



# Dokumentation Modul

# 165

---

Lekhana Godugunuri  
Thasmini Thayaparan  
Masha Madani


---

Ausstellungsstrasse 70  
8005 Zürich



# Inhaltsverzeichnis

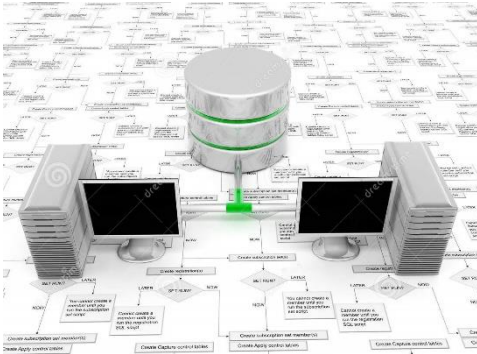
1.	Grundwissen .....	4
1.1	Datenbank .....	4
1.2	DBMS .....	4
2.	SQL vs. NoSQL .....	5
2.1	SQL .....	5
2.2	NoSQL .....	5
2.3	SQL vs NoSQL .....	5
2.4	Skalierung .....	6
3.	JSON vs. XML .....	7
3.1	JSON .....	7
3.2	XML .....	7
3.3	Ähnlichkeiten .....	7
3.4	Differenzen .....	7
4.	Open Data .....	8
5.	Index .....	8
5.1	Bitmap Index .....	9
5.2	Clustered Index .....	9
5.3	Non - Clustered Index .....	10
5.4	Funktionaler Index .....	11
5.5	Reverse Index .....	11
5.6	Partitionierter Index .....	11
6.	MariaDB .....	11
7.	Cassandra .....	12
7.1	Vorteil und Nachteil .....	12
8.	Transaktionen .....	12
8.1	ACID .....	13
	<a href="https://static1.tothenew.com/blog/wp-content/uploads/2016/06/img3.png">https://static1.tothenew.com/blog/wp-content/uploads/2016/06/img3.png</a> .....	13
8.2	CAP .....	14
8.3	BASE .....	14
9.	Replication .....	15
9.1	Replikation (Datenverarbeitung) .....	15
9.2	Cassandra Replikation .....	16
10.	Einleitung .....	17
10.1	Vorwort .....	17
10.2	Warum dieses Projekt .....	17
10.3	Wie sind wir vorgegangen .....	17



10.4	Verwendete Methoden.....	18
	Unser Prozess.....	23
10.5	Hauptteil .....	24
10.6	Was sind unsere Befürchtungen .....	24
10.7	Was wir erhoffen .....	24
10.8	Verwendete Tools.....	24
11.	Schluss .....	25
11.1	Endergebnisse.....	25
11.2	Quellenverzeichnis.....	25
11.3	Reflektion .....	25

# 1. Grundwissen

## 1.1 Datenbank



Grundsätzlich sind Datenbanken, Dateien oder sogar ganze Partitionen auf einem System, in welchen Daten geordnet gespeichert werden können. Somit unterscheiden sich diese Dateien auf den ersten Blick nicht gross von einem Word oder einem Excel File. Die Art jedoch, wie in einer Datei die Werte strukturiert und abgelegt werden, macht den grossen Unterschied und ist abhängig vom jeweiligen Produkt und Hersteller. Datenbanken haben den Vorteil, dass man die Inhalte strukturieren, sortieren und analysieren kann.

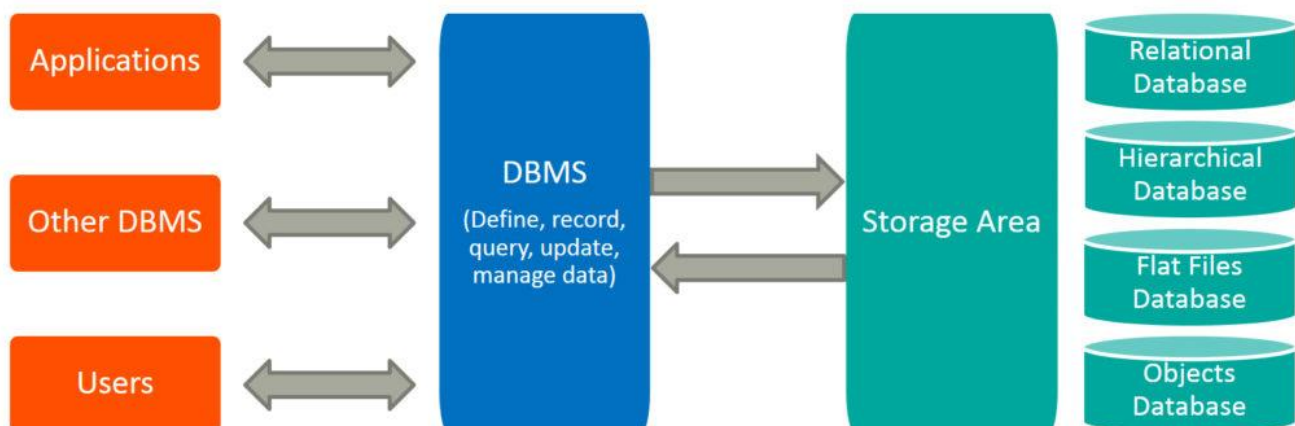
<https://thumbs.dreamstime.com/z/datenbank-3d-19909914.jpg>

## 1.2 DBMS

Ein Datenbankmanagementsystem, kurz DBMS, bildet gemeinsam mit der Datenbasis ein Datenbanksystem. Erst wenn das jeweilige Datenbankmanagementsystem installiert und eingerichtet ist, können Nutzer den gewünschten Datenbestand einfügen und auslesen lassen. Die bekannteste dieser Sprachen ist SQL.

<https://www.tekportal.net/wp-content/uploads/2019/01/dbms-3675.jpg>

## Database Management System



## 2. SQL vs. NoSQL

### 2.1 SQL

SQL ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen. Die Sprache basiert auf der relationalen Algebra, ihre Syntax ist relativ einfach aufgebaut und semantisch an die englische Umgangssprache angelehnt.



<https://th.bing.com/th/id/R.c896d903d562cedad6d7601e0356c1e4?rik=Hupv2Heoaf%2f4bg&pid=ImgRaw&r=0>

### 2.2 NoSQL

NoSQL-Datenbanken verwenden verschiedene Datenmodelle für den Zugriff auf und die Verwaltung von Daten. Diese Typen von Datenbanken sind speziell für Anwendungen optimiert, die grosse Datenmengen, geringe Latenz sowie flexible Datenmodelle erfordern.

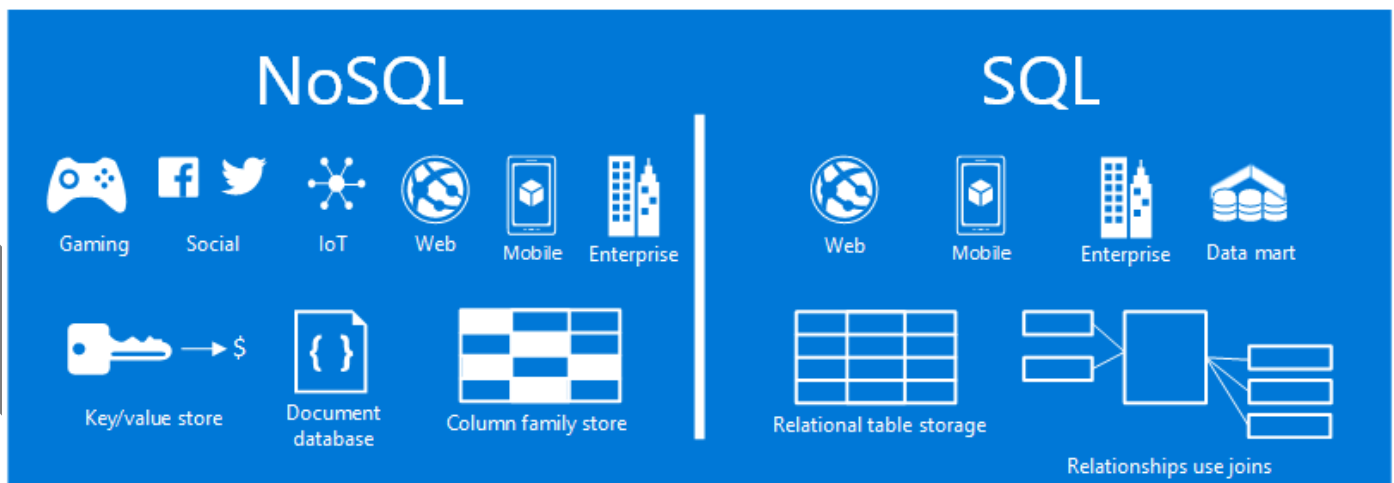
<https://www.leixue.com/uploads/2019/05/NoSQL.png>



### 2.3 SQL vs NoSQL

#### Übersicht:

Hier werden kurz aufgelistet, in welchen Bereichen NoSQL und SQL-Datenbanken verwendet werden und auch was für Modelle die jeweiligen Datenbankarten haben.



