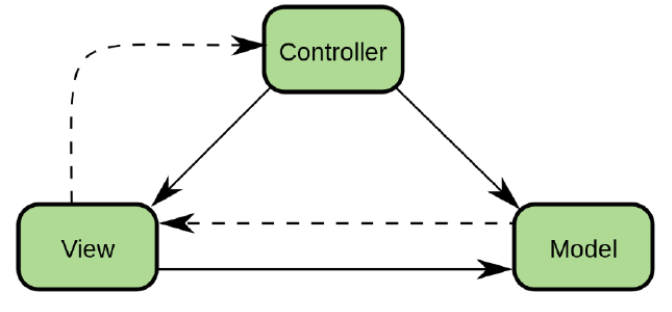
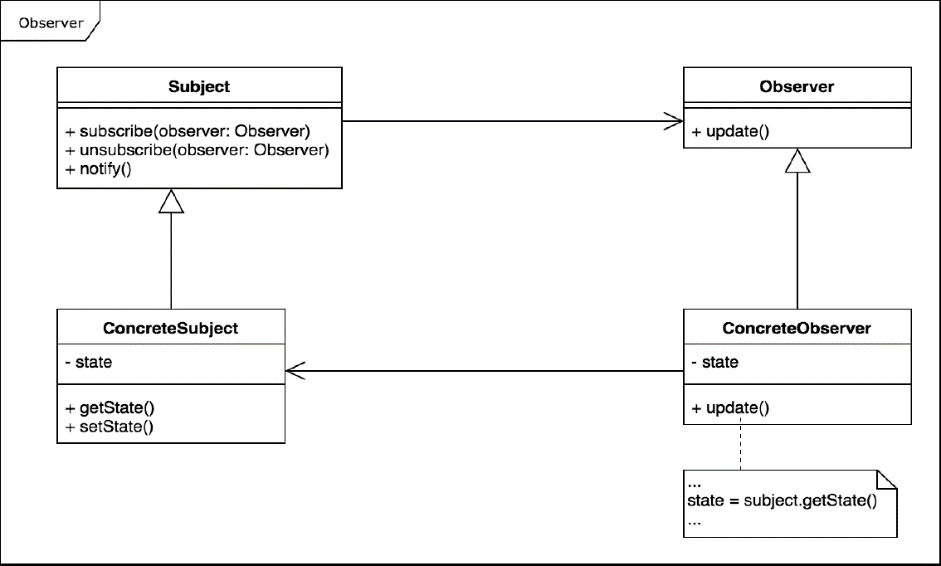
## Der Lernende kennt das MVC-Muster und weiss aus welchen Mustern dieses zusammengesetzt ist.

Eine grundlegende Regel in einer dreischichtigen Architektur ist, dass die Client-Schicht niemals direkt mit der Datenschicht kommuniziert. In einem dreistufigen Modell muss die gesamte Kommunikation über die mittlere Ebene erfolgen. Die Ansicht sendet Aktualisierungen an den Controller, der Controller aktualisiert das Modell und die Ansicht wird direkt vom Modell aktualisiert.

## Der Lernende kennt das Observer-Muster und weiss wo dieses im MVC-Muster eingesetzt wird.

**Zweck:** Mithilfe des Observer Pattern definiert man eine Abhängigkeit zwischen einem und beliebig vielen anderen Objekten, sodass die Änderung des Zustands des einen Objekts dazu führt, dass alle abhängigen Objekte automatisch benachrichtigt werden.

Das MVC-Muster lässt sich als zusammengesetztes Muster, bestehend aus Observer-, Strategy und Composite-Muster beschreiben. Das Model (Observable oder Subjekt) implementiert das Observer-Muster. Interessierte Objekte, wie die View (Observer), können sich registrieren und werden über Zustandsänderungen informiert.

