**M165 Pokémon Quiz Dokumentation**

Inhaltsverzeichnis

[Unser Thema 2](#_Toc155278554)

[Quiz Ablauf 2](#_Toc155278555)

[DB erstellen 4](#_Toc155278556)

[DB testen 5](#_Toc155278557)

[DB werte lesen in Java 6](#_Toc155278558)

[Datenbank durchlaufen 6](#_Toc155278559)

[Von BSON zu Java 6](#_Toc155278560)

## Unser Thema

Beim Projektauftrag, welcher einen Grossteil unserer Note ausmacht, entschieden wir uns für ein Java basiertes Projekt. Das Projekt: eine Quiz Applikation welche auf eine MongoDB lesend und schreibend zugreifen kann.

Wir wählten als Thema unseres Quiz Pokémon Spielkarten. Die Attribute oder Themenbereiche nach welchen die Fragen gestellt werden, wählten wir wie folgt:

1. HP, Healthpoints
2. DMG, Strength
3. Energy Points
4. Stage, Evolution Stage

Wenn man sich für eine der Kategorien entschieden hat, erhält man eine Frage mit drei zufälligen Pokémons aus der Datenbank.

----------------------------------------------------- Hier kommt der quiz ablauf --------------------------------------------------

## Quiz Ablauf

Beispielsweisse wird dann für das Quiz die Frage kommen für welche Kategorie man befragt werden will, man gibt die Antwort im Terminal ein und bekommt eine Frage mit 3 Pokémons zur Auswahl.

Wir machen mal ein Beispiel Lauf

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

wir wählen die Kategorie «Strength»

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

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungWir bekommen jetzt diese 3 zufällige Pokémons zur Auswahl:

  

! im Spiel hat man keine Bilder zu den Pokémons !

Und wir geben unsere Antwort:

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

Automatisch generierte Beschreibung

Nachdem wir die Antwort geben, wird es aufgelöst, ob wir richtig oder falsch sind und welches die richtige Antwort ist.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

------------------------------------------------- hier endet der quiz ablauf --------------------------------------------------------

## DB erstellen

Mit einem config file erstellen wir die Anforderungen für unsere Mongodb in welcher wir die Pokémons und die dazugehörigen Daten speichern. Diese haben wir in einem init.js File notiert. Beim erstellen des Docker-containers werden die Pokémons mit ihren Werten in der Datenbank gespeichert.

Als Anmeldedaten wählten wir für Nutzernamen: "root" und "1234" als Passwort was auch im config file deffiniert ist.

Mit dem command: "docker-compose up" konnten wir schliesslich den Container erstellen, welcher bei uns auf Docker Desktop am laufen ist.

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

*# Monog DB and Mongo-Express and initial pokemondb*version: '3.9'  
  
services:  
  
 mongodb:  
 image: mongo  
 restart: always  
 ports:  
 - 27017:27017  
 environment:  
 MONGO\_INITDB\_ROOT\_USERNAME: root  
 MONGO\_INITDB\_ROOT\_PASSWORD: 1234  
 volumes:  
 - ./init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro  
  
 mongo-express:  
 image: mongo-express  
 restart: always  
 ports:  
 - 8081:8081  
 environment:  
 ME\_CONFIG\_MONGODB\_ADMINUSERNAME: root  
 ME\_CONFIG\_MONGODB\_ADMINPASSWORD: 1234  
 ME\_CONFIG\_MONGODB\_URL: mongodb://root:1234@mongodb:27017/

### DB testen

Um zu testen ob die Pokémons nun in der Datenbank vorhanden sind, verwendeten wir MongoDB Compass. Mit diesem Tool kann man sich mit der DB verbinden und die Werte betrachten/filtern. Das gleiche ist in IntelliJ selber möglich. Dies ist so möglich, dass man in der IntelliJ auf der rechten Seite Database auswählt. Somit erscheint ein Fenster wo man das "+" Symbol antippt. Man wählt "Data Source" und sucht nach MongoDB.

Damit erscheint erneut ein Fenster mit welchem man sich mit der DB verbindet.

Ein Bild, das Text, Screenshot, Software, Display enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Software, Zahl enthält.

Automatisch generierte BeschreibungUm die Datenbank betrachten zu können, muss man unter anderem Host, Port, URL und in unserem Fall auch User und Passwort angeben. Man kann die Verbindung benennen oder mit einer Farbe markieren.

## DB werte lesen in Java

Um jetzt mit den Pokémons aus der Datenbank arbeiten zu können, muss man die Werte mit Java einlesen. Wir haben als Vorbereitung eine Pokémon Klasse erstellt, um eine Vorlage für die Pokémons in der Datenbank zu haben. Zum Verbinden zur Datenbank haben wir die Mongodb-driver dependency genutzt, welche es uns erlaubt sich mit einer Mongo Datenbank zu verbinden.

String connectionString = "mongodb://root:1234@localhost:27017/?authSource=admin";

Mithilfe der Datenbank URI welche sowohl den Username als auch Password beinhaltete konnten wir uns erfolgreich mit dem MongoClient verbinden. Mit dem Mongoclient verbunden, wählten wir unsere Datenbank und unsere Pokémon collection.

// Connecting to database & get collection  
try (MongoClient mongoClient = MongoClients.*create*(connectionString)) {  
 MongoDatabase database = mongoClient.getDatabase("mongodb");  
 MongoCollection<Document> collection = database.getCollection("pokemons");

### Datenbank durchlaufen

Um drei zufällige Pokémons zu erhalten, wird bei der collection die aggregation "$sample" verwendet. Mit dem Nutzen dieser aggregation ist es möglich, die Anzahl der zufälligen Pokémons zu bestimmen.

var aggregation = Arrays.*asList*(Aggregates.*sample*(3));

Die Werte werden nun in ein BSON Document hinzugefügt. In unserem Fall die drei zufällige Pokémons. Zuerst wird ein "cursor" erstellt, welchem die drei Documents übergeben werden.

### Von BSON zu Java

Nun da man die Werte des Pokémons in einem BSON Document zur verfügung hat, kann man diese in ein Java Objekt umwandeln. Dazu wird in der Methode das Document als Parameter mitgegeben. Mit den Settern werden die Werte gesetzt und ein fertiges Pokémon Objekt ist bereit.

private Pokémon documentToPokemon(Document document) {

Pokémon pokemon = new Pokémon();

pokemon.setName(document.getString("name"));

pokemon.setHealth(document.getInteger("health"));

pokemon.setStrenght(document.getInteger("strenght"));

pokemon.setEnergy(document.getInteger("energy"));

pokemon.setStage(document.getInteger("stage"));

return pokemon;

}