# Datenbank

Innerhalb der technischen Anforderungen des Auftraggebers wurde eine Datenbank zur programminternen Datenverwaltung vorausgesetzt. Dabei soll dieses System die anfallenden Datenmengen persistent speichern und die Konsistenz der Nutzdaten einer Institution gewährleisten. Zudem soll dadurch im Programm auf die darin gespeicherten Daten zugegriffen werden und diese in bedarfsgerechten Darstellungsformen präsentieren.

Um die richtige Datenbank auszuwählen wurde sollte zunächst ein genauerer Blick in den Aufbau und die Funktionen einer solchen Datenbank geworfen werden und diese dann folglich mit den im Projekt benötigten Anforderungen abgeglichen werden. Im Grunde besteht eine Datenbank aus genau zwei Teilen einmal einer Verwaltungssoftware, welche als Datenbankmanagementsystem (kurz. DBMS) bezeichnet wird. In dieser wird intern die strukturierte Speicherung der benötigten Daten organisiert und alle lesenden und schreibenden Zugriffe auf die Datenbank kontrolliert. Hierdurch können der Anwender bzw. der Entwickler die gesamte Datenbank mit Hilfe einer Datensprache vorerst einmal aufrufen und Daten anlegen bzw. diese bearbeiten oder löschen. Zudem legt ein solches Datenbankmanagementsystem das Datenbankmodell fest und entscheidet maßgeblich über die Funktionalität, sowie die Geschwindigkeit des benutzten Systems.[[1]](#footnote-1) Der Zweite Teil der Datenbank stellen die Menge der zu verwalteten Daten, der Datenbank (DB) an sich, dar. Dieser stellt sich als logisch zusammengehörender Datenbestand dar und wird von einem laufenden Datenbankmanagementsystem verwaltet.[[2]](#footnote-2) Eine solche Software zur Verwaltung der im Programm zu speichernden Daten wird in diesem Projekt benötigt. Die zuvor erwähnten Funktionen eines Datenbankmanagementsystems sind jedoch auch abgesehen von der Speicherung, der Überschreibung und der Löschung von Daten im Programm noch eine Menge andere. Dazu gehören die Verwaltung dieser Metadaten, eine Vorkehrung zu der Datensicherung, welche eine große Rolle für die in den Anforderungen verlangte Benutzerverwaltung darstellt. Zudem sind die weiteren Funktionen noch Vorkehrungen zur Datenintegrität und eine Ermöglichung des Mehrbenutzerbetriebs, welche in diesem Projekt von essentiellem Wert für die gerechte Verwaltung der Datenbank sind.

## Auswahl des Datenbankmodells

Um nun herauszufinden, welche Datenbank für die Software die richtige ist, beschäftigte sich der Backend-Teil des Teams während der Anforderungsanalyse mit den verschiedenen Arten der Datenbanken. Dabei wurden folgende Datenbanken Arten angesehen:

## Hierarchische Datenbank

In diesem Datenbankmodell, stehen die verschiedenen Objekte ausschließlich in einer Eltern-Kind-Beziehung. Das ist so, da dieses Modell versucht die reale Welt durch eine hierarchische Baumstruktur abzubilden. Jeder Satz hat hier genau einen Vorgänger, lediglich der Anfang dieses Datenbankmodells die sog. Wurzel hat keinen Vorgänger. Die Daten werden in einer Reihe von Datensätzen darin gespeichert, diese sind folglich mit weiteren verschiedenen Feldern verknüpft. Die Verknüpfungen in einem solchen Modell werden als Eltern-Kind-Beziehung oder auch Parent-Child-Relationship genannt, da es dem bekannten Familienstammbaum sehr ähnlich ist. Nachteilig ist jedoch, dass in einer solchen hierarchischen Datenbank lediglich mit einem solchen Baum umgegangen werden kann. Zudem können keine Verknüpfungen zwischen verschiedenen Bäumen oder über mehrere Ebenen des Baumes getätigt werden.[[3]](#footnote-3)

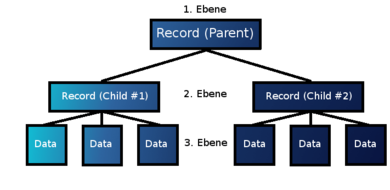


Abbildung : Hierarchische Datenbank Quelle: <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/>

## Vernetzte Datenbank

Hierfür werden die Daten Instanzen miteinander in einem Netz verbunden. Dieses Datenbankmodell erfordert keinerlei strenge Hierarchie, weshalb m zu n Beziehungen ohne Probleme abgebildet werden können und somit ein Datensatz mehrere Vorgänger haben kann. Um einen bestimmten Datensatz zu suchen gibt es mehrere Verschiedene Wege, weshalb sich die Suche nach einem bestimmten Datensatz etwas schwer darstellt.[[4]](#footnote-4)