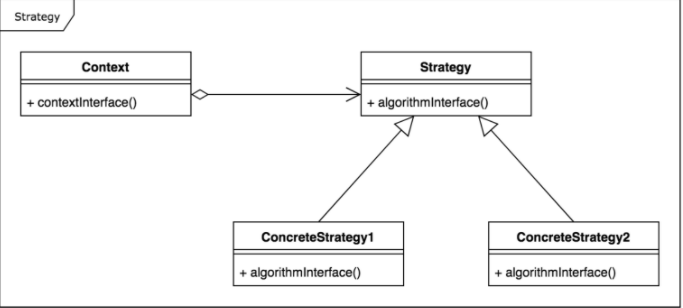
Vorteile:

* Zustandkonsistenz
* Observer können jederzeit hinzugefügt oder entfernt werden
* Subjekte und Beobachter sind lose gekoppelt

Nachteile:

* Hohe Anzahl an Beobachtern -> bei Änderungen hohe Änderungskosten
* Quellcode des Subjekts zeigt incht, welche Beobachter genau informiert werden

## Der Lernende kennt das Strategy-Muster und weiss wo dieses im MVC-Muster eingesetzt wird.

**Zweck:** Durch das Strategy Pattern wird eine Familie von Algorithmen definiert, die einzeln gekapselt und austauschbar sind. Somit kann zur Laufzeit der gewünschte Algorithmus ausgeführt werden. Mit dem Strategy Pattern sorgen wir dafür, dass diese einzelnen Algorithmen für sich gekapselt werden und damit austauschbar sind. Damit können wir zur Laufzeit immer den gewünschten Algorithmus ausführen den wir brauchen.

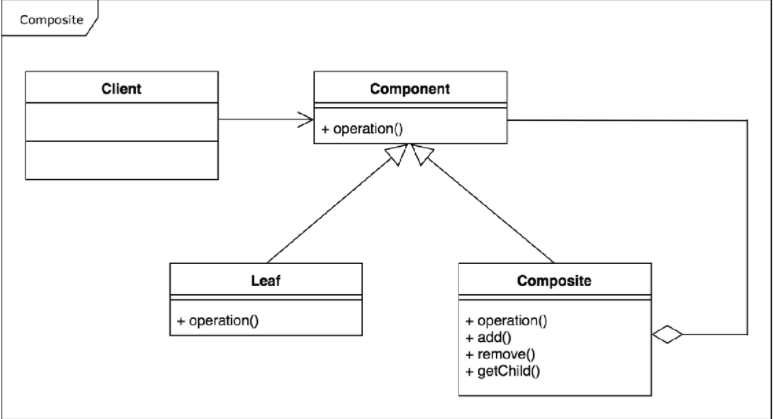
Vorteile:

* Erweiterbarkeit
* Algorithmen können zur Laufzeit gewechselt werden
* Algorithmen sind in eigenen Klassen gekapselt

Nachteile:

* Zusätzlicher Kommunikationsaufwand zwischen Strategie und Kontext

## Der Lernende kennt das Composite-Muster und weiss wo dieses im MVC-Muster eingesetzt wird.

**Zweck:** Das Composite Pattern stellt einen Mechanismus zur Verfügung, um einfache Objekte und komplexe zusammengesetzte Objekte einheitlich zu behandeln. Die Objekte werden zu sogenannten Baumstrukturen zusammengefügt, um so Teil-Ganze Hierarchien zu repräsentieren.

Vorteile:

* Einheitliche Behandlung von primitiven und Kompositionen
* Flexibilität und Erweiterbarkeit
* Verschachtelte Strukturen mit beliebiger Tiefe möglich

Nachteile:

* Ein zu allgemeiner Entwurf erschwert es, Kompositionen auf bestimmte Klassen zu beschränken

## Der Lernende weiss, wo Benutzereingaben im MVC-Muster entgegengenommen werden.

Ein Button-Click wird beispielsweise durch den Controller in einen entsprechenden Methodenaufruf zum Schreiben von geänderten Daten im Model übersetzt. Man beachte, dass Benutzereingaben und -aktionen direkt vom Controller (und nicht vom View) entgegengenommen werden.

## Der Lernende weiss wie Datenbank Abfragen/Befehle verarbeitet werden.

Grundsätzlich wird jede einzelne Abfrage/Befehl sequenziell abgearbeitet. Sequenziell bedeutet es bearbeitet die Ausgabeaufträge in der Reihenfolge ihrer Erzeugung.

