Konzeption zu MERS BattleApp

# Frameworks und Tools

## NodeJS Server (hosted on Heroku free)

## CouchDB (hosted on Cloudant)

## IntelliJ Webstorm (IDE)

## Bootstrap 3 (CSS)

## Knockout 3 (HTML Generation)

## NodeModules

## Nano

## Passport-Facebook (Milestone 2)

## Socket-IO (Milestone 2)

# Struktur

## Sämtliche Tabellendaten (zukünftig Charaktere etc.) werden auf der CouchDB. Eine Unterscheidung der Entwicklungstabellen und Produktionstabellen erfolgt über z prefix. Der Server wird grundsätzlich als API behandelt, eine Kommunikation zwischen Client und Server erfolgt lediglich über diese. Eine Client wird (Milestone 2) mit Sockets realisiert. Der Client wird zu einem gekapselten Modul.

## Client

### Der Client wird vollständig mit Bootstrap realisiert. Die dynamische Generierung erfolgt via Knockout.

## API Aufbau

### Allgemein wird die API über einen Ajax request angesprochen und erfolgt über den Allgemeinen Pfad (CP) /api\_v1.0.

### API get/delete

#### Zum Auslesen der Daten aus der Datenbank wir zum CP noch /get angehängt.

#### Zum Löschen der Daten aus der Datenbank wir zum CP noch /delete, nicht für Tables, angehängt.

#### Weitere Suboptionen erfolgen über

#### /table/\*tablename\* Liefert die entsprechende Tabelle als JSON

#### /character/\*playerID\*/\*name\* Liefert/Löscht den entsprechenden Character als JSON

#### /monster/\*name\* Liefert/Löscht das entsprechende Monster als JSON

#### /item/\*name\* Liefert/Löscht das entsprechende Item als JSON

### API list /list/\*type\* liefert eine Namensliste aller existierenden Dokumente des Angegebenen Typs, types: armor, shield, weapon, equipment, damageTable, criticalTable, character (for all players), playerID (for characters of player), monster

### API insert

#### Zum Einfügen von Daten aus der Datenbank wir zum CP noch /insert angehängt.

#### Weitere Suboptionen erfolgen über

#### /character trägt das dem AJAX request beigelegte JSON in die Character DB ein.

#### /monster trägt das dem AJAX request beigelegte JSON in die Monster DB ein.

#### /item trägt das dem AJAX request beigelegte JSON in die Item DB ein.

### API update

#### Zum Aktualisieren der Daten aus der Datenbank wir zum CP noch /update angehängt. Die Angehängten Daten müssen das \_id Attribut besitzen.

#### Weitere Suboptionen erfolgen über

#### /character Ersetzt den entsprechenden Charakter mit dem beigefügten JSON

#### /monster Ersetzt das entsprechende Monster mit dem beigefügten JSON /item \* Ersetzt das entsprechende Item mit dem beigefügten JSON

## Monster/Item Editor

### Erstellen von Objekten mit ihren Metadaten.

### Einrichten eines Monster-/Gegenstandbildes

## Monster-Scale

### Scale Up

#### neueDB = alteDB + 2\*(neuesLevel – AltesLevel)

#### neueTreffer =

#### alteTreffer / (altesLevel / (altesLevel + (neuesLevel – altesLevel) \* 2/3) – 10\*(neuesLevel –AltesLevel) > alteTreffer + 10 ?

#### alteTreffer / (altesLevel / (altesLevel + (neuesLevel – altesLevel) \* 2/3) – 10\*(neuesLevel – AltesLevel)

#### :

#### alteTreffer / (altesLevel / (altesLevel + (neuesLevel – altesLevel) \* 2/3)

#### neueOB =

#### alteOB / (altesLevel / (altesLevel + (neuesLevel – altesLevel) \* 1/2) – 7,5\*(neuesLevel – AltesLevel) > alteOB + 40 ?

