**Informationspyramide**

Daten, Informationen, Wissen, Weisheitshierachie auch die DIKW-Pyramide, bezieht sich lose auf eine Klasse von Modellen zur Darstellung angeblicher strukturekker und funktioneller Beziehungen zwischen Daten, Informatik, Wissen und Weisheit. Information werden in Bezug auf Daten, Wissen in Bezug auf Informationen und Weisheit in Bezug auf Wissen definiert.

**Daten**

Daten sind die Sammlung diskreter Fakten über Ereignisse in Form von Zahlen, Zeichen und spezifischen oder relativen Werten, die die Organisation sammeln kann. Die wichtigsten Aktivitäten, die in dieser Phase durchgeführt werden müssen, sind:

* Identifizieren einer zuverlässigen Quelle zum Abrufen von Daten
* Archivieren und Löschen der entsprechenden Daten

**Informationen**

In dieser Phase geht es darum, den Daten Kontext hinzuzufügen. Die wichtigsten Aktivitäten in dieser Phase sind:

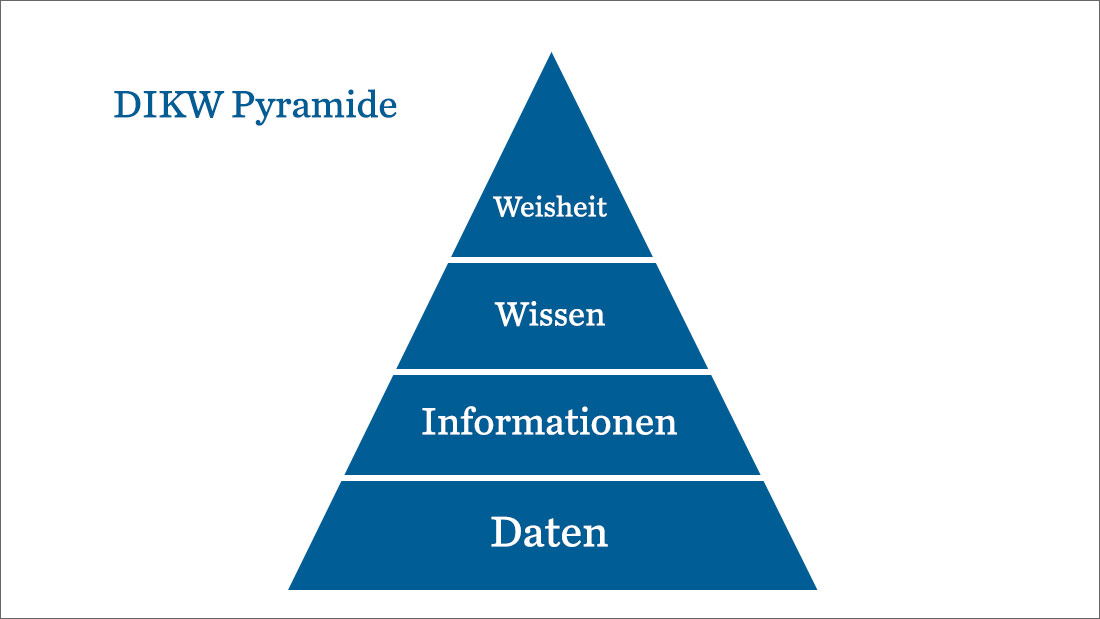
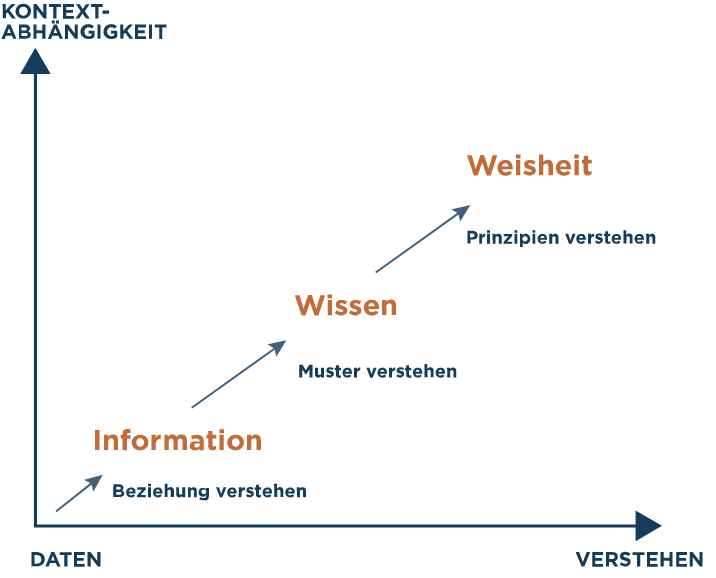
* Konvertieren von Daten in Informationen unter Wahrung der Datenintegrität
* Informationen richtig verwalten, indem Sie Endbenutzern die Suche und Verwendung erleichtern.

