

TDI - HA 7 - Dokumentation

Mitwirkende

Lucas Held - 35022624

Keanu Stückrad - 35198923

Gianluca Voss - 35197942

Patrick Michla - 35206231

Ausführung des Projekts

Um das Projekt zu kompilieren, wird zuerst das Modul `serve` global mittels `npm` installiert:

```
npm install -g serve
```

Anschließend wird die Kompilierung mit folgendem Befehl gestartet:

```
npm run build
```

Das kompilierte Projekt liegt nun im `build` Ordner. Um das Projekt mittels `serve` auszuführen, wird folgender Befehl benötigt:

```
serve -s build
```

Hinweis: Das Projekt wurde nur für Browser und Tablets optimiert, die eine Bildschirmbreite von mindestens 800 Pixeln aufweisen. Sonst kann es im Strategiemodus zu Problemen kommen.

Weiterhin sollte dort weder gescrollt noch gezoomt werden.

Projektstruktur

Neben der `index.js` wurde das Projekt in einen `ui`- und `redux`-Ordner unterteilt.

Der `redux`-Ordner wurde weiterhin neben der `reducer-list.js` in einen `actions`-, `data-classes`-, `mapper`- und `reducers`-Ordner unterteilt und ähnelt damit der Struktur, die in der Vorlesung vorgestellt wurde.

Die zusätzlichen Ordner, die nicht in der Vorlesung vorkamen, haben wir hinzugefügt, um eine bessere Übersicht zu bewahren. Die `mapStateToProps` und `matchDispatchToProps` aus den Komponenten wurden in eigene Dateien ausgelagert und in den `mapper`-Ordner gepackt. Die `Actions` liegen alle im `actions`-Ordner. Der `data-classes`-Ordner und die Klassen dort werden genutzt, um nicht jedes Mal alle Attributnamen in einem JavaScript-Objekt angeben zu müssen. So kann z.B. `new Player("id", "name")` anstatt `{id: "id", name: "name"}` geschrieben werden, wenn per `Action` und `Reducer` ein Wert dem `Store` hinzugefügt werden soll.

Der `ui`-Ordner ist unterteilt in einen `components`- und `pages`-Ordner. Im `components`-Ordner befinden sich UI-Elemente, die mehrmals auf unterschiedlichen Seiten genutzt werden. Im `pages`-Ordner sind alle Seiten, die laut Aufgabenstellung gefordert, zur Seeschlacht gehören. Für einige Dateien gibt es auch gleichnamige `CSS`-Dateien im gleichen Ordner. Im `ui`-Ordner befindet sich auch die `App.js`.

Routing

In der App benutzen wir einen `Memory-Router`, der die URLs zu den einzelnen Seiten angibt.

Weiterhin ist es so möglich, dass bei einem `Reload` oder beim Aufrufen einer Seite immer auf die Startseite gewechselt wird, damit der `Store` immer richtig aufgebaut wird und der Spielfluss gewährleistet bleibt.

Redux

An vielen Stellen im Projekt nutzen wir `Redux`, um Daten aus dem `Store` zu propagieren, anstatt nur einzelnen `states` in den Komponenten zu halten. `Redux` ist durch seine Struktur zwar etwas komplizierter, aber stets sinnvoll, da sonst Daten über mehrere Komponenten gesendet werden müssten. Mit `Redux` wird dies vermieden, da der `Store` von überall aus zugänglich ist, wenn die

nötigen states in `mapStateToProps` in unserem jeweiligen Mapper (in `/redux/mapper`) angegeben werden. Der Reducer gleichen dem Standard Schema (switch cases und initial states). Weiterhin gibt es in jedem Reducer einen Fall, um alle states auf den initial state zurückzusetzen (wird immer auf der Startseite aufgerufen). Alle Reducer werden in der `reducer-list.js` exportiert und in der `index.js` als store erstellt, damit sie in den anderen Dateien nutzbar sind.

Pages

Start

Umgesetzt in `StartScreen.js`.

Hier wird ein neues Spiel gestartet.

Seeschlacht

Version 20.1.7

Werde Herrscher(in) über die Meere

Neues Spiel

Spielerprofil

Umgesetzt in `PlayerProfileScreen.js`.

Spielerprofil

Spieler 1


Nick

test

Pin

....

Avatar



<

>

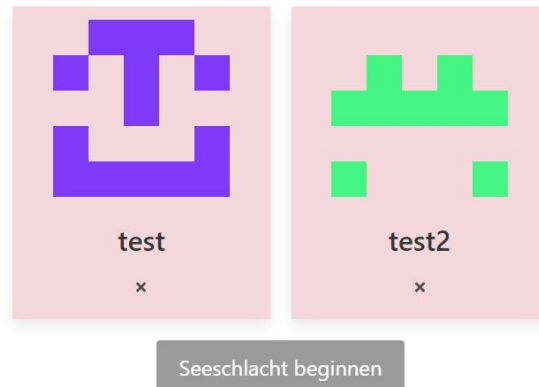
Speichern

Hier kann ein Nick (Mindestlänge sind drei Zeichen) und ein Pin (Muss genau vier Zeichen lang sein) eingegeben werden. Der Avatar ist auch wählbar und wird mit einem Seed (zufällige Zahl) in ein <Ideticon> erzeugt. Der Button wird nur aktiviert, wenn die oben genannten Bedingungen stimmen.

