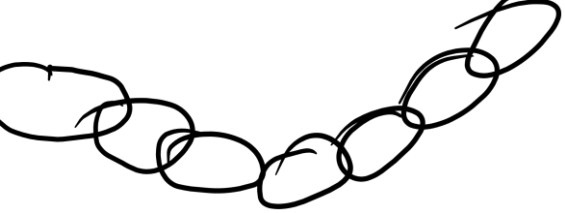
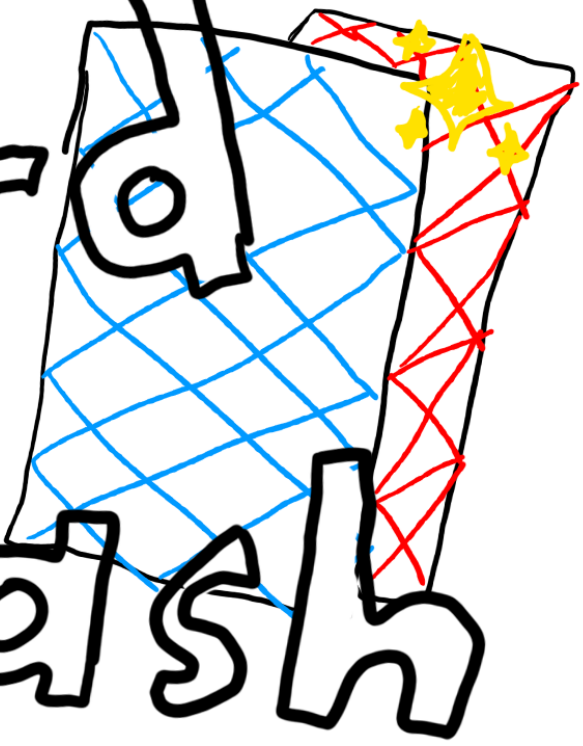
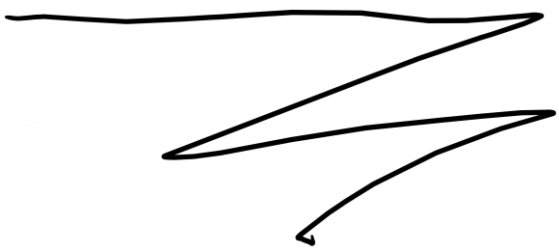


Card Clash

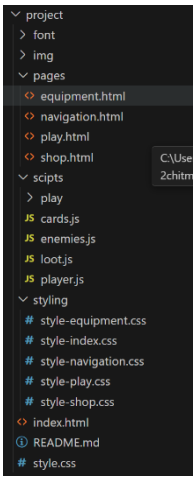


1. Sprint-Doku

Roland Bauer



Grundgerüst



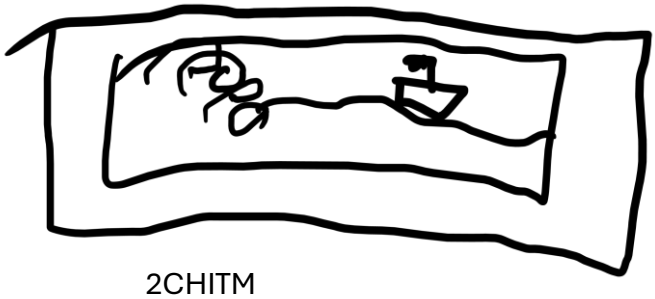
Das Grundgerüst meines Servers wurde erstellt. Es gibt eine Startseite, von der aus man zur Unterseite **Navigation** gelangt, um die anderen Unterseiten zu erreichen. Außerdem wurden verschiedene Ordner mit ihren jeweiligen Funktionen angelegt.

Für die wichtigsten Themen wurden eigene Unterseiten erstellt, zum Beispiel **Equipment**, **Navigation**, **Play** oder der **Shop**. Das sind die Bereiche, in denen sich der User später bewegen wird.

Es gibt eine zentrale **style.css**, die für alle Unterseiten gilt. Darin sind grundlegende Dinge wie Schriftarten, Layouts oder Formatierungen für bestimmte Elemente festgelegt.