**Wissen**

Wissen entsteht aus Erfahrung. Dabei kann es sich um Fachwissen, Werte und Urteile von Wissensmanagern, Fachexperten, Technikern oder Endbenutzern handeln. Das in dieser Phase geschaffene Wissen ermöglicht eine intelligente Entscheidungsfindung.

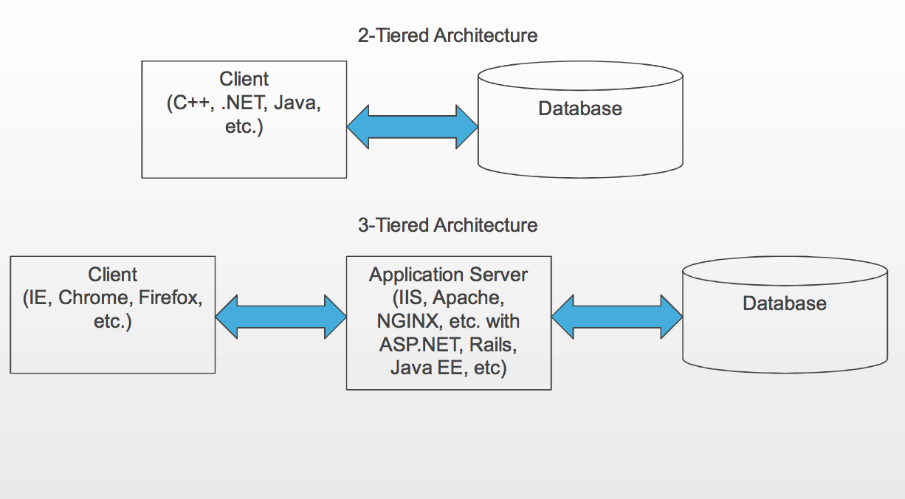
**Weisheiten**

Weisheit ist die Kulmination von Daten, Informationen und Wissen. Es wertet das Wissensmanagement auf. Dieser Wert wird durch dynamische Planung, Problemlösung, strategische Planung und Urteilsvermögen geschaffen.



**Unterschied zwischen 2. und 3. Tier Architektur**

|  |  |
| --- | --- |
| **2. Tier Arichtektur** | **3. Tier Architektur** |
| Es ist eine Client-Server-Architektur. | Es ist eine webbasierten Anwendung. |
| Bei der Zweischichtigkeit ist die Anwendungslogik entweder in der Benutzeroberfläche des Clients oder in der Datenbank des Servers (oder in beiden) vergraben. | Bei der Dreischicht befindet sich die Anwendungslogik oder der Prozess in der mittleren Schicht, sie ist von den Daten und der Benutzeroberfläche getrennt. |
| Zweischichtige Architektur besteht aus zwei Schichten: Clientschicht und Datenbank (Datenschicht). | Die dreischichtige Architektur besteht aus drei Schichten: Client Layer, Business Layer und Data Layer. |
| Es ist einfach zu bauen und zu warten. | Es ist komplex zu bauen und zu warten. |
|  |  |
| Zweischichtige Architektur läuft schneller | Die Drei-Tier-Architektur läuft langsamer. |
| Es ist weniger sicher, da der Client direkt mit der Datenbank kommunizieren kann. | Es ist gesichert, da der Client nicht direkt mit der Datenbank kommunizieren darf. |
| Es führt zu Leistungsverlusten, wenn die Benutzer schnell zunehmen. | Es führt zu Leistungsverlusten, wenn das System im Internet ausgeführt wird, bietet jedoch mehr Leistung als eine zweistufige Architektur. |