Setup

Umgesetzt in SetupScreen.js.

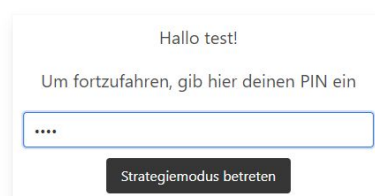
Durch klicken auf die Karten gelangen die Nutzer zum Lock-Bildschirm des jeweiligen Nutzers. Der Button ist nur aktiviert, wenn beide Spieler bereit sind, d.h. beide Spieler im Strategy-Bildschirm ihre Schiffe gesetzt haben.



Lock

Umgesetzt in LockScreen.js.

Hier muss der oben auf der Karte genannte Spieler seinen vier-stelligen Pin eingeben, um auf die nächste Seite (Strategy-/Fight-Bildschirm) zu kommen. Zum Spielstart (erster Wechsel zu Fight) wird



per Zufall ein Spieler ausgelost. Der Zusatztext auf dem übernächsten Screenshot wird dann auch angezeigt

Hallo gvo!

Die 🎲 🎲 sind gefallen. Du darfst beginnen!

Um fortzufahren, gib hier deinen PIN ein

Kampfmodus betreten

Strategiemodus

Umgesetzt in StrategyModeScreen.js.

Beim Klicken auf ein Schiff in der rechten Liste, öffnet sich das Fenster in der Mitte, um die Richtung

Strategiemodus

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

Fertig

3. Destroyer
(Ausrichtung wählen)

Battleship

Cruiser

Cruiser

Destroyer

Destroyer

Destroyer

Submarine

Submarine

Submarine

Submarine

auszuwählen. Wird von einem Kästchen im Fenster ein Schiff gezogen, wird dieses Kästchen als Ankerpunkt genutzt, um das Schiff im Spielfeld zu platzieren. Dort färbt es sich beim Platzieren lila. Während des Ziehens färbt es sich rot oder grün, je nachdem ob es möglich ist, das Schiff zu platzieren. Gezogene Schiffe werden in der Liste grün angezeigt und platzierte Schiffe grau. Die

Schiffe können weiterhin auf dem Spielfeld verschoben werden. Wenn nachträglich die Richtung geändert werden soll, muss das Schiff vorher aus dem Spielfeld gezogen werden. Während des ziehens werden alle angrenzenden Felder von Schiffen auf dem Spielfeld hellrot angezeigt. Dort können ebenfalls keine Schiffe platziert werden. Der Button ist aktiviert, sobald alle Schiffe gesetzt worden und bringt den Spieler zum Setup zurück.

Kampfmodus

Umgesetzt in FightModeScreen.js.

Hier kann der Spieler rechts sein eigenes Spielfeld mit allen im Strategiemodus platzierten Schiffen sehen. Weiterhin werden die gegnerischen Treffer oder Misserfolge angezeigt. Über den Button "Spielerprofil bearbeiten" wird der Spieler zurück zum Spielerprofil gebracht, um gegebenenfalls seine Daten zu ändern. Der Button "Zug beenden" ist nur aktiv, wenn der Spieler einmal beim Schießen und Klicken eines Feldes im gegnerischen Meer einen Misserfolg hatte. Dieser wechselt dann zum Lock-Bildschirm des Gegners. Sobald der Spieler, der inaktiv ist, keine Felder mehr mit nicht getroffenen Schiffen besitzt, wird das Spielende aufgerufen.

Kampfmodus

Spielerprofil bearbeiten

Hallo gvo
Willkommen zurück in der Kommandozentrale

Das Meer von gvo

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

Dein Meer

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

Zug beenden

Das Spielerprofil sieht nach dem Wechsel vom Kampfmodus ein wenig verändert aus. Beide Buttons bringen den Spieler zurück zum Kampfmodus, der Button "Speichern" ist nur aktiv, wenn die weiter oben beschriebenen Bedingungen (siehe Spielerprofil) in beiden Eingabefeldern erfüllt sind. Der alte Nick und Pin wird ebenfalls angezeigt.

Spielerprofil

[Zurück zum Kampfmodus](#)

Spieler 1

Nick

gvo


Old nick: test

Pin

....

Old pin: 1**1

Avatar



<

>

Speichern

Spielende

Umgesetzt in EndScreen.js.