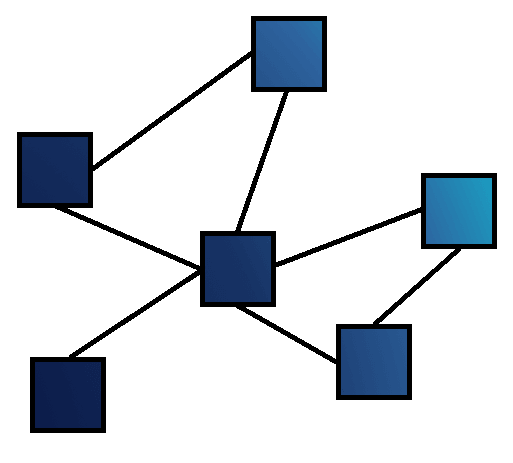


Abbildung : Vernetzte Datenbank Quelle: <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/>

## Objektorientierte Datenbank

## In diesem Datenbankmodell werden die Beziehungen zwischen verschiedenen Datenobjekten vom Datenbanksystem selbst und Daten werden hier speziell als Objekte verwaltet. Dabei kann eine Vererbung von Eigenschaften und Daten anderer Objekte stattfinden. Ein bestimmtes Objekt modelliert hier im Normalfall einen gewissen Gegenstand und enthält somit die jeweilig essentiellen Attribute. Somit wird bei einer Anlage eines Haus-Objektes die Farbe, die Größe und der Standort als Attribute dieses Objektes gespeichert. Dazu werden jedoch auch die dazu aufrufenden Methoden für die jeweiligen Attribute mitgespeichert, wie es mit den „getter“ und „settern“ in der Objektorientierung in Java bekannt ist gespeichert. Vorteile einer solchen Datenbank sind zum einem das dieses Modell Lücken schließt, welche bei der Programmierung moderner Datenbankanwendungen entstehen können, wenn sie in einer objektorientierten Programmiersprache entwickelt wurde. Und zudem wird das Problem der Objektidentität gelöst. Jedoch ist der Nachteil, dass für dieses Modell notwendige Schnittstellen und Tools, wie JDBC oder Spring, nicht für den Einsatz mit diesem Modell vorbereitet sind.[[5]](#footnote-5)

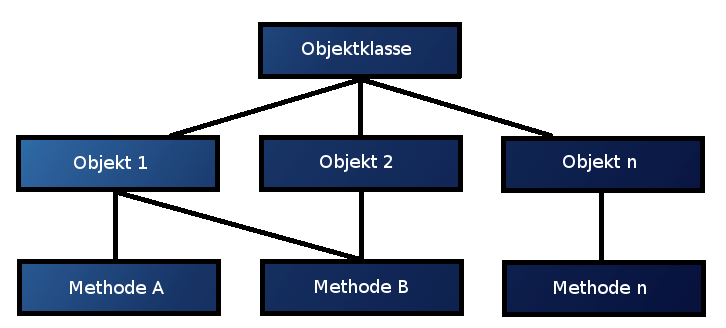


Abbildung : Objektorientierte Datenbank Quelle: <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/>

## Relationale Datenbank

Die in relationalen Datenbanken gespeicherten Daten werden in dafür erstellten Tabellen verwaltet. Es kann hier verschieden beliebige Beziehungen zwischen den Daten erfolgen, da diese durch Werte bestimmter Tabellenspalten festgelegt werden. Vorteilhaft hierbei sind die Anwendung von „Joins“ zum Verknüpfen von mehreren Datenbank Tabellen. Zudem sind mehrere zusätzliche Table-Funktionen, wie das addieren, subtrahieren, dividieren oder multiplizieren, bis hin zum aufsummieren möglich.[[6]](#footnote-6)

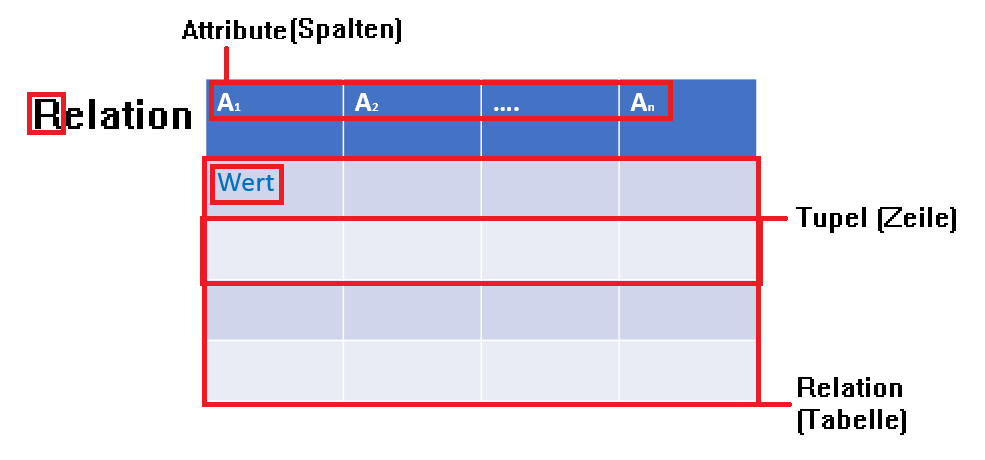


Abbildung : Relationale Datenbank Quelle: <https://www>.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/