**Zugriffsvarianten auf Informationen in Datenbanken**

**Liste/CSV:**

CSV, durch Kommas getrennter Wert, Komma-Trennzeichen, CSV ist ein gängiges Dateiformat zum Speichern von Stapeldaten, das häufig zum Speichern von 1D- und 2D-Daten verwendet wird.

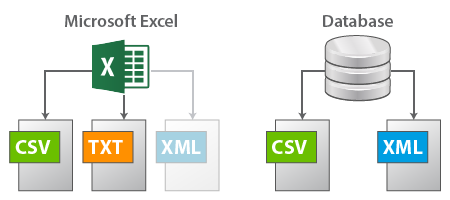


Abb. 1 CSV als Zugriffsvariante auf Informationen

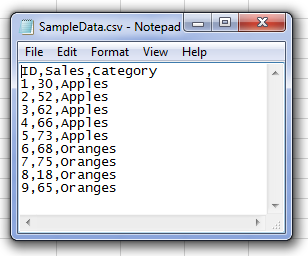


Abb. 2 CSV Beispiel

|  |  |  |
| --- | --- | --- |
|  | **Vorteile** | **Nachteile** |
| **Technische**: | Anwendbar für die meisten Programme; Nicht von anderen Programmen abhängig; | Daten sind nicht verschlüsselt; |
| **Organisatorische**: | Für einfache Verarbeitung;  Kein; Einheitlich; | Daten sind schwer zu finden; kein Überblick; |
| **Rechtliche**: |  | Fehlende Sicherheit;  Nachverfolgung der Verarbeitung ist schwierig; Alle Daten sind sichtbar; |

**Webseite:**

Um an die Daten zukommen muss erst eine Verbindung zur Datenbank gemacht. Jede Befehl läuft dann über diese Verbindung. Die benötigten Daten werden verarbeitet und an die Website gesendet. Über die Ausgabe kann selbst entschieden werden, wie sie aussehen soll.

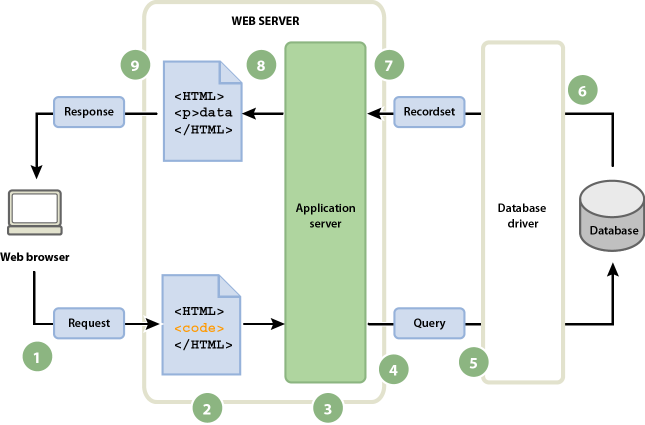


Abb. 3 Webseiten als Zugriffsvariante auf Informationen

|  |  |  |
| --- | --- | --- |
|  | **Vorteile** | **Nachteile** |
| **Technische**: |  | Benötigen einen Web-Server und Eine Website; |
| **Organisatorische**: | Eigen Kontrolle über die ausgegeben Daten; Übersichtlich;  Eigene Verantwortung; | Großer Aufwand; |
| **Rechtliche**: | Sicherheit; Nachverfolgung der Verarbeitung; Datenschutz; |  |

**Rest/Api:**

REST(Representational State Transfer) bzw. RESTful-API(Application-Program-Interface-Typ), dient dazu Daten zwischen Nutzern und Applikationen auszutauschen. Sie übermitteln die Anfrage des Benutzers an den Server oder die Datenbank und leitet die Antwort wieder zurück auf den Bildschirm.