<https://th.bing.com/th/id/R.5ab92bf68438b82a7f73ce8a68b0f082?rik=VnyW9nWjSzaJgQ&pid=ImgRaw&r=0>

## Vergleich:

Diese Tabelle zeigt kurz und knapp die Eigenschaften beider Datenbankarten und somit auch den Vergleich und die Unterschiede.

	SQL	NoSQL
Beschreibung	relational	nicht-relational
Anwendung	Abfrage zum Analysieren und Abrufen von Daten	für eine Vielzahl moderner Anwendungen wie WebApps geeignet
Abfragesprache	SQL	mehrere Sprachen je nach Anwendung
Typ	Tabelle	Dokument / Graph / Key-Value
Schema	festgelegt und vordefiniert	dynamisch
Data Management System (Beispiele)	Oracle, PostGres, MySQL	MongoDB, Neo4J
Eignet sich für	komplexe und intensive Abfragen	Große Datenbanken, Big Data
Entwicklungsjahre	70er Jahre	2000er
Open Source	Open Source (PostGres, MySQL) und proprietäre Systeme (Oracle)	Open Source
Vorteile	optimierte Datenspeicherung und Stabilität	einfache und flexible Speicherung
Nachteile	keine Flexibilität, erforderliche Expertise	manchmal zu flexibel

[SQL vs. NoSQL: Unterschiede, Anwendungen, Vor- und Nachteile. \(datascientest.com\)](https://datascientest.com/)

## 2.4 Skalierung

NoSQL:

- horizontale Skalierung
- verteilt die Daten auf mehrere Server
- erhöht sich die Datenmenge, so werden einfach neue Server hinzugefügt
- Jeder Administrator kann neue Commodity- und Cloud-Server hinzufügen, die NoSQL-Datenbank sendet die Daten automatisch an alle Server.

SQL:

- Vertikale Skalierung
- Stützen ihre gesamte Leistungskraft auf einen einzelnen Server
- Um ihre Kapazität zu erhöhen, müsste in einen stärkeren Server investiert werden à teuer und schränkt auch die Möglichkeiten in der Anwendungsentwicklung ein
- Ein einzelner Server muss die Leistung des kompletten Datenbanksystems tragen, was einen Leistungsabfall bei grossen Datenmengen zur Folge hat.

## 3. JSON vs. XML

Sowohl JSON als auch XML können verwendet werden, um Daten von einem Webserver zu empfangen.

[JSON vs XML \(w3schools.com\)](https://www.w3schools.com/json/)

### 3.1 JSON

#### JSON Example

```
{ "employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}
```

### 3.2 XML

#### XML Example

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

### 3.3 Ähnlichkeiten

- Sowohl JSON als auch XML sind "selbstbeschreibend" (menschenslesbar)
- Sowohl JSON als auch XML sind hierarchisch (Werte innerhalb von Werten)
- Sowohl JSON als auch XML können von vielen Programmiersprachen analysiert und verwendet werden
- Sowohl JSON als auch XML können mit einer XMLHttpRequest abgerufen werden

### 3.4 Differenzen

- JSON verwendet kein End-Tag
- JSON ist kürzer
- JSON ist schneller zu lesen und zu schreiben
- JSON kann Arrays verwenden

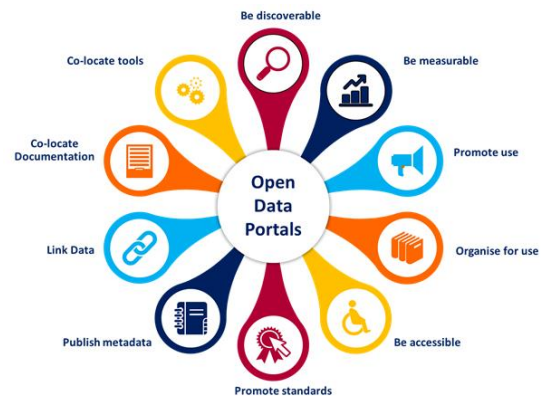
➔ Der größte Unterschied ist:

XML muss mit einem XML-Parser analysiert werden. JSON kann von einer Standard-JavaScript-Funktion analysiert werden.

## 4. Open Data

Die freie Verfügbarkeit und Verfügbarkeit von Daten im Web werden oft als Open Data bezeichnet. Erstens ist Open Data ein sehr allgemeiner Begriff, der anders benannt werden kann, wenn er in einem bestimmten Kontext verstanden werden soll. Die Idee von Open Data ist, dass die freie Weiterverwendung für mehr Transparenz sorgt und mehr Kollaboration schafft. Open Data ist Teil der sogenannten Open-Science-Bewegung. Dazu gehören zum Beispiel Open Source, Open Education, Open Government und das bereits erwähnte Open Access.

Offene Daten sind Daten, die ohne Einschränkungen kostenlos genutzt, weiterverbreitet und weiterverwendet werden können. Dazu gehören Unterrichtsmaterialien, geografische Daten, statistische Daten, Verkehrsinformationen, wissenschaftliche Veröffentlichungen, medizinische Forschungsergebnisse oder Radio- und Fernsehsendungen. Daten können über verschiedene Lizenzen als offene Daten gekennzeichnet werden.



[https://cdn-images-1.medium.com/max/1600/0\\*fMZTkLQ49qEoklhW.png](https://cdn-images-1.medium.com/max/1600/0*fMZTkLQ49qEoklhW.png)

## 5. Index



<https://thumbs.dreamstime.com/b/word-index-made-wood-building-blocks-195210152.jpg>

Ein Datenbankindex ist eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank. Sie beschleunigt die Suche und das Sortieren nach bestimmten Daten. Ein Index besteht aus Ansammlungen von Zeigern, also Verweisen, die eine Ordnungsrelation auf eine Spalte der Tabelle definieren. So wird bei einer Abfrage eine indizierte Spalte als Suchkriterium herangezogen und somit wird von der DBMS die gewünschten Datensätze anhand dieser Zeiger gesucht. Ohne diesen Index müsste die Spalte sequenziell durchsucht werden.

Syntax: `CREATE INDEX Indexname ON Tabellename (Spaltenname (n))`



## 5.1 Bitmap Index

### Bitmap Index Example

*(A=19)  
one bitvector  
len: N  
098-55-1234  
10000...0  
01000...0  
Compress*

RowId	FacSSN	...	FacRank	Asst	Prof	RLG
1	098-55-1234		Asst	1	0	0
2	123-45-6789		Asst	1	0	0
3	456-89-1243		Asst	0	1	0
4	111-09-0245		Prof	0	0	1
5	931-99-2034		Asst	1	0	0
6	998-00-1245		Prof	0	0	1
7	287-44-3341		Asst	0	1	0
8	230-21-9432		Asst	1	0	0
9	321-44-5588		Prof	0	0	1
10	443-22-3356		Asst	0	1	0
11	559-87-3211		Prof	0	0	1
12	220-44-5688		Asst	1	0	0

*Salars  
30 40 50.*

*RLG  
minimize  
disk space*

FacRank	Bitmap
Asst	110010010001
Asst	001000100100
Prof	000101001010

Dieser Bitmap Index basiert auf der Speicherung der Spaltenwerte und dies in Form von Bitketten. Dieser Typ der Indexierung wird nur bei geringer Selektivität und niedriger Aktualisierungswartung der indizierenden Spalte verwendet.