## Der Lernende kennt den Nutzen der Transaktionen.

Transaktionen definieren was alles zusammen in einem Durchgang ausgeführt werden sollte. Zudem kann bei Fehlern während einer Transaktion diese Zurückgespielt werden, dies nennt man Rollback.

**Wichtig:** Eine Transaktion führt ihre Aktionen im Prinzip **Alles oder Nichts** aus!

## Der Lernende kennt die SQL-Befehle (PostgreSQL) die bei einer Transaktion angewendet werden.

Mit dem Keyword **BEGIN** wird eine Transaktion gestartet. Alles was zwischen dem Start und dem **COMMIT** ausgeführt wird, wird innerhalb einer Transaktion getätigt. Mittels **ROLLBACK** kann es rückgängig gemacht werden. Wenn man es genau möchte, kann mittels **SAVEPOINT** einen Savepoint erstellt werden.

## Der Lernende kennt die Möglichkeit einen zu ändernden Datensatz auf Lesezugriffe zu blockieren bis die Änderung vollzogen ist. (PostgreSQL)

-

## Der Lernende kennt Schreib- und Lese-Lock und kann diese anwenden.

-

## Der Lernende weiss, was unter Atomarität verstanden wird.

Eine Transaktion wird global erst «für gültig erklärt und in Kraft gesetzt», wenn sie erfolgreich vollständig abgeschlossen sind. Sollte sich jedoch während der Transaktion herausstellen, dass diese nicht vollständig abgeschlossen werden kann, wird der ursprüngliche Bereich als gültig erklärt und ein [Rollback](https://de.wikipedia.org/wiki/Rollback) durchgeführt

## Der Lernende weiss, was unter Konsistenz verstanden wird.

Heisst, dass eine Transaktion nach Beendigung einen konsistenten Datenbankzustand hinterlässt, falls die Datenbank davor auch konsistent war. Dies beinhaltet, dass alle im Datenbankschema definierten [Integritätsbedingungen](https://de.wikipedia.org/wiki/Integrit%C3%A4tsbedingung) vor dem Abschluss der Transaktion überprüft werden. Ist das nicht möglich, oder tritt ein Fehler auf, wird die gesamte Transaktion rückgängig gemacht.

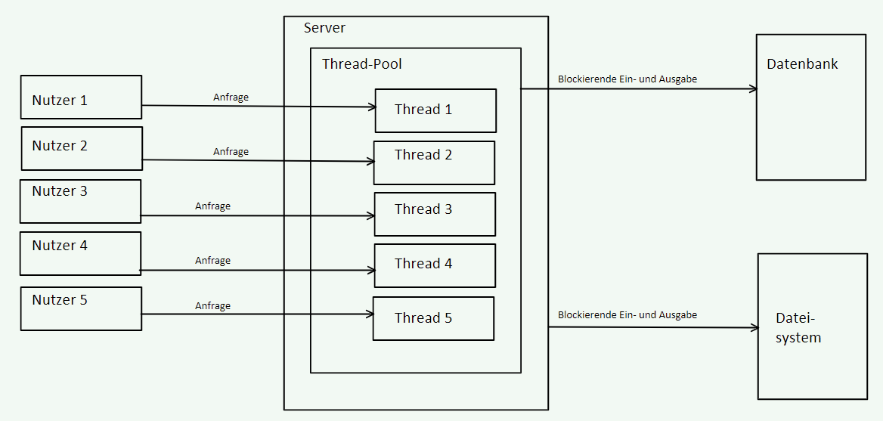
1. Der Lernende weiss, was unter Isolation verstanden wird

Wird verhindert/eingeschränkt, dass sich nebenläufig in Ausführung befindliche Transaktionen gegenseitig beeinflussen. Realisiert wird dies üblicherweise durch Sperrverfahren, die vor einem Datenzugriff die benötigten Daten für andere Transaktionen sperren. Verbreitete Einstellungen sind dabei READ COMITTED, REPEATABLE READ, SERIALIZABLE.

