf den Boden, legt sich hin und fängt an sich zu drehen.
---------------------	--

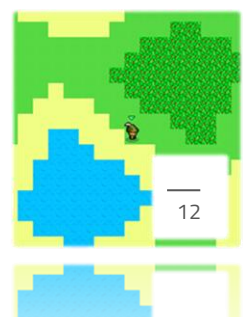


Abschnitt	Inhalt
ID	T-09
Vorbedingungen	T-03
Ablauf	Halte eine beliebige Pfeiltaste gedrückt
Erwartetes Resultat	Der Charakter läuft in die richtige Richtung.

Abschnitt	Inhalt
ID	T-10
Vorbedingungen	T-03
Ablauf	Laufe gegen Wald oder gegen Wasser.
Erwartetes Resultat	Der Charakter läuft nicht ins Wasser oder in den Wald, sondern ist blockiert.

Abschnitt	Inhalt
ID	T-11
Vorbedingungen	Inkognito Tab und normaler Tab mit Lost Island offen
Ablauf	Melde dich auf beiden Tabs mit unterschiedlichen Accounts an. Auf dem einen drückst du auf Host a game und füllst die Angaben aus. Im anderen Tab drückst du auf Join a game und wählst den Namen der Map die du vorhin erstellt hast.
Erwartetes Resultat	Es befinden sich zwei Spieler auf der Map. Der eigene Spieler hat einen grünen Pfeil über den Kopf.

Abschnitt	Inhalt
ID	T-12
Vorbedingungen	T-11
Ablauf	Laufe in einem der beiden Tabs herum.
Erwartetes Resultat	Im anderen Tab sieht man wie der Spieler bewegt.



7 Testprotokoll

Für jede Durchführung braucht es ein Testprotokoll.

Name des Testers: Kevin Berger

Release: v1.0

Datum und Uhrzeit: Date.now(); 1657716940508

ID Erfolgreich Bemerkungen

T-01	Ja	
T-02	Ja	
T-03	Ja	
T-04	Ja	
T-05	Ja	
T-06	Ja	
T-07	Ja	
T-08	Ja	
T-09	Ja	
T-10	Ja	
T-11	Ja	
T-12	Ja	



8 Fazit

8.1 Sebastian Kull

Am Anfang des Projektes habe ich mir für das Commiten zu wenig Mühe gegeben. Ich fand das am Anfang eher schwierig. Auch wenn ich für die Planung dieses Projektes das Gefühl hatte, dass ich mir Mühe gegeben habe, war die praktische Umsetzung der Ideen schwierig. Es war sehr schwierig alle Arbeiten voneinander abzugrenzen und in kleinere Tasks einzuteilen.

Die meisten Code Beispiele und Anleitungen in Netz waren nicht für React geschrieben. Diesen Code umzuschreiben ist anspruchsvoll. Ich hatte ständig das Gefühl die ganze Logik selbst erfinden zu müssen. Mir ist aufgefallen das ich immer wie mehr Dokumentationen durchlese, und diese auch verstehe.

Die Team Arbeit war in diesem Fall nicht so gut, da zu wenig kommuniziert wurde. Viele Ideen habe ich zum ersten Mal praktisch umgesetzt. Dies hatte zur Folge, dass viel Code immer wieder überarbeitet werden musste.

Ich habe gelernt, dass es einfacher ist lange Variablen zu benutzen anstatt kurze. Es ist einfach die Variable nach dem zu benennen was sie beinhaltet, anstatt sie möglichst kurz auszuwählen.

Im Allgemeinen hat mir diese Projekt Arbeit bis jetzt am meisten gefallen. Ich habe mich sehr verwirklichen können. Mir gefällt es sehr zum Abschluss im BBC so viel Spass zu haben.



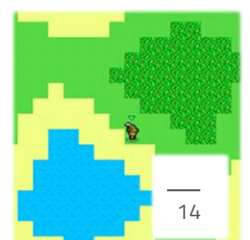
'Mis Backend isch es wackend'

8.2 Nepomuk Lehmann

Das Projekt war sehr anspruchsvoll für mich. Ich war sehr schnell abgehängt und motivationslos. In den ersten zwei Wochen habe ich fast nichts gemacht. Dann kam Dimi um die Ecke und hat mich ein bisschen angekurbelt. Ab diesem Zeitpunkt habe ich immer mal wieder etwas geleistet, jedoch auch nur mit viel Hilfe. Dieses Projekt hat mein Selbstwertgefühl komplett «zerscheppert». Im wahrsten Sinne: !(Mir hat das Projekt sehr gefallen)



'I cha ds alles nümme'



9 Impressum

Grafiken: <https://merchant-shade.itch.io/16x16-puny-characters>

Perlin-Noise Implementation: Dimitri Waber