#### Vorteile

- Effizienz in Bezug auf das Löschen und Aktualisieren von Einfügungen.
- Schnelleres Abrufen von Datensätzen

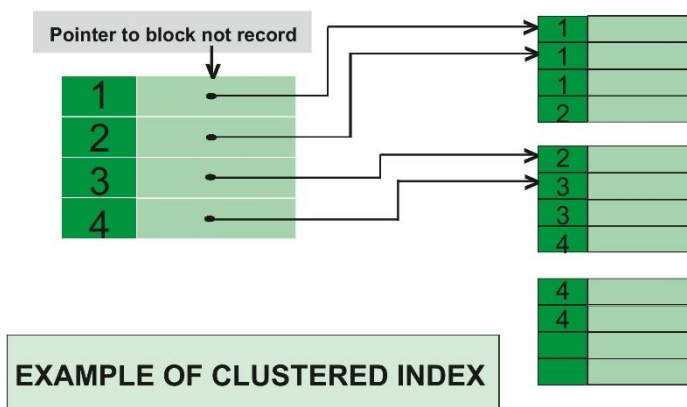
#### Nachteile

- Nur für große Tische geeignet
- Bitmap-Indizierung ist zeitaufwändig

PPT - Bitmap Indexing PowerPoint Presentation, free download - ID:3872879 (slideserve.com)

Syntax: `CREATE BITMAP INDEX Indexname ON Tabellennamen (Spaltenname)`

## 5.2 Clustered Index



Einfach gesagt ist ein Clustered Index dasselbe wie ein Wörterbuch, bei dem die Daten in alphabetischer Reihenfolge angeordnet sind. Mit einem clustered Index versucht das DBMS nicht nur die Vorlegung einer Liste der Zeiger auf die Datensätze in sortierter Form, sondern versucht zusätzlich auch neu eingefügte Datensätze, die nah beim Index liegen, auch physikalisch im Speicher nah beieinander abzulegen.

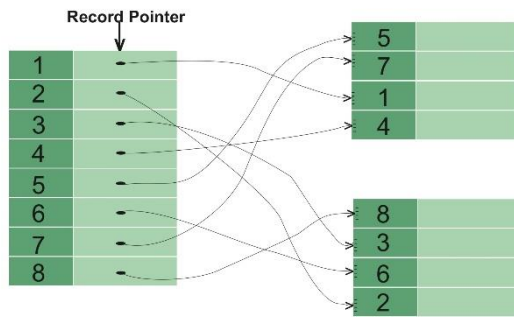
[https://media.geeksforgeeks.org/wp-content/uploads/20200410115906/Clustered\\_Index.jpg](https://media.geeksforgeeks.org/wp-content/uploads/20200410115906/Clustered_Index.jpg)

Dies beschleunigt die Suche noch mehr. Dieser Index kann nur erstellt werden, wenn:

- Die Daten oder Dateien, die man in den sekundären Speicher verschieben will, sollten in sequenzieller oder sortierter Reihenfolge vorliegen.
- Es sollte einen Schlüsselwert geben, was bedeutet, dass er keine wiederholten Werte haben kann.

Syntax: `CREATE CLUSTERED INDEX index_name ON schema_name.table_name (column_list);`

## 5.3 Non - Clustered Index



EXAMPLE OF NON-CLUSTERED INDEX

Einfach gesagt ist ein Non-Clustered Index ähnlich wie den Index ein Buch. Es besteht aus einem Titel und die Seitenzahl. Somit kann man direkt zu der gewünschten Seite gehen, ohne durch jede einzelne Seite zu gehen müssen.

Die Daten werden an einer Stelle gespeichert und das Index in einer anderen. Dadurch kann man auch mehrere non-clustered Index in einer Tabelle haben. Bei diesem Index enthält der Index den Pointer auf Daten.

[https://media.geeksforgeeks.org/wp-content/uploads/20200410120011/Non-clustered\\_Index.jpg](https://media.geeksforgeeks.org/wp-content/uploads/20200410120011/Non-clustered_Index.jpg)

### Clustered VS non-Clustered by GeeksForGeeks:

CLUSTERED INDEX	NON-CLUSTERED INDEX
Clustered index is faster.	Non-clustered index is slower.
Clustered index requires less memory for operations.	Non-Clustered index requires more memory for operations.
In clustered index, index is the main data.	In Non-Clustered index, index is the copy of data.
A table can have only one clustered index.	A table can have multiple non-clustered index.
Clustered index has inherent ability of storing data on the disk.	Non-Clustered index does not have inherent ability of storing data on the disk.
Clustered index store pointers to block not data.	Non-Clustered index store both value and a pointer to actual row that holds data.
In Clustered index leaf nodes are actual data itself.	In Non-Clustered index leaf nodes are not the actual data itself rather they only contains included columns.
In Clustered index, Clustered key defines order of data within table.	In Non-Clustered index, index key defines order of data within index.
A Clustered index is a type of index in which table records are physically reordered to match the index.	A Non-Clustered index is a special type of index in which logical order of index does not match physical stored order of the rows on disk.
The size of clustered index is large.	Size of non-clustered index is comparatively smaller.
Primary Keys of the table by default are clustered index.	Composite key when used with unique constraints of the table act as non-clustered index.

## 5.4 Funktionaler Index

Ein funktionaler Index ist ein Index in einer relationalen Datenbank, der auf eine Funktion oder Berechnung anstelle von einem Spaltenwert basiert. Ein Beispiel für einen funktionalen Index wäre ein Index, der auf die Länge eines Textfelds in einer Tabelle basiert. Funktionaler Index ermöglicht es, schneller auf bestimmte Daten zuzugreifen, die durch eine bestimmte Berechnung erhalten werden. Es gibt auch die Möglichkeit, einen funktionalen Index auf mehrere Spalten anzuwenden.

Im Gegensatz zu einem normalen Index werden nicht die reinen Feldwerte (z.B. Vorname) in den Index aufgenommen, sondern mittels Datenbankfunktionen, die den Wert transformieren (mit z.B. `to_upper(Vorname)`) in Grossbuchstaben.

## 5.5 Reverse Index

Ein Reverse Index, auch Umkehrindex genannt, ist ein Index, der verwendet wird, um Wörter oder Phrasen in einem Dokument oder einer Sammlung von Dokumenten zu suchen. Es funktioniert, indem es jedes Wort oder jede Phrase in einem Dokument identifiziert und die Dokumente, in denen es vorkommt, notiert. Dann werden diese Informationen in einem Index gespeichert, der es ermöglicht, schnell auf die Dokumente zuzugreifen, die ein bestimmtes Wort oder eine bestimmte Phrase enthalten. Reverse-Indizes werden häufig in Suchmaschinen, Textverarbeitungsprogrammen und anderen Anwendungen verwendet, die Textinhalte durchsuchen müssen.

## 5.6 Partitionierter Index

Ein partitionierter Index ist ein Index, der eine grosse Tabelle in kleinere Partitionen unterteilt. Jede Partition enthält einen Teil der Daten der Tabelle und hat ihren eigenen Index. Dies ermöglicht es, die Abfragezeiten zu beschleunigen, da nur die Partition durchsucht werden muss, die die gewünschten Daten enthält. Partitionierung kann auf verschiedene Arten durchgeführt werden, z.B. basierend auf einem bestimmten Wertebereich oder auf einem bestimmten Zeitrahmen. Partitionierte Indizes können auch dazu beitragen, die Verwaltung von grossen Tabellen zu vereinfachen, da jede Partition separat verwaltet werden kann.

# 6. MariaDB

[https://i0.wp.com/res.cloudinary.com/dcyta1qo0/image/upload/v1488060377/mariadb-logo\\_zpiqlb.jpg](https://i0.wp.com/res.cloudinary.com/dcyta1qo0/image/upload/v1488060377/mariadb-logo_zpiqlb.jpg)



MariaDB ist eine Open-Source-Relational-Datenbank-Management-System (RDBMS), die auf MySQL basiert. Es wurde von den ursprünglichen Entwicklern von MySQL entwickelt und hat viele der gleichen Funktionen wie MySQL, einschliesslich der Unterstützung für die Structured Query Language (SQL) und die Fähigkeit, mit vielen Programmiersprachen zu arbeiten. Es hat jedoch auch einige zusätzliche Funktionen, wie z.B. die Unterstützung für die NoSQL-Sprache X protocol,

und die Möglichkeit, Daten in Column-Store- und Row-Store-Tabellen zu speichern. MariaDB wird häufig als Ersatz für MySQL verwendet und hat eine aktive Community und viele verfügbare Plugins und Erweiterungen.

## 7. Cassandra



Als verteiltes System verwendet Cassandra keinen Master. Alle Cluster haben die gleichen Rechte, um jede Datenbankabfrage zu verarbeiten, was zu einer erheblich verbesserten Leistung führt. Daten werden über Knoten verteilt. Das System ist auch leicht skalierbar, indem einfach weitere Knoten hinzugefügt werden. Nach der Installation müssen Sie lediglich die Konfigurationsdateien auf die neuen Knoten verteilen. Cassandra stellt dafür die richtigen Tools zur Verfügung. Wurde von Facebook erfunden.