## Der Lernende weiss, was unter Dauerhaftigkeit verstanden wird.

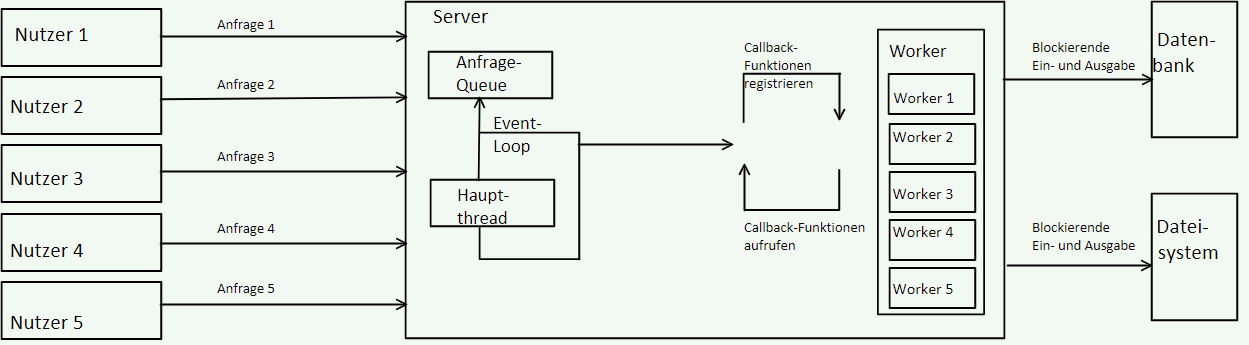
Der Begriff Dauerhaftigkeit sagt aus, dass Daten nach dem erfolgreichen Abschluss einer Transaktion garantiert dauerhaft in der Datenbank gespeichert sind. Die dauerhafte Speicherung der Daten muss auch nach einem Systemfehler (Software-Fehler oder Hardware-Ausfall) garantiert sein.  Insbesondere darf es nach einem Ausfall des Hauptspeichers nicht zu Datenverlusten kommen. Dauerhaftigkeit kann durch das Schreiben eines [Transaktionslogs](https://de.wikipedia.org/wiki/Logdatei) sichergestellt werden.

## Der Lernende weiss wie Multi-Threaded Server aufgebaut sind.

Bei diesen Servern wird bei jeder HTTP-Anfrage eines Nutzers an den Server auf Serverseite ein neuer Thread erzeugt und die HTTP-Anfrage innerhalb dieses Threads bearbeitet. Daher nennt man diese Arten von Servern auch**Multi-Threaded Server**. Solange der Thread wartet, belegt er trotzdem Speicherplatz. Das wiederum führt bei vielen Anfragen an den Server (die blockierende Operationen zur Folge haben) zu einer höheren Speicherauslastung des Servers

## Der Lernende weiss wie Single-Threaded Server aufgebaut sind.

Es gibt einen Thread der alle Anfragen entgegennimmt und in einer sogenannten Anfrage Queue verwaltet. Aufgrund dieser Tatsache wird Node.js auch **Single-Treaded Server** genannt.



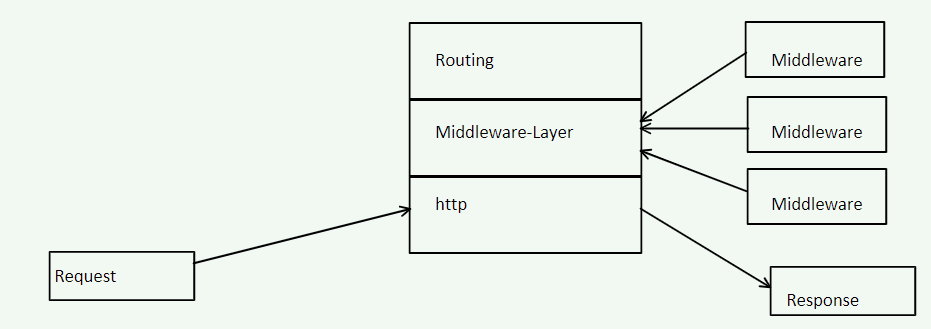
## Der Lernende kennt den Aufbau des Single-Threaded Server von Node.js

Statt jeweils einen Thread pro eingehender Anfrage vom Client zu erzeugen, verwendet Node.js

nur einen einzelnen (Haupt-)Thread, der alle Anfragen entgegennimmt und in einer sogenannten Anfrage-Queue verwaltet. Aufgrund dieser Tatsache wird Node.js (bzw. ein damit erzeugter Webserver) auch **Single-Threaded Server** genannt. Der Cloud dabei ist die sogenannte Event-Loop. Es handelt es sich um eine interne Schleife, die kontinuierlich Anfragen aus der Anfrage-Queue prüft und Ereignisse von Ein- und Ausgabeoperationen verarbeitet.

