**Documentazione: Setup Backend con MongoDB, Express e Mongoose**

**1. Installazione di MongoDB**

**Download e Installazione**

1. Scaricare MongoDB dalla pagina ufficiale: [MongoDB Download](https://www.mongodb.com/try/download/community)
2. Installare scegliendo l'opzione **MongoDB as a Service**
3. Verificare l'installazione aprendo un terminale e digitando:
4. mongod --version
5. Avviare il servizio MongoDB: creazione aurandor\_rpg dalla lista database a sinistra

**2. Configurazione del Progetto Node.js**

**Inizializzazione del progetto**

1. Creare una cartella per il progetto e inizializzare Node.js:
2. mkdir aurandor\_rpg
3. cd aurandor\_rpg
4. npm init -y
5. Installare le dipendenze principali:

npm install express mongoose

**3. Struttura del Progetto**

aurandor\_rpg/

├── config/

│ ├── db.js

├── controllers/

│ ├── characterController.js

├── models/

│ ├── Character.js

├── routes/

│ ├── characterRoutes.js

├── server.js

├── .env

├── package.json

**4. Configurazione di MongoDB**

**Creare il file config/db.js**

const mongoose = require('mongoose');

require('dotenv').config();

const connectDB = async () => {

try {

await mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log('MongoDB connesso!');

} catch (error) {

console.error('Errore di connessione:', error);

process.exit(1);

}

};

module.exports = connectDB;

**Creare il file .env**

MONGO\_URI=mongodb://localhost:27017/aurandor\_rpg

PORT=3000

**5. Creazione del Modello Character**

**models/Character.js**

const mongoose = require('mongoose');

const CharacterSchema = new mongoose.Schema({

name: String,

race: String,

class: String,

alignment: String,

description: String,

abilities: [{ name: String, power: Number }],

relationships: Object,

inventory: [{ item: String, quantity: Number }],

image: String,

attack: {

weapon: String,

damage: Number

}

});

module.exports = mongoose.model('Character', CharacterSchema);

**6. Creazione del Controller**

**controllers/characterController.js**

const Character = require('../models/Character');

exports.getAllCharacters = async (req, res) => {

try {

const characters = await Character.find();

res.json(characters);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.addCharacter = async (req, res) => {

try {

const newCharacter = new Character(req.body);

const savedCharacter = await newCharacter.save();

res.json(savedCharacter);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.updateCharacter = async (req, res) => {

try {

const updatedCharacter = await Character.findByIdAndUpdate(req.params.id, req.body, { new: true });

res.json(updatedCharacter);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.deleteCharacter = async (req, res) => {

try {

await Character.findByIdAndDelete(req.params.id);

res.json({ message: 'Personaggio eliminato' });

} catch (err) {

res.status(500).json({ error: err.message });

}

};

**7. Creazione delle Rotte**

**routes/characterRoutes.js**

const express = require('express');

const router = express.Router();

const characterController = require('../controllers/characterController');

router.get('/', characterController.getAllCharacters);

router.post('/', characterController.addCharacter);

router.put('/:id', characterController.updateCharacter);

router.delete('/:id', characterController.deleteCharacter);

module.exports = router;

**8. Configurazione del Server**

**server.js**

const express = require('express');

const cors = require('cors');

const bodyParser = require('body-parser');

const connectDB = require('./config/db');

const characterRoutes = require('./routes/characterRoutes');

require('dotenv').config();

const app = express();

connectDB();

app.use(cors());

app.use(bodyParser.json());

app.use('/characters', characterRoutes);

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server in ascolto su http://localhost:${PORT}`);

});

**9. Test delle API**

Utilizzare **Postman** per testare:

**GET - Recupera tutti i personaggi**

GET http://localhost:3000/characters

**POST - Aggiungi un nuovo personaggio**

POST <http://localhost:3000/characters>

Body>raw>json

'{"name":"Inti","race":"Umano","class":"Mago","alignment":"Neutrale","description":"Sciamano potente","attack":{"weapon":"Bastone","damage":25}}'

**PUT - Modifica un personaggio**

PUT http://localhost:3000/characters/ID\_DEL\_PERSONAGGIO   
Body>raw>json

{"attack":{"weapon":"Spada","damage":40}}'

**DELETE - Elimina un personaggio**

DELETE http://localhost:3000/characters/ID\_DEL\_PERSONAGGIO