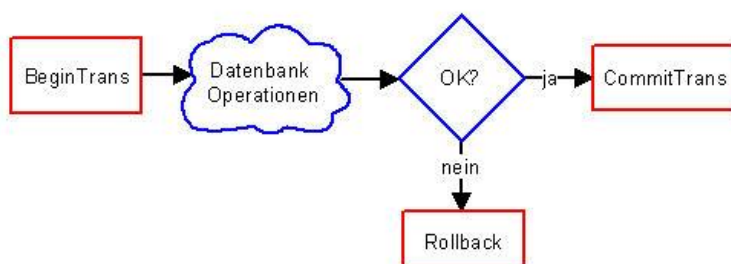
[https://th.bing.com/th/id/R.954baac6708253a60ea3849gae8d0d23?rik=z9DYIKo7UPuc1g&riu=http%3a%2f%2fec30.org%2fwp-content%2fuploads%2f2017%2f01%2flarge\\_CassandraLogo-1443223369.png&ehk=EYbyTzyWGBMUNArAgwIPdhgxgglQZAsxiblwc%2bo2p1o%3d&risl=&pid=ImgRaw&r=0](https://th.bing.com/th/id/R.954baac6708253a60ea3849gae8d0d23?rik=z9DYIKo7UPuc1g&riu=http%3a%2f%2fec30.org%2fwp-content%2fuploads%2f2017%2f01%2flarge_CassandraLogo-1443223369.png&ehk=EYbyTzyWGBMUNArAgwIPdhgxgglQZAsxiblwc%2bo2p1o%3d&risl=&pid=ImgRaw&r=0)

### 7.1 Vorteil und Nachteil

Einer der wichtigsten Vorteile von Cassandra ist die gute Skalierbarkeit und sehr hohe Zuverlässigkeit – Grundvoraussetzungen für Big-Data-Anwendungen. Cassandra bezieht sich auf horizontale Skalierbarkeit, was bedeutet, dass die Kapazität und Leistung des Systems durch das Hinzufügen von Knoten erhöht werden kann. Im Gegensatz zur vertikalen Skalierung müssen einzelne Datenbankserver mit leistungsfähigeren Prozessoren und größeren Festplatten aufgerüstet werden, wenn Leistung oder Kapazität den Anforderungen nicht mehr genügen. Da Sie Standard-Server-Hardware verwenden können, ist die horizontale Skalierung in den meisten Fällen die günstigere Lösung.

## 8. Transaktionen

Als Transaktion wird in der Informatik allgemein eine Abfolge von Teilaktionen bezeichnet, die entweder fehlerfrei und vollständig oder gar nicht durchgeführt wird. Auf Datenbanken bezogen werden die Teilaktionen auf Tabellen ausgeführt. Tritt bei einer Teilaktion einer Transaktion ein Fehler auf, muss die Transaktion abgebrochen und alle bisherigen Änderungen rückgängig gemacht werden (Rollback). Erst nachdem alle Teilaktionen fehlerfrei durchlaufen wurden, wird die Transaktion dauerhaft abgeschlossen (Commit). Transaktionen verhindern, dass in Datenbanken ein inkonsistenter Zustand entsteht, selbst wenn bei der Durchführung von Teilaktionen Fehler auftreten.



<https://i2.wp.com/www.accessblog.de/wp-content/uploads/2008/04/snap2.jpg>

## 8.1 ACID

In der Informatik werden vier Eigenschaften definiert, die Transaktionen haben müssen. Aus den Anfangsbuchstaben der vier Eigenschaften entstand das Akronym ACID:

<https://static1.tothenew.com/blog/wp-content/uploads/2016/06/img3.png>

### **ATOMICITY (ATOMARITÄT)**

Eine Transaktion wird entweder vollständig oder gar nicht ausgeführt. Wird eine Transaktion abgebrochen, wird das System (eine Datenbank) in den vorherigen Zustand zurückgesetzt.

### **CONSISTENCY (KONSISTENZ)**

Ist ein System (eine Datenbank) in einem konsistenten Zustand, muss das System nach der Transaktion auch in einem konsistenten Zustand sein.

### **ISOLATION (ISOLATION)**

Bei der gleichzeitigen Durchführung von mehreren Transaktionen müssen die einzelnen Transaktionen unabhängig voneinander (isoliert) abgearbeitet werden und dürfen sich nicht gegenseitig beeinflussen.

### **DURABILITY (DAUERHAFTIGKEIT)**

Das Ergebnis einer Transaktion muss dauerhaft erhalten bleiben und darf nicht durch andere Transaktionen verloren gehen. Damit sind auch verschachtelte Transaktionen nicht erlaubt, da ein Rollback der äußeren Transaktion das Ergebnis einer bereits abgeschlossenen inneren Transaktion ungültig machen würde.



## 8.2 CAP

**CAP** steht für **Konsistenz**, **Verfügbarkeit** und **Partitionstoleranz**.

**(Consistency)** Konsistenz bedeutet, dass Sie, wenn Sie Daten in das verteilte System schreiben, in der Lage sein sollten, dieselben Daten zu jedem Zeitpunkt von jedem Knoten des Systems zu lesen oder einfach einen Fehler zurückzugeben, wenn sich die Daten in einem inkonsistenten Zustand befinden. Niemals inkonsistente Daten zurückgeben.

**(Availability)** Verfügbarkeit bedeutet, dass das System Lese-/Schreibvorgänge auf allen nicht ausgefallenen Knoten des Clusters immer erfolgreich und fehlerfrei durchführen sollte. Die Verfügbarkeit hängt hauptsächlich mit der Netzwerkpartition zusammen, d. h. ob ein Knoten bei einem Lese-/Schreibvorgang eine erfolgreiche Antwort oder einen Fehler zurückgibt, wenn eine Netzwerkpartition vorhanden ist.

**(Partition Tolerance)** Partitionstoleranz bedeutet, dass das System auch dann noch funktioniert, wenn es eine Trennung zwischen den Knoten gibt oder die Teile des Clusters in einem verteilten System nicht miteinander kommunizieren können.

## 8.3 BASE

BASE beschreibt die Datenbankverarbeitung, die einer NoSQL-Datenbank, wie z.B. einem Data Lake, zugrunde liegt. Laut DAMA DMBoK wurde die BASE-Philosophie durch eine steigende Anzahl von Datenvolumen und -variabilität vorangetrieben. Seine Popularität stieg im Jahr 2008. BASE bietet weniger Sicherheit als ACID, ist aber sehr gut skalierbar und reagiert gut auf schnelle Datenänderungen. Die BASE-Konstruktion hat drei Eigenschaften:

**Basically Available:** (Grundlegend verfügbar) Garantiert die Verfügbarkeit der Daten. Es gibt eine Antwort auf jede Anfrage (kann auch ein Fehler sein).

**Soft state:** (Weicher Zustand) Der Zustand des Systems kann sich im Laufe der Zeit ändern.

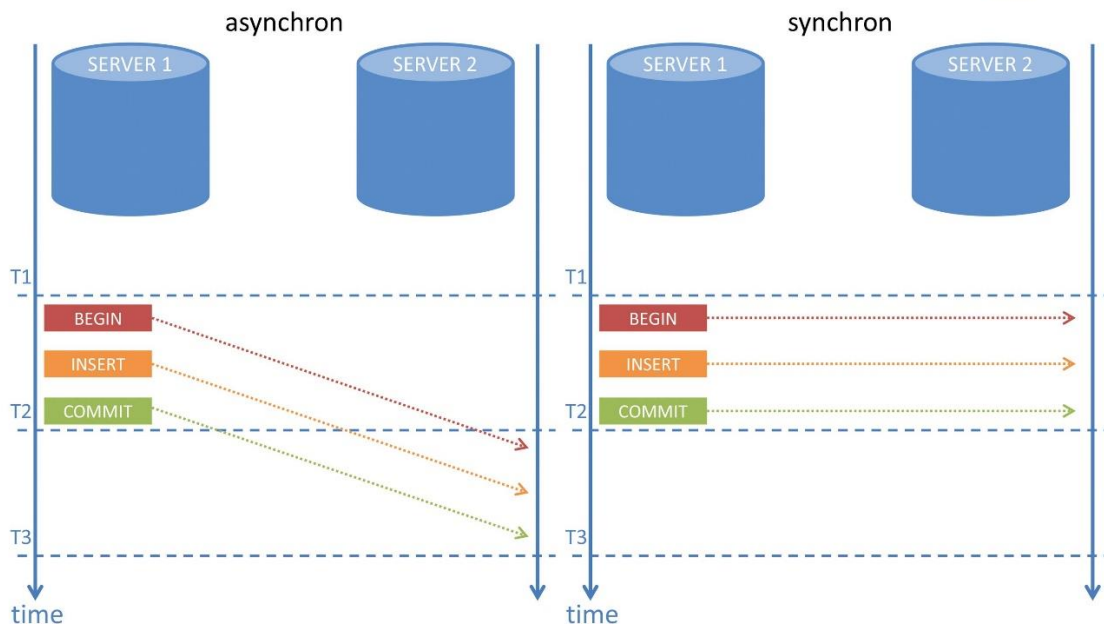
**Eventual consistency:** (Eventuelle Konsistenz) Das System wird schließlich konsistent, sobald es keine Eingaben mehr erhält.