## Der Lernende kennt den grundsätzlichen Aufbau von Express.

**Express** ist ein kompaktes Framework mit einem überschaubaren Funktionsumfang. Es lässt sich jedoch mit sogenannten Middleware-Komponenten gut erweitern. Der Aufbau von Express weist, ähnlich wie Node.js selbst, einen mehrschichtigen Aufbau auf.



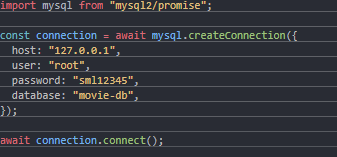
## Der Lernende kennt das Middleware-System von Express und weiss wie Middleware eingebunden werden.

Die zweite Ebene der Architektur von Express stellt der Middleware-Layer dar. Im Kontext von Express ist eine Middleware eine Funktion, die zwischen der eingehenden Anfrage und der Antwort des Servers an den Client steht. Mehrere Middleware- Funktionen können zu einer Kette zusammengesetzt werden, um basierend auf der Anfrage des Clients bestimmte Aktionen auszuführen.

## Der Lernende kennt die unterstützten Routing-Funktionen von Express.

Die dritte Ebene der Architektur von Express bildet der Router. Diese Komponente von Express steuert, welche Funktion abhängig von der aufgerufenen URL ausgeführt werden soll, um eine Antwort an den Client zu generieren. Beim Routing werden sowohl die HTTP-Methode als auch der URL-Pfad betrachtet.

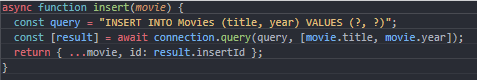
## Der Lernende weiss wie in Node.js eine Verbindung zur MySQL Datenbank im Model konfiguriert und eine Verbindung zur Datenbank aufgebaut wird.



## Der Lernende kennt die query-Methode der Promise-Schnittstelle des mysql2-Moduls (Node.js) und kann diese anwenden.

## import mysql from "mysql2/promise";

## Der Lernende weiss wie man SQL-Injection durch Platzhalter und Escaping verhindern kann (Node.js).



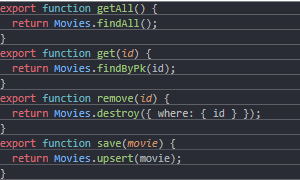
In dem Insert-Statement werden Platzhalter für die einzufügenden Werte gesetzt. Die Fragezeichen bei «VALUES» sind hier die Platzhalter. Der Datenbank-Treiber kümmert sich danach um das Escaping der Werte. Die zu ersetzenden Werte werden im Parameter in einem Array zur Verfügung gestellt.

## Der Lernende weiss wie das Problem: Objekte und Daten der relationalen Datenbank gelöst wurde (ORM).

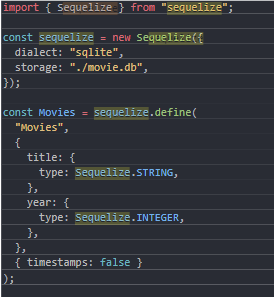
Eine Möglichkeit, das Mapping von Objekten auf Relationen und umgekehrt zu automatisieren, sind sogenannte Object-Relational Mappings (kurz ORM). Innerhalb eines Programms definieren Sie beim Zugriff auf die Daten nicht direkt SQL-Statements der jeweiligen Datenbankanfragensprache, sondern arbeiten direkt mit Objekten, die den Zugriff auf die Datenbank abstrahieren.

## Der Lernende kennt die Einsatzmöglichkeit von ORM mit SQLite in Node.js.





## Der Lernende weiss wie man ein einfaches (1-Tabelle) Model mit Sequelize (Node.js) definiert.



## Der Lernende kennt die vier von uns verwendeten Methoden im ORM (Sequelize, Node.js) findAll, findByPk, destroy und upsert und weiss was diese machen.