Nach der Absprache im Team wurde sich dazu entschlossen eine Relationale Datenbank im Projekt mit einzubinden. Der Grund dafür ist, dass eine tabellarische Abspeicherung der Daten zu den Anforderungen der Anforderungsanalyse am besten passen. Zudem lässt sich mit einer solchen Datenbank ein relativ einfaches Konzept des Datenbankaufbaus erstellen. Indem mehrere Tabellen der Datenbank verbunden werden können, wird die Verwaltung und vor allem auch die Suche und folglich die Ausgabe der gewünschten Daten vereinfacht.

## Aufbau der Datenbank (Referenz auf Johannas Anforderungsanalyse und Modelle)

Wie in Kapitel (Johanna 2.4 Entity Relationship Model) beschrieben wurde zur Darstellung der Datenbank ein Entity-Relationship-Modell erstellt. Dies wurde gewählt, da dadurch ein genauer und einfacher Überblick über die Bestandteile der Datenbank aufgezeigt werden. Dazu wurde zwischen den Berichtigungen der User im Login und den Daten der gespeicherten ConfigItems unterschieden.



Abbildung : Berechtigungsdatenbank-ERM

Um einen Login in der relationalen Datenbank des Projektes zu realisieren wurden die Tabellen „User“ und die Tabelle „Berechtigung“ erstellt. Die Berechtigungen sind in der Berechtigungs-Tabelle vorgefertigt. Diese sind „Admin“ und „Benutzer“ (Berechtigungstype vielleicht verbessern). Durch diese kann im Programm unterschieden werden, wer welche Berechtigung hat und folglich welche Maßnahmen dieser auf der Oberfläche tätigen kann. Bei der Erstellung eines neuen Users schreibt das Programm in die User-Tabelle den jeweiligen Namen, das Passwort in Form eines Hash und die Referenz auf die Berechtigungs-ID der Person als Fremdschlüssel. Dabei kann jedoch jeder User nur eine einzige Berechtigung besitzen und dadurch auch nur die mit der Berechtigung verbundenen Tätigkeiten im Programm ausführen. Somit kann der Benutzer sich nach dem Starten der Anwendung im Login Fenster solchen Login-Daten anmelden und gelangt dadurch auf die eigentliche Anwendungsoberfläche.



Abbildung : CI-Datenbank-ERM

Nachdem der Anwender sich angemeldet hat, wird dieser voraussichtlich CI-Typen verwalten wollen, dazu müssen die Daten dieser gespeichert werden. Hierfür hat das Backend-Team einen besonderen Aufbau der Datenbank entwickelt. Vom Anwender können Configitemtypen erstellt werden, wozu der lediglich der Name dieses in der Tabelle „ConfigItem“ gespeichert wird. Zu jedem Cofigitemtyp werden verschiedene „Attributtypen“ definiert und später einmal beladen. Als Beispiel werden bei der Erstellung eines Configitemtyp mit dem Namen Handy zusätzlich die Attributtypen Marke, Speicher und Bildschirmgröße definiert. Dazu werden jeweils in der Tabelle „AttributTyp“ der Name des Attributtyps und der dazugehörige ConfigItemtypname als Fremdschlüssel gespeichert. Diese können durch den Fremdschlüssel genau einem ConfigItemtypen zugeordnet werden. Werden nun vom Anwender eine bestimmte Instanz eines Configitemtyps erstellt, werden die jeweilige ID der Instanz, der ConfigItemtypname als Fremdschlüssel und der Name der Instanz gespeichert. Durch die Speicherung des Fremdschlüssels ist es möglich die Daten später einem gewissen ConfigItemtypen zuzuordnen, weshalb es sich hier um eine 1 zu cn Beziehung handelt. Um nun noch die Daten des ConfigItems verwalten zu können, werden zu jeder Instanz eines ConfigItemtyps die definierten Attributtypen in der Tabelle „Attribut gespeichert. Hierfür werden die ConfigItemID und die AttributTypID jeweils als Fremdschlüssel übergeben, um einen Attributwert eines CofigItem den Attributtypen des ConfigItemTyps zuzuordnen. Weshalb es sich bei beiden um eine 1 zu cn Beziehung handelt. Zudem wird der Wert des Attributes in dieser Tabelle gespeichert, welcher somit nun über die jeweiligen Fremdschlüssel abrufbar ist.