Zusätzlich gibt es den Ordner **styling**, in dem die individuellen Styles für jede Unterseite verwaltet werden.

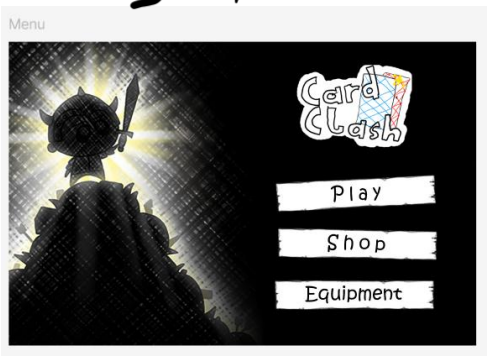
Außerdem gibt es einen **img**-Ordner, in dem alle wichtigen Bilder gespeichert werden. Dazu gehören z. B. Karten, Spielfiguren, Spieler-Avatare und Hintergründe. Die Dateien sind darin übersichtlich und gut strukturiert abgelegt.



Figma nachbauen

Meine Webseite wurde nach dem Design meines Prototyps aus **Figma** aufgebaut. Layout, Farben und Anordnung der Elemente orientieren sich stark an der Vorlage, um ein stimmiges und benutzerfreundliches Gesamtbild zu schaffen.

Figma



Web-side

Wiederkehrende Elemente wie Buttons, Textfelder oder Kartenansichten wurden in einem **universellen Style-Bereich** zusammengefasst. So müssen sie nicht jedes Mal neu gestaltet werden, was Zeit spart und für ein einheitliches Design sorgt.

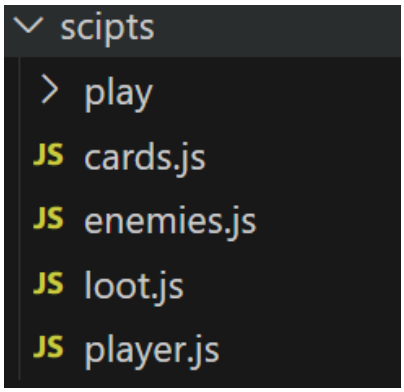

```scss
/***** Scroll *****/
.scroll {
    width: 8.5vw;
    height: 7vh;

    background-image: url(img/plate/scroll.png);
    background-size: contain;
    background-position: center;
    background-repeat: no-repeat;

    display: flex;
```

Zusätzlich habe ich ein **Root-Verzeichnis** für CSS-Variablen angelegt, über das ich Farben, Schriftgrößen und Abstände zentral steuern kann. Die Farbpalette stammt aus meinem **Moodboard** und sorgt für einen konsistenten Look über alle Seiten hinweg.

```scss
:root{
    --white: #fff;
    --black: #000;
    --blue: #8DC9DC;
    --red: #E73E3E;
    --gold: #E0CA53;
}
```

Javascript

Im Ordner **scripts** werden alle wichtigen Funktionen gespeichert. Dazu gehören zum Beispiel der Spielablauf, die dynamische Content-Generierung sowie Logiken für den Gegner und den Spieler selbst.




```
`Hallo ${name}!`
```

Die Karten, der Spieler, Gegner und Loot-Objekte werden mithilfe von Klassen erstellt. Je nachdem, um welches Objekt es sich handelt, besitzen sie unterschiedliche Eigenschaften und Funktionen – das ist besonders wichtig für das Kampfsystem.

Karten

```
/**********************
 * Card Klass
 **********************/
class Card {
  constructor(name, attack, magie, attackBlock, magieBlock, universalBlock, live, img) {
    this.name = name;
    this.attack = attack;
    this.magie = magie;
    this.attackBlock = attackBlock;
    this.magieBlock = magieBlock;
    this.universalBlock = universalBlock;
    this.live = live;
    this.img = img;
  }
}
```

Gegner

```
/**********************
 * Enemie class
 * ********************/
class enemie{
    constructor(name, deck, live, img){
        this.name = name;
        this.deck = deck;
        this.live = live;
        this.currentLive = live;
        this.img = img
    }
}
```