# 9. Replication

## 9.1 Replikation (Datenverarbeitung)

Asynchrone / synchrone Replikation



<https://www.linux-magazin.de/wp-content/uploads/2017/04/artikel-3.jpg>

**Definition:** Unter Replikation oder auch Replizieren (lateinisch: Replicare «wiederholen») versteht man die mehrfache Speicherung der Kopie derselben Daten an verschiedenen Orten und die Synchronisation dieser Daten.

Replikationen dienen zur Verkürzung der Antwortzeiten und zur Datensicherung. In dem häufig vorkommenden Master/Slave-Replikation (Hierarchische Verwaltung) unterscheidet man Originaldaten (Primärdaten) und abhängige Kopien.

Je nach Art der Replikation gibt es eine gewisse Zeitspanne (timelines oder latency) zwischen der Erstellung oder Bearbeitung der Primärdaten und ihrer Replizierung.

### Asynchrone Replikation

Wenn zwischen der Bearbeitung der Primärdaten und der Replizierung eine Latenz (Verzögerung) liegt, spricht man von asynchroner Replikation. In diesem Fall sind die Daten nur zu dem Zeitpunkt der Replikation synchron(identisch).

Eine Variante der asynchronen Replikation ist beispielsweise die «File Transfer Replication», also die Übergabe von Dateien via FTP(File) oder SSH(Secure Shell).

Eine Strategie für asynchroner Replikation ist beispielsweise die Merge-Replikation. Es ist eine Methode zur Synchronisation von Daten zwischen zwei Microsoft-SQL-Server-Datenbanken, bei der eine Datenbank offline modifiziert und erst später mit den Zentralbanken oder anderen Datenbanken abgeglichen werden kann.

### Synchrone Replikation

Wenn eine Änderung an einem Datenobjekt nur dann tatsächlich abgeschlossen werden kann, wenn sie auch auf den Replikaten durchgeführt wurde. Um die Atomarität (Unteilbarkeit) gewährleisten zu können. Kann man ein Commit-Protokoll anwenden.

Eine Strategie für synchroner Replikation ist beispielsweise das ROWA-Verfahren (Read-One-Write-All). Änderungen an einem Datenobjekt wird immer synchron auf allen Replikaten durchgeführt, damit alle Replikate immer auf dem gleichen Stand sind und keines veraltet ist.

### **Zweck und Einsatz**

- Verfügbarkeit
- Performance - Lastverteilung
- Backup

### **Anwendungsmöglichkeiten**

- Data-Warehousing
- Abgleich von Datenbanken in der Groupware

### **Vorteile der Replikation**

- Erhöhte Verfügbarkeit der Daten
- Beschleunigung von Lesezugriffen, also bessere Antwortzeiten
- Bessere Query-optimierung und Lastverteilung

### **Nachteile der Replikation**

- Hoher updateaufwand
- Bessere Query-optimierung und Lastverteilung
- Mögliche Redundanz

## **9.2 Cassandra Replikation**

Cassandra speichert Replikate auf mehreren Knoten, um Zuverlässigkeit und Fehlertoleranz zu gewährleisten. Eine Replikationsstrategie bestimmt die Knoten, auf denen Replikate platziert werden. Die Gesamtzahl der Replikate im gesamten Cluster wird als Replikationsfaktor bezeichnet. Ein Replikationsfaktor von 1 bedeutet, dass es von jeder Zeile nur eine Kopie im Cluster gibt. Wenn der Knoten, der die Zeile enthält, ausfällt, kann die Zeile nicht abgerufen werden. Ein Replikationsfaktor von 2 bedeutet, dass es von jeder Zeile zwei Kopien gibt, wobei sich jede Kopie auf einem anderen Knoten befindet. Alle Replikate sind gleich wichtig; es gibt kein primäres oder Master-Replikat. In der Regel sollte der Replikationsfaktor die Anzahl der Knoten im Cluster nicht überschreiten. Sie können jedoch den Replikationsfaktor erhöhen und später die gewünschte Anzahl von Knoten hinzufügen.

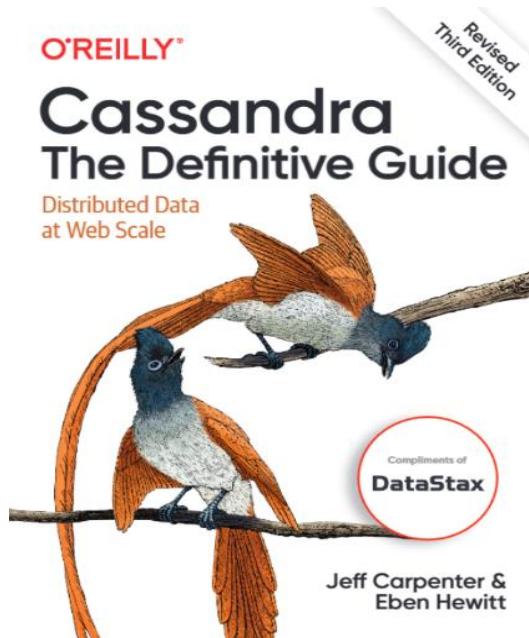
Es stehen zwei Replikationsstrategien zur Verfügung:

- SimpleStrategy: Nur für ein einziges Rechenzentrum und ein Rack verwenden. Wenn Sie mehr als ein Rechenzentrum planen, verwenden Sie die NetworkTopologyStrategy.
- NetworkTopologyStrategy: Sehr empfehlenswert für die meisten Bereitstellungen, da es viel einfacher ist, auf mehrere Rechenzentren zu erweitern, wenn dies für zukünftige Erweiterungen erforderlich ist.



# 10. Einleitung

## 10.1 Vorwort



Im Modul 165 erhielten wir einen eigenen Projektauftrag, an dem wir in einer dreier Gruppe arbeiten sollen. Jeder von uns hatte schon eigene Erfahrung mit Datenbanken, jedoch waren diese Relationaldatenbanken. Dieses Mal mussten wir ein Wir waren uns anfangs uneinig, ob wir mit Cassandra, Neo4j oder MongoDB arbeiten wollen. Der Grund, warum wir und für Cassandra entschieden haben ist, dass uns ein Buch über Cassandra von unserem Lehrer schon zur Verfügung gestellt worden ist.

Da wir nun wussten, was für ein Datenbank Engine wir verwenden werden, haben wir uns ein Thema für unser Projekt ausgedacht, und zwar eine Filmbibliothek. Für ein einfacheres Verständnis haben wir die ganze Applikation in Englisch programmiert.

## 10.2 Warum dieses Projekt

Wie gesagt, haben wir keinen spezifischen Auftrag erhalten. Unser Hauptziel war, dass unser gewähltes Konzept eine NoSQL Datenbank mit einer Liste von Filmen sein soll. Das Konzept mit den Filmen gefällt uns allen, da wir gerne in unserem Freizeitfilme von Bollywood zu Hollywood über Netflix oder auch nur YouTube anschauen. In unsere früheren Module haben wir sogar einen eigenen Trailer zusammengeschnitten. Der Vorteil, wenn man an einem Projekt mit einem selbst gewählten Thema arbeitet, ist, dass man auch Lust und Laune hat, die Freizeit dafür zu „opfern“. Unsere Projektidee war, ein Frontend, ein Backend und eine NoSQL Datenbank aufsetzen. Ausserdem wollten wir noch eine Dokumentation schreiben zu all den Themen, die wir im Unterricht angeschaut haben. Schlussendlich berichten wir darüber, wie wir mit unserem Projekt vorgegangen sind.

## 10.3 Wie sind wir vorgegangen

es einzelnen Inputs, die wir anfangs Lektion bekamen, waren sehr beachtenswert. Um diese später in unserer Dokumentation genauer zu anschauen, haben wir die Themen kurz auf einem txt Datei aufgeschrieben. Und haben angefangen, die Themen zu behandeln. Gleichzeitig haben wir angefangen, einen Docker für das Backend zu erstellen und Cassandra Datenbank aufzustellen. Zudem haben wir im Frontend die einzelnen React TypeScript Seiten vorbereitet, die die Daten vom Backendholen und auflisten konnten. Natürlich haben wir auch auf das Layout geachtet. Dann mussten wir das Backend und Frontend miteinander verbinden.