Glückwunsch!

Hey test, du hast gewonnen.

[Neues Spiel?](#)

Bei Spielende wird dem Sieger gratuliert und es ist möglich, ein neues Spiel zu starten. Durch Klick auf den Button gelangen die Spieler wieder zur Startseite.

Components

SetupCardComponent.js

Durch diese Komponente ist es möglich, die Karten der Spieler im Setup anzuzeigen. Weiterhin signalisiert es, ob der jeweilige Spieler für die Schlacht bereit ist.

Playground.js

Mit dieser Komponente wird das Spielfeld erzeugt. Das Spielfeld besteht aus einer Div-Tabelle. Die Ränder sind einfache Div-Elemente und das nutzbare innere Spielfeld besteht aus Feldern vom Typ Playground, die mit "Field.js" verwaltet werden. Die Felder einer Zeile liegen dabei in jeweils einer Div mit dem Style {display: "flex"}, damit innerhalb einer Zeile nicht umgebrochen wird.

Dem Playground wird beim Erstellen der dazugehörige Spieler und die Spielfeld-Id (mittels eines Enum-Types) übergeben (es gibt vier Spielfelder, ein eigenes, ein gegnerisches und die zwei Spielfelder im Kampfmodus, die zunächst verdeckt sind).

Field.js

In der Field-Komponente sind sämtliche Methoden und Listener enthalten, um Schiffe zu ziehen oder im Kampfmodus Felder anzuklicken. Gerendert wird hier einmal das Feld als Div (vom Spielfeld (Typ Playground) oder vom Dialog (Typ Overlay)) und darin nur gegebenenfalls die Schiff-Kopie als Ship-Komponente, wenn ein Schiff gezogen werden soll. Auf den Feldern des Spielfeldes liegt ein onClick-Listener, der im Kampfmodus die Felder einfärbt, je nachdem, ob ein Schiff getroffen wurde oder nicht. Auf Feldern des Spielfeldes und des Dialogs liegt ein onMouseDown-Listener (onTouchStart auch, um mobile Endgeräte zu unterstützen), der das Signal gibt, die Schiff-Kopie zu rendern und danach auf dem "document" einen onMouseMove-Listener (onTouchMove mobil) und onMouseUp-Listener (onTouchEnd mobil) erzeugt. Weiterhin werden durch ihn alle Felder auf dem Spielfeld, die neben einem besetzten Feld liegen, hellrot gerendert und sind so auch für die Schiff-Kopie belegt. Wird der gedrückte Cursor nun bewegt, wird mittels onMouseMove (onTouchMove) Event das Schiff an die Cursorposition gerendert und berechnet, ob das Schiff an dieser Position platziert werden kann. Funktioniert dies, wird es grün gerendert, funktioniert es nicht, wird es rot gerendert. Sobald die Maustaste losgelassen wird (oder der Bildschirm nicht mehr gedrückt wird/das Schiff gezogen wird), wird das onMouseUp (onTouchEnd) Event aufgerufen. In der dazugehörigen Funktion wird der onMouseMove-Listener (onTouchMove) gelöscht und das Schiff an die Position im Spielfeld gemalt, wo es hingezogen wurde, falls dies möglich ist. Dabei werden die einzelnen Felder des Spielfeldes gerendert und die Schiff-Kopie entfernt. Die hellroten Felder (neben einem Schiff) werden auch zurück auf die Ausgangsfarbe (hellblau) gerendert. Die Listener werden spätestens beim Aufruf der Methode "componentWillUnmount" gelöscht. Beim Verschieben eines Schiffes im Spielfeld werden die Felder dieses Schiffes mittels onMouseDown (onTouchStart) Event auch auf die Ausgangsfarbe (hellblau) gerendert. Die Berechnungen, ob ein Schiff beim Ziehen platziert werden darf bzw. ob es beim Loslassen platziert wird, funktionieren so, dass immer geschaut wird, ob das Schiff außerhalb des Spielfeldes ist ("isValid") oder ob es neben einem anderen Schiff ist ("noShipNear").

Ship.js

Diese Komponente wird genutzt, um die Schiffe im Strategiemodus anzuzeigen (hier wird das Label-Element mitgerendert) und um die gezogenen Schiffe aus dem Fenster im Strategiemodus zu erzeugen. Diese Schiffe sind nur Klone der eigentlichen Schiffe und werden nur solange gerendert, wie der Spieler das Schiff zieht. Schiffe bestehen aus Div-Elementen.

ShipDirectionDialog.js

Diese Komponente öffnet sich im Strategiemodus, sobald ein Schiff in der Liste angeklickt wird. Hier wird die Ausrichtung der Schiffe gewählt. Diese Komponente enthält die Div-Tabelle, aus der das Schiff gewählt wird. Die Zellen, die zu einem Schiff gehören sind Field-Komponenten vom Typ Overlay.