## 1.3 Auswahl des Datenbankmanagementsystems

Nachdem entschieden wurde eine relationale Datenbank zu nutzen, sollte nun zum Zeitpunkt des Projektentwurfes ausgewählt werden, welches Datenbankmanagementsystem genutzt wird. Um nun die Datenbank im Programm zu verwenden. Dazu wurde zunächst entschieden eine MySQl Datenbank in das Programm mit einzubauen, da diese eine sehr gute Integration von verschiedenen Systemen mit sich bringt und mit einer sehr guten Performance ausgelegt ist.[[7]](#footnote-7) Dieses ist das meist verbreitete Open-Source-Software relationale Datenbankverwaltungssystem. Um diese zu Nutzen hat jeder des Backend-Teams die Anwendung XAMP heruntergeladen. Durch dieses Programm ermöglicht es dem Team einen Apache Webserver und eine MySQL Datenbank auf dem Rechner zu installieren und zu konfigurieren. Damit mit dieser gearbeitet werde konnte, wurde vom Backend-Team über den mitgelieferten PHP-Admin verschiedene Testdaten in die Datenbank importiert. Diese Datenbank wurde vom Programm angesprochen und die darin vorgespeicherten Testdaten wurde auf den ersten Oberflächentests ausgegeben. Im Zuge der Entwicklung und den Blick auf die Auslieferung des Programms entschied sich jedoch das Team statt einer MySQL-Datenbank eine H2-Datenbank in das Programm mit einzubauen. Diese Datenbank ist ohne eine Installation auf jedem System lauffähig, was auf ein sogenanntes „Rapid Prototyping“ zurückzuführen ist. Zusätzlich unterstützt diese einen „Embedded Mode“ als auch einen „Server Mode“.[[8]](#footnote-8) Jedoch sind die Hauptargumente warum sich das Team für eine H2-Datenbank entschieden haben und somit keine MySQl-Datenbank verwenden noch einmal andere. Die Gründe sind nämlich zum einem, dass nun die im Programm benutze Datenbank nicht extern vom Programm aufgerufen werden muss, wie es bei einer MySQL-Datenbank der Fall ist. Sondern die Datenbank befindet sich mit implementiert im Programm und läuft somit mit dem Programm gleichzeitig ohne Probleme im Arbeitsspeicher läuft, was mit der geringen Speicheranforderung des Datenbanksystems zu tun hat. Der andere Grund ist jedoch, dass durch die Verwendung des Spring Boot Frameworks eine einfache H2-Datenbank Einbindung schon mitgeführt wird und somit einfach mit ins Projekt eingebaut werden kann.

## 1.4 Einbau der Datenbank in das Projekt

Die Einbindung der Datenbank wird durch das Spring-Framework sehr vereinfacht. Die Datenbank liegt in diesem Projekt genau im entwickelten Programm. Um etwas genauer zu gehen, liegt diese H2-Datenbank in einer Text Datei mit der Bezeichnung „h2.hdb.mv.db“ in den Ressourcen des Maven Projekt. Die Datei wird beim Start des Programms aufgerufen und somit wird die Datenbank gleichzeitig gestartet. Die genaueren Aufrufe des Programms auf die Datenbank wird jedoch im Kapitel (Tils Kapitel zu Quellcodeerklärung) beschrieben.

# Quellen

1. <https://entwickler.de/online/datenbanken/grundkurs-datenbanken-datenbanksysteme-579859345.html>
2. <https://link.springer.com/referenceworkentry/10.1007/978-3-662-49054-9_829-1>
3. <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/>
4. Jarosch H. (2016) Die Zielstruktur: Datenbank-Modelle. In: Grundkurs Datenbankentwurf. Springer Vieweg, Wiesbaden. https://doi.org/10.1007/978-3-8348-2161-4\_4

1. Vgl. Jarosch H. (2016) S. 20 [↑](#footnote-ref-1)
2. Vgl. Jarosch H. (2016) S.19 [↑](#footnote-ref-2)
3. <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/> [↑](#footnote-ref-3)
4. <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/> [↑](#footnote-ref-4)
5. <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/> [↑](#footnote-ref-5)
6. <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbanken/> [↑](#footnote-ref-6)
7. <https://entwickler.de/online/datenbanken/grundkurs-datenbanken-datenbanksysteme-579859345.html> [↑](#footnote-ref-7)
8. <https://entwickler.de/online/datenbanken/grundkurs-datenbanken-datenbanksysteme-579859345.html> [↑](#footnote-ref-8)