## 10.4 Verwendete Methoden

Natürlich haben wir wie jedes Mal mit **IPERKA** gearbeitet. Dies ist eine Methode, die wir alle drei bei der NoserYoung schon seit dem Beginn angewendet haben. Viele, die mit Projekten arbeiten, verwenden diese Methode.

### Informieren

Am Anfang von der ersten Lektion bekamen wir einen kleinen Auftrag. Unsere selbst gestellte Aufgabe war, zu einem Thema unser Vorgehen notieren und es auf Papier oder digital festhalten. Gegeben haben wir uns eine Lektion für diese Aufgabe. Die haben wir dann in der Gruppe angeschaut. Dann mussten wir offene Fragen klären. Alle der Klassen hatten unterschiedliche Ideen.

Offene Fragen:

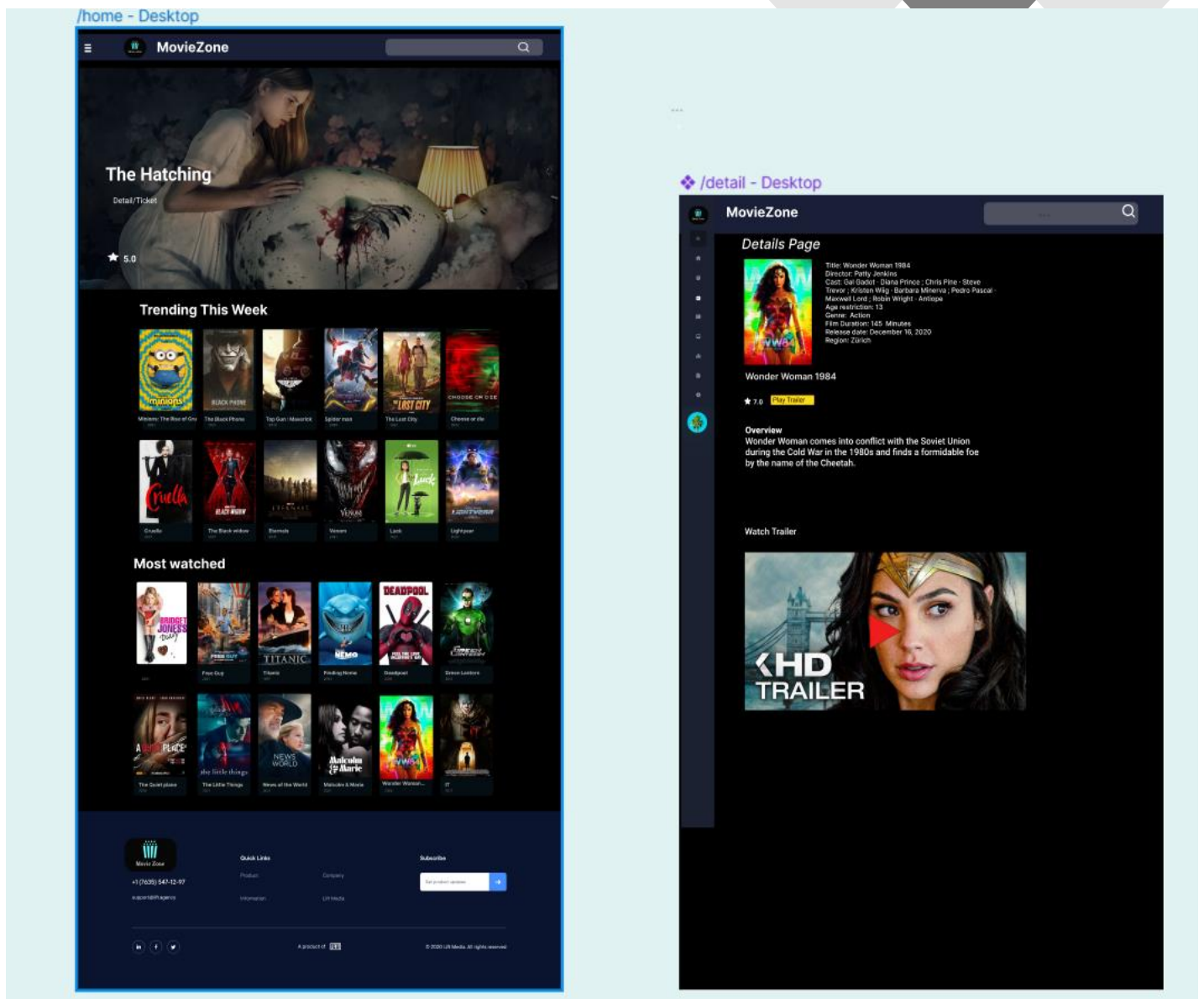
Nr.	Frage	Priorität
1	Welches Datenbankverwaltungssystem nehmen wir?	1
2	Welche Tools verwenden wir?	1
3	Welches Framework verwenden wir für Frontend?	1
4	Welches Framework verwenden wir für Backend?	1
5	Welches Thema hat unser Projekt?	2
6	Was für Eigenschaften und Daten haben wir?	2
7	Wer übernimmt welche Aufgabe?	3
8	An welchen Tagen wird ausserhalb der Schule gearbeitet?	4
9	Welchen Teil vom Code müssten wir evtl. testen?	5

Zu beachten war auch, wie detailliert man die einzelnen Schritte beschreibt und auch wie genau wir die einzelnen Fragen anschauen müssten. Einige dieser Fragen hatten höhere Prioritäten als die anderen. Da wir unsere Fragen priorisiert hatten, fiel uns leichter, organisierter und schneller vorwärtszukommen. Wir mussten uns über die einzelnen DBMS-Gedanken machen. Denn diese kommen in andere Projektaufträge sicher mal vor. In unserem Fall hatten wir keinen Auftraggeber. Nötig zu wissen war, welchen Auftrag man genau hat und ob man alle nötigen Unterlagen hat, um dies zu erledigen. Unsere Lernaufgabe war einen Projektauftrag herzustellen und den zu präsentieren. Der Lehrer muss den von uns gestellten Auftrag auch bewilligen.

Zur beginn haben wir nach dem Prozess gesucht, das am besten für das Vorgehen geeignet war. Wir haben danach festgestellt, dass der Entity Film bestimmte Eigenschaften, also Felder hat. Wir haben uns notiert, welche Felder mit welchen Datentypen wir verwenden werden, dass den Film beschreibt. Und mit weiteren Recherchen fanden wir heraus, welche weiteren Details am besten für die Beschreibung den einzelnen Filmen geeignet sind. Jedoch haben wir kein UML-Diagramm gezeichnet, da wir nur wenige Entitäts haben. Denn viel Zeit wollten wir nicht bei dem verschwenden. Danach haben wir wichtige Punkte notiert, welches den UI/UX der User verbessern könnte.

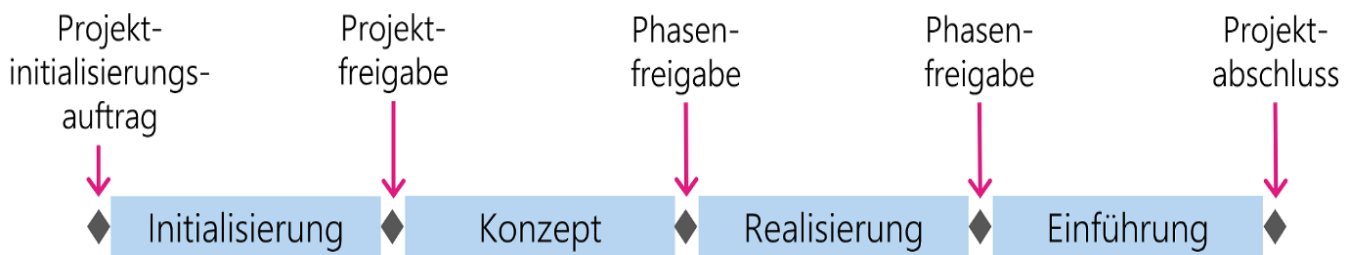
Figma Design:

Von unserem letzten Modul, den wir genau so für unseres übernommen haben.



