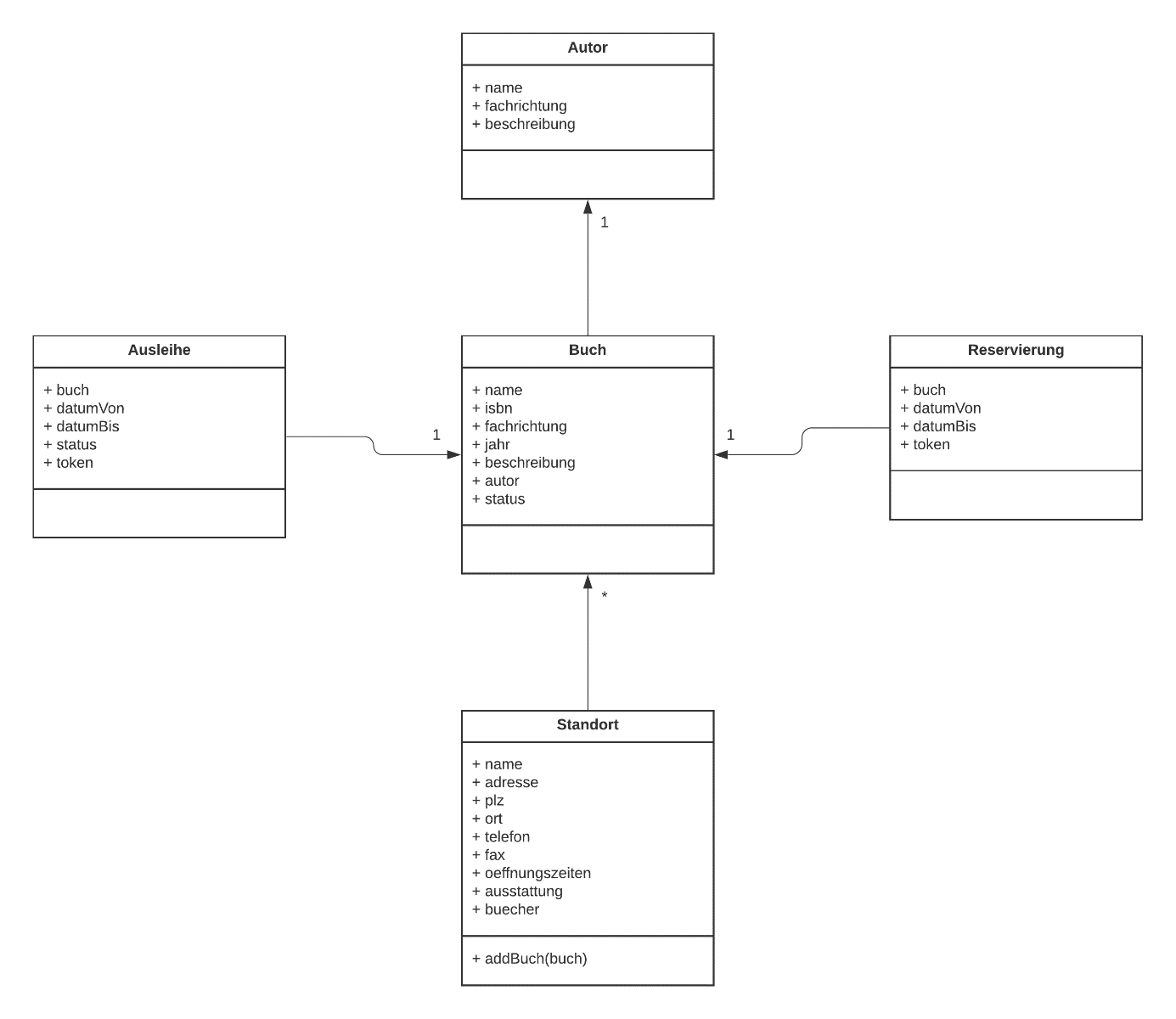
**1. Beschreibung des Anwendungsszenarios**

Hier einen Text einfügen

**2. Fachobjekte der Anwendung**



Hier noch zusätzlichen Erklärungstext einfügen.