Loot

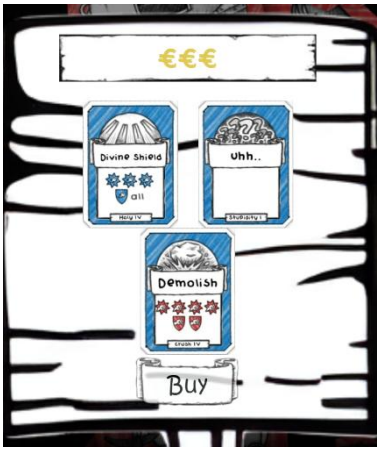
```
/**********************
 * Loot Class
 **********************/
class Loot{
    constructor(name, label, deck, price, img){
        this.name = name;
        this.label = label
        this.deck = deck;
        this.price = price
        this.img = img
    }
}
```

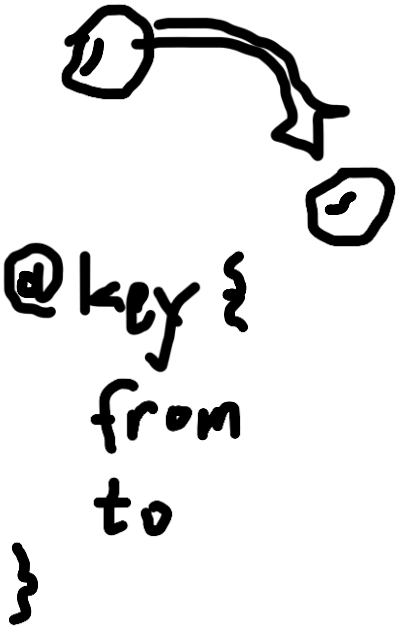
Spieler

```
/************************
 * New Player
 ************************/
let newPlayer = {
    name: 'Held',
    health: 5,
    deck: [],
    startDeck: startdeck[Math.floor(Math.random() * startdeck.length)],
    gold: 0,

    // Buyed Loot
    weaponArr: [],
    armorArr: [],
    skilArr: [],
    // loot Cards
    weapon: null,
    armor: null,
    skil: null,

    getAllCard: function () {
        this.deck = [];
        this.deck.push(this.startDeck),
        this.deck.push(this.weapon);
        this.deck.push(this.armor);
        this.deck.push(this.skil);
    }
}
```

Meine Ziele

Als nächstes möchte ich daran arbeiten, dass der **Shop** funktioniert. Ziel ist es, dass der Spieler bereits einkaufen kann und die gekauften Items korrekt in seiner **Datenbank** gespeichert werden. So wird der Fortschritt des Spielers dauerhaft festgehalten, und gekaufte Items können später im Spiel verwendet werden.

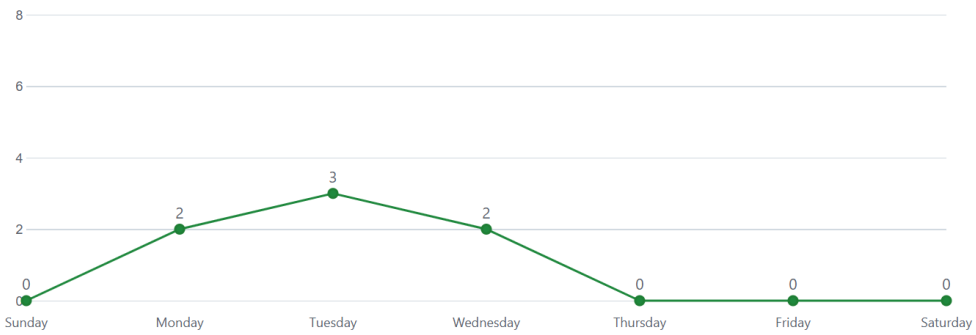




Zusätzlich plane ich, einige **Animationen** einzubauen, um das Nutzererlebnis zu verbessern. Ein Beispiel wäre: Wenn der Spieler nicht genügend Geld hat, soll der **Kauf-Button** eine kleine Fehlermeldung anzeigen – zum Beispiel durch ein kurzes Wackeln oder eine visuelle Rückmeldung, dass der Kauf nicht möglich ist. Solche kleinen Effekte machen die Seite lebendiger und intuitiver.

Github - Insights

Commits



Code frequency