* **findAll**: Liesst alle Datensätze auf Basis des Models aus der Datenbank aus
* **findByPk**: Liesst den Datensatz, welcher mit der mitgegebenen ID übereinstimmt, aus der Datenbank aus.
* **Destroy**: Löscht den Datensatz auf Basis des Models aus der Datenbank raus.
* **Upsert**: Die upsert Methode kann zwischen der einfügenden und aktualisierenden Operation unterscheiden. Entweder wird ein neuer Datensatz eingefügt oder aktualisiert.

## Der Lernende kann Kriterien zur Bestimmung schutzwürdiger Informationen wie besondes schützenswerter Personendaten und sensibler Geschäftsdaten nennen.

-

## Der Lernende kann das Vorgehen zur Gewährleistung der Datenintegrität, Verfügbarkeit, Authentizität und Vertraulichkeit in Web-Applikationen beschreiben.

Um die Sicherheit einer Web-Applikation sicherzustellen, müssen folgende Gefahren ausgeschaltet werden:

* **Unbefugter Datenzugriff:** Unbefugte Personen können Daten einsehen, die nicht für sie bestimmt sind.
* **Unerlaubte Datenmanipulation:** Unbefugte Personen können Daten verändern oder löschen, die nicht für sie bestimmt sind.
* **Verlust der Verfügbarkeit:** Daten oder Applikationen sind im gewünschten Zeitraum nicht verfügbar.

## Der Lernende kann die Sicherheitsrisiken sowie organisatorische und technische Sicherheitsvorkehrungen zur Sicherheit von Informationen / Daten und Web-Applikationen aufzählen.

Organisatorische Mängel:

* Fehlende Planung und Abstimmung des Gesamtsystems
* Mangelhaftes Design einer neuen Applikation
* Keine oder mangelhafte Regelung der Zugriffsberechtigungen
* Keine oder mangelhafte Datenpflege
* Keine oder mangelhafte Kontrolle der Schutzmassnahmen

Technische Mängel:

* **Konzeptionelle Fehler:** Die Konzeption der Web-Applikation entählt bereits Fehler. Wird ein fehlerhaftes Softwarekonzept umgesetzt, können Sicherheitslücken entstehen.
* **Fehlerhafte Programmierung:** Die eingesetzte Webserver-, Datenbank- oder E-Business-Software wurde fehlerhaft programmier oder ist untereinander inkompatibel.
* **Fehlerhafte Konfiguration:** Die eingesetzte Webserver-, Datenbank- oder E-Business-Software wurde fehlerhaft konfiguriert bzw. parametrisiert.
* **Fehlende oder fehlerhafte Daten:** Die auf dem Webserver oder in der Datenbank gespeicherten Daten fehlen (Datenverlust) oder sind mangelhaft (Datenintegrität).

## Der Lernende kann auflisten, welche Methoden der Datenverschlüsselung und der Gewährleistung der Authentizität (HTTPS, Zertifikate) existieren.

-

## Der Lernende kann Testverfahren erläutern, um die Lösungsansätze gegen Cross-Site-Scripting und Script Injection (SQL Injection) zu prüfen.

Die Entwickler sollten die Eingaben der Anwender auf bestimmte Sonder- beziehungsweise Metazeichen filtern. Durch das Kodieren dieser Eingaben können sowohl persistente als auch reflektierte Attacken verhindert werden. Die folgenden Sonderzeichen sollten immer ausgefiltert werden, damit sie nicht für Attacken missbraucht werden können: <, >, «, ‘, (, ), ;, /, #, &

## Der Lernende kennt des OWASP-Top-10-Projekt.

Es ist ein wesentliches Ziel der OWASP Top 10, Entwickler, Designer, Architekten und Führungskräfte von Organisationen und Unternehmen über die Risiken der wichtigsten Schwachstellen von Webanwendungen aufzuklären. Die Top 10 stellen grundlegende Techniken zum Schutz gegen diese hochriskanten Probleme vor. Sie zeigen auch auf, wie es danach weitergeht.