**3. Beschreibung und Begründung der Implementierungsentscheidungen**

**3.1. Client**

**3.1.1. Strukturierung der HTML-Seiten**

**3.1.2. Layout**

**3.1.3. Responsivität**

**3.2. Server**

**3.2.1. Modulstruktur**

Grundsätzlich wurde das Projekt nach dem MVC (Model View Controller) Muster aufgebaut. Hierbei werden die verschiedenen Zuständigkeiten sauber in eine Modell-Schicht (Ordner „models“), eine Präsentationsschicht (Ordner „views“) und eine Steuerungsschrift (Ordner „controller“) unterteilt. Dies hat vor allem eine saubere Trennung von Zuständigkeiten und eine hohe Wartbarkeit zur Folge.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Quelle: <https://commons.wikimedia.org/wiki/File:ModelViewControllerDiagram2.svg>

Die Modell-Schicht repräsentiert die Datenhaltung bzw. die Datenbank. Für jede Objektklasse wurde ein eigenes Modell erstellt, um eine noch bessere Übersichtlichkeit und Wartbarkeit zu realisieren. Die Brücke zwischen allen Modellen bildet hierbei die Datei „models.js“. Hier werden auf Basis der einzelnen Modelle erste Objekte erzeugt und diese in das entsprechende Array des Modells geladen, um beim Start der App einen ersten Datensatz zu realisieren.

Die Präsentationsschicht ist zuständig für die Darstellung der Seiten. Die Templates wurde mithilfe der Template Engine EJS (<https://ejs.co/>) realisiert. So lassen sich dynamische Inhalte einfacher darstellen und die Möglichkeit des Einbindens von Templates (in Fall der App ist dies der Header, der Footer und die Sidebar) erhöht die Wartbarkeit enorm.

Die Steuerungsschicht repräsentiert die Verbindung zwischen der Modell-Schicht und der Präsentationschicht. Der Controller importiert alle Modelle (bzw. die „Datenbank“) und kann somit Änderungen an diesen vornehmen. Außerdem sorgt der Controller für eine dynamische Ausgabe von Inhalten. Innerhalb des Controllers gibt es für jede Objektklasse passende Methoden (Darstellung der Übersicht, Darstellung der Detailseite, Hinzufügen eines Objektes), die innerhalb des Routings einfach aufgerufen bzw. ausgeführt werden können, die die oben genannten Funktionen realisieren.

Den Ausgangspunkt der Applikation bildet jedoch die Datei app.js. Hier wird der Webserver mithilfe von Express konfiguriert (über die Middleware) und gestartet. Alle benötigten Node.js-Module werden hier importiert. Auch das Routing wurde als eigenes Modul ausgelagert und wird hier importiert und über eine geeignete Middleware als solches definiert.

Das Routing, welches über die Datei router.js repräsentiert wird, importiert das benötigte Express-Modul, startet die Router-Funktion von Express und bindet diese an die Konstante „router“ und lädt den oben genannten Controller. Innerhalb des Routings werde die aufzurufenden URLs abgefangen. Gegebenenfalls werden nun Controller-Funktionen aufgerufen, zum Beispiel um eine Objektinstanz hinzuzufügen. Ein Statuscode wird gesetzt und das zur URL gehörige Template (EJS, siehe oben) wird gerendert. Innerhalb der Render-Funktion werden dem Template noch Variables mithilfe des Controllers übergeben, welche für eine dynamische Inhaltsausgabe sorgen.

**3.2.2. Unterstütze URLs/Endpunkte**

|  |  |  |
| --- | --- | --- |
| **URL** | **HTTP-Methode** | **Aktion** |
| **/** | **GET** | **Anzeige des Dashboards** |
| **/login** | **GET** | **Anzeige einer Loginmaske** |
| **/logout** | **GET** | **Anzeige einer Logoutseite** |
| **/impressum** | **GET** | **Anzeige der Impressumsseite** |
| **/datenschutz** | **GET** | **Anzeige der Datenschutzseite** |
| **/autoren** | **GET** | **Anzeige der Autorenübersicht** |
| **/autoren** | **POST** | **Neuanlage eines Autors** |
| **/autor/detail** | **GET** | **Anzeige der Detailseite eines Autors** |
| **/autor/neu** | **GET** | **Anzeige des Anlegeformulars eines Autors** |
|  |  |  |

Hier muss die Liste noch ergänzt werden!!!

**4. Start der Anwendung**

Die App benötigt die JavaScript-Laufzeitumgebung „Node.js“ (https://nodejs.org/en/). Diese bildet die Basis für den Webserver und die Applikation.

Zum Start der Anwendung man das Verzeichnis über die Konsole mithilfe des cd-Befehls („change directory“) aufrufen. Anschließend wird über den Befehl „node app.js“ die Anwendung gestartet. Das Node.js-Modul „open“ sorgt für ein automatisches Öffnen Root-URL im Standard-System-Browser. Alternativ kann die Applikation auch einfach über die URL <http://localhost:8040> aufgerufen werden.

**5. Autorenschaft der Quellcode-Dateien**