## Planen

Bei diesem Schritt hatten wir uns gefragt, wieso das Planen eigentlich der wichtigste Schritt ist. Dazu lernten wir, dass die Zielsetzung und die Zielverfolgung wichtig und hilfreich sind. Wenn man sich bestimmte Ziele setzt, kann die Zusammenarbeit im Team sehr erleichtert werden. Wir haben eine Lerntechnik kennengelernt, was mit der Zielsetzung hilft. Die erste Hilfe ist der Meilenstein. Was sind aber Meilensteine überhaupt? Meilensteine sind wichtige Punkte im Projektverlauf, an denen ein bestimmtes Ziel erreicht oder ein definiertes Ergebnis erarbeitet worden sein soll. Oft werden sie auch als Prüfpunkte bezeichnet. Aber die Meilensteine zeigen nicht nur, was noch zu erledigen ist, sondern auch, was man schon erreicht hat. Uns hatten die Meilensteine erstaunlicherweise sehr motiviert. Die Freude, die wir hatten, wenn wir ein Zwischenergebnis zum geplanten Termin erreichen konnten, war unbeschreiblich. Und wir bemerkten, dass wir generell härter arbeiteten, wenn eine Deadline feststeht. Meilensteine sind essenzielle Punkte im Projekt, an denen vordefinierte Lösungen erreicht werden sollen. Diese wichtigen Prüfpunkte liegen oftmals am Ende von Projektphasen und stellen auf diese Weise definitiv, dass die eine Phase abgeschlossen und die nächste begonnen werden kann. Für die praktische Arbeit ist eine saubere Formulierung wichtig.



Danach mussten wir feststellen welche Aufgaben zu lösen waren, in welcher Reihenfolge wir sie lösen mussten, wie lange es dauern könnte, bis die Aufgaben gelöst waren, welche Aufgaben als erstes zu lösen waren, wer dafür verantwortlich war und welche Mittel wir dafür benötigten. Nach dem Beantworten dieser Fragen, wurde uns einiges viel klarer.

## Entscheiden

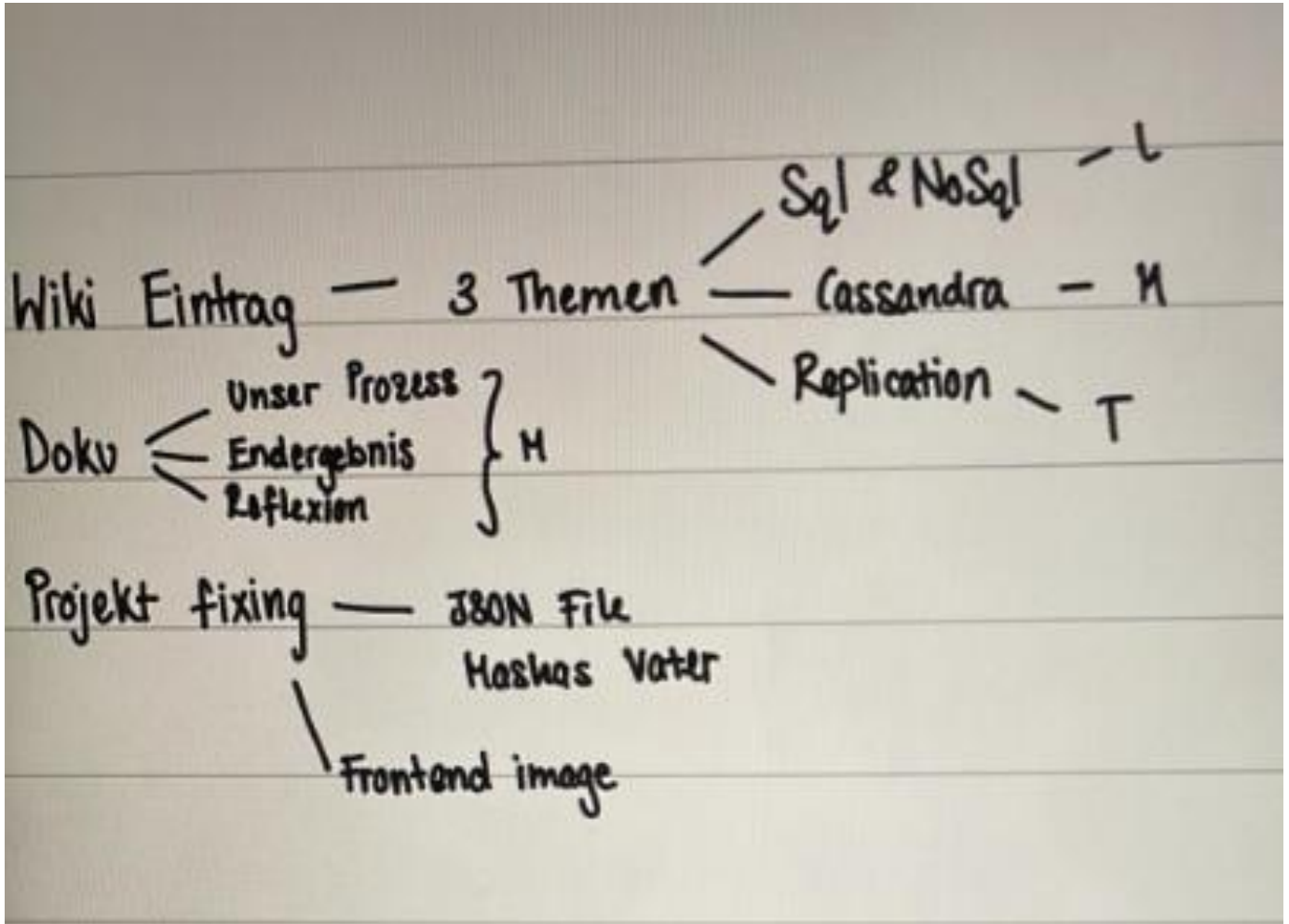
Oft wenn man bei einem Projekt zwei Möglichkeiten hat etwas auszuführen, entscheidet man sich für die bestmögliche: Doch wie entscheidet man, welches die beste Möglichkeit ist? Jetzt mussten wir uns entscheiden! Und zwar für einen Lösungsweg und definieren, welche Kriterien uns zu dieser Entscheidung geführt haben. Was gemacht werden musste war, die Varianten erst mal zu vergleichen. Wir lernten zu diesem Schritt die SWOT-Analyse kennen. Eines der bekanntesten Strategietools des Projektmanagements ist die SWOT-Analyse. Der Name SWOT setzt sich aus den Anfangsbuchstaben von Strengths (Stärken), Weaknesses (Schwächen), Opportunities (Chancen) und Threats (Risiken) zusammen. Eine solche Analyse wird zur Situationsanalyse und zur strategischen Planung genutzt. Eine weitere Strategie, die man zu Hilfe nehmen konnte, war der Entscheidungsmatrix. Eine Entscheidungsmatrix ist ein Hilfsmittel zur Auswahl der besten Alternative aus verschiedenen Optionen unter Berücksichtigung definierter Kriterien. Es ist eine Tabelle, die verschiedene Entscheidungsoptionen bzw. Alternativen in Spalten und Kriterien bzw. Eigenschaften in Zeilen anordnet. Durch die unterschiedliche Gewichtung einzelner Kriterien ergibt sich auf einmal ein klares Bild. Hier ist der Entscheidungsmatrix, den wir für unser Projekt gemacht haben.

Kriterien	Gewichtung	Cassandra (Variante 1)	MariaDB (Variante 2)
Hilfsmittel zur Verfügung	5	X	
Noch nie damit gearbeitet	1	X	X
Ist NoSQL	5	x	X
Java	5	X	X
Schema free	1	x	

Wir wollen unsere Auswahl auf zwei beschränken, also haben wir Cassandra und MariaDB ausgewählt, denn sie sind bekannt. Die Kriterien noch nie damit gearbeitet und Schemafree schien uns nicht so wichtig zu sein wie die anderen. Denn hauptsächlich geht es um ein NoSQL Datenbank.

## Realisieren

Dies ist die Phase, in der wir analysieren müssen, wo wir uns jetzt befinden. Und um diesen Prozess zu erleichtern, brachte uns unsere Auftraggeber bei, wie man eine Kanban Methode herstellen könnte. Die Kanban-Methode wird verwendet, um den Arbeitsablauf zu verbessern und gleichzeitig die Produktivität und Qualität des Endprodukts zu steigern. Kanban macht Arbeitsprozesse deutlich flexibler.