#### alteOB / (altesLevel / (altesLevel + (neuesLevel – altesLevel) \* 1/2) – 7,5\*(neuesLevel – AltesLevel)

#### :

#### alteOB / (altesLevel / (altesLevel – (neuesLevel – altesLevel) \* 1/2)

### Scale Down

#### neueDB = alteDB + 2\*(neuesLevel – AltesLevel)

#### neueTreffer = alteTreffer \* (neuesLevel / (altesLevel + (neuesLevel – altesLevel) \* 2/3

#### neueOB = alteOB \* (neuesLevel / (altesLevel+ (neuesLevel – altesLevel) \* 1/10

## BattleUI

### Hinzufügen von Spielern / Monstern

### Editieren von Spielern / Monstern

### Anzeige der Kampfteilnehmer

## BattleLog

### Zuweisen von Aktionen für S und M

### Abbrechen von Aktionen

## BattleCalc

### Zuweisen von OB zu DB

### Auswählen von Angriffsart / Reload

### Auswählen von A-Ziel

### Auswählen von D-Ziel

### Angriff abbrechen

### Ablauf der Kampfphasen und Ausführung

## Hauptmenü

### Auswählen von BattleApp, MonsterEditor, ItemEditor, CharacterSetup, CharakterManagement

# Milestones

## GM UI + Server

### Heroku und Cloudant Einrichtung

### Projekterstellung

### Tabellenerstellung

### Hauptmenü

### Monster-Editor

### Item-Editor

### Monster-Scaler

### BattleUI

### BattleLog

### BattleCalc

## Player UI (optional)

### EP Calc

### Login für Spieler

### Charakter Setup und Verwaltung

### Party Lobby

### WW und Patzer

### Spell

## Design Customizing (optional)

### Custom Glyphicons

### Custom Pictures

### Animationen

# Accounts

## Heroku

### Login: [denny@mvh-ag.de](mailto:denny@mvh-ag.de)

### PW: Mortifer1993!

### HerokuLink: <http://merpg.herokuapp.com/>

### Git: git@heroku.com:merpg.git

### API-Key 94e59bb1-8640-4e1a-b327-2ed2d918d74e

## Cloudant

### Name: merpg

### PW: Mortifer1993!

# DB Aufbau

## Monsters

### Enthält alle Monster samt ihrer Daten als Dokumente

### Dokumente

#### { type: string, level: int, speed: int, armor: string, DB: int, OB: int, OBType: string,

#### atSize: string, criticalType: string,

#### descr :string, stunImmun :bool, bloodImmun :bool }

## Characters

### Enthält alle Charaktere samt ihrer Daten als Dokumente

### Dokumente

#### { }

## Tables

### Enthält alle Tabellen die zur Auswertung benötigt werden.

### Dokumente

#### { type: string (damageTable, criticalTable), displayName :string

#### subtype: creature (optional), content (damage): {

#### 1: {

#### armorType: { damage: int, critical: string }, …, armorType: { damage: int, critical: string },

#### } , …, 150: {

#### armorType: { damage: int, critical: string }, …, armorType: { damage: int, critical: string },

#### } , content (crit) : {

#### 1: {

Text: string,  
 hits: int,

Stun: int,  
 activity: int,  
 roundsTillDeath :int,

hitsPerRound :int,

knockedOut :bool,

noHelmet :object,

noArmGreaves :object,

noLeagGreaves :object,

noShield :object

#### } , …, 150: {

Text: string,  
 hits: int,

Stun: int,  
 activity: int,  
 roundsTillDeath :int,

hitsPerRound :int,

knockedOut :bool,

noHelmet :object,

noArmGreaves :object,

noLeagGreaves :object,

noShield :object

} ,  
},

## Items

### Weapon : {

### Name :string,

### Weight: float,

### Price :int, weaponType :string, fumbleRage :int

### attackType :string,

### primaryCritical: {

### type :string, max :string },

### secondaryCritical: {

### type :string, max :string }, type: (weapon)

},

### Shield : {

### Name :string,

### Weight: float,

### Price :int,

### DB : int, type: shield

}

### Armor : {

### Name :string,

### Weight: float,

### Price :int,

### Type: armor,

### armorType: (none, softLeather, rigidLeather, chain, plate), armGreaves :bool,

### legGreaves :bool,

### helmet :bool,

}

# Klassen und Objekte