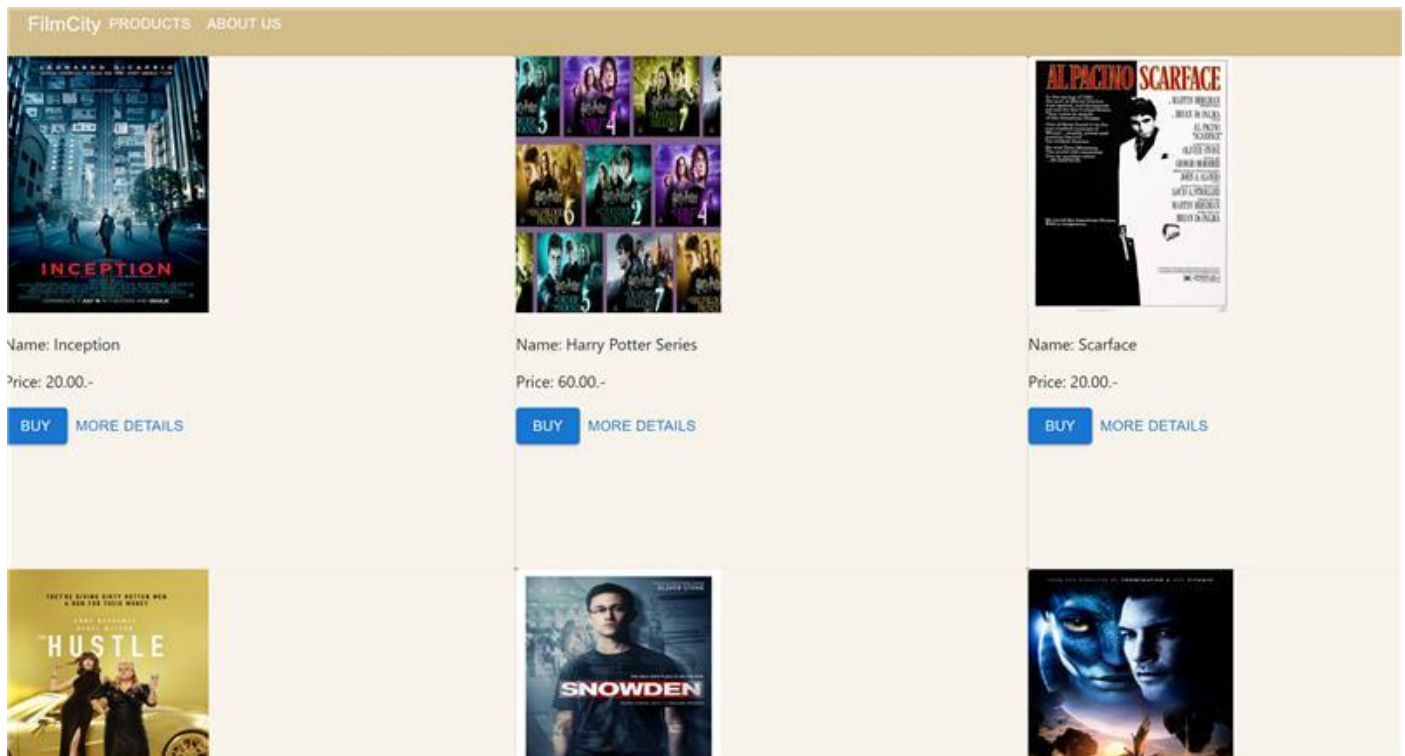


Wir haben die Aufgaben unter uns aufgeteilt. Diese haben wir auf dem iPad festgelegt.



## Unser Prozess

Wir haben und entschieden, eine Applikation aufzustellen, welche ein Frontend und ein Backend besitzt. Da wir schon bei der Noser Erfahrungen gemacht haben, mit dem Aufsetzen von Fullstack Applikationen, ist uns das relativ einfach gefallen und wir konnten uns auf die Doku, unseren Wiki-Eintrag und das Verbinden mit der Datenbank fokussieren. Das Backend besitzt die Endpoint's CREATE, DELETE, UPDATE und GET. Wir sind leider nicht so weit gekommen, dass wir diese, abgesehen von Test Requests, wirklich ausführen können. Ansonsten gibt es bei uns eine Tabelle Movie, welche den Namen, den Preis und das Genre beinhaltet.



## Kontrollieren

Für die Kontrolle haben wir manuell getestet. Ausserdem haben wir auch einen Klassenkameraden gefragt und ihn dazu beauftragt unsere Webseite zu manipulieren. Unsere Webseite scheint resistent zu sein und seine Funktion zu erfüllen.

## Auswerten

Allgemein sind wir zufrieden mit unserer Dokumentation und Fullstack Applikation. Auch wenn unsere Applikation nicht das Beste ist, schätzen wir unsere gegenseitige Leistung sehr. Wir haben inzwischen vieles neues zum Thema NoSQL und Cassandra gelernt. Nicht nur über die Datenbanksysteme wissen wir jetzt Bescheid, sondern auch von verschiedene Prinzipien wie ACID und BASE. Es ist besser für das Informieren und Planen investieren, anstatt direkt zu realisieren. Somit kommt man nicht mehr so viele Hindernisse entgegen. Damit kann man Zeit und vielleicht sogar Geld sparen. Wir empfehlen Springboot Gradle für die Leute, die als Anfänger ein Backend erstellen wollen. Wir würden so ein Projekt gerne wieder machen. Es hat Spass gemacht zusammen in einem Team zu arbeiten und es fühlt sich gut an, wenn man etwas erreicht hat.

# 10.5 Hauptteil

## 10.6 Was sind unsere Befürchtungen

Unsere Befürchtungen waren, dass wir das Projekt nicht vor der Deadline abgeben können und dass wir letzte Sekunde in Stress geraten. Wir haben aber auch unsere Freizeit dafür investiert, damit wir mit dem Projekt so weit wie möglich kommen. Da wir im gleichen Betrieb arbeiten, war es auch einfach für uns über die Fortschritte im Projekt zu reden.

## 10.7 Was wir erhoffen

Grund wollen wir ein funktionsfähiges Endprodukt. Was wir erhofft haben, war eine Fullstack Applikation mit der Verbindung von einer Cassandra Datenbank. Zudem wollen wir auch, dass unsere Datenbank bei der Präsentation keine Probleme verursacht

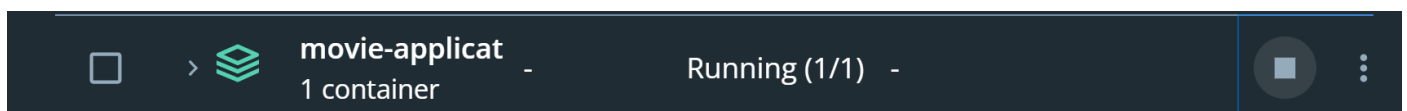
## 10.8 Verwendete Tools

Die Frameworks Springboot für das Backend und React im Frontend. Um die Datenbank zu hosten, haben wir Docker verwendet.

```
C:\Users\MashaMadani>docker ps
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS	PORTS
46ed4a071526	cassandra	"docker-entrypoint.s..."	mycassandra	2 weeks ago	Up 3 seconds	7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp

Eingerichtete Cassandra Datenbank



Laufender Docker Container



# 11. Schluss

## 11.1 Endergebnisse

Schlussendlich hatten wir Probleme, die unsere eingerichtete Datenbank mit dem Backend zu verbinden und haben uns entschieden, dass wir einen Wiki-Eintrag machen in Addition zu dieser Dokumentation und das halb fertige Projekt.

## 11.2 Quellenverzeichnis

Download von Cassandra: [Apache Cassandra](#) | [Apache Cassandra Documentation](#)

Cassandra: [Apache Cassandra](#) | [Apache Cassandra Documentation](#)

## 11.3 Reflektion

Wir finden, dass wir die Arbeit gut unter uns gut aufteilen, konnten. Wir haben unser Projekt auch so geplant, dass es recht umfangreich ist und wir immer etwas zu tun haben. Die Kommunikation in unsere Gruppe war auch sehr gut. Wir konnten uns gegenseitig immer fragen, wenn wir Fragen oder Schwierigkeiten hatten. Verschiedene Themen, die wir im Unterricht angeschaut haben, haben wir auch in der Gruppe besprochen.

## Wiki Eintrag

Wir haben uns überlegt, dass wir auch einen Wiki-Eintrag in der TBZ Wiki Seite machen, da uns dieser Vorschlag vom Lehrer gegeben wurde. Wir wollten in diesem Eintrag Themen beschreiben, die uns besonders interessiert haben in diesem Modul. Jeder von uns hat, haben je ein Thema ausgewählt und das im Eintrag festgehalten.

Hier ist der Link zu der Tbz Wiki Seite:

[https://www.tbzwiki.ch/index.php?title=Modul\\_165](https://www.tbzwiki.ch/index.php?title=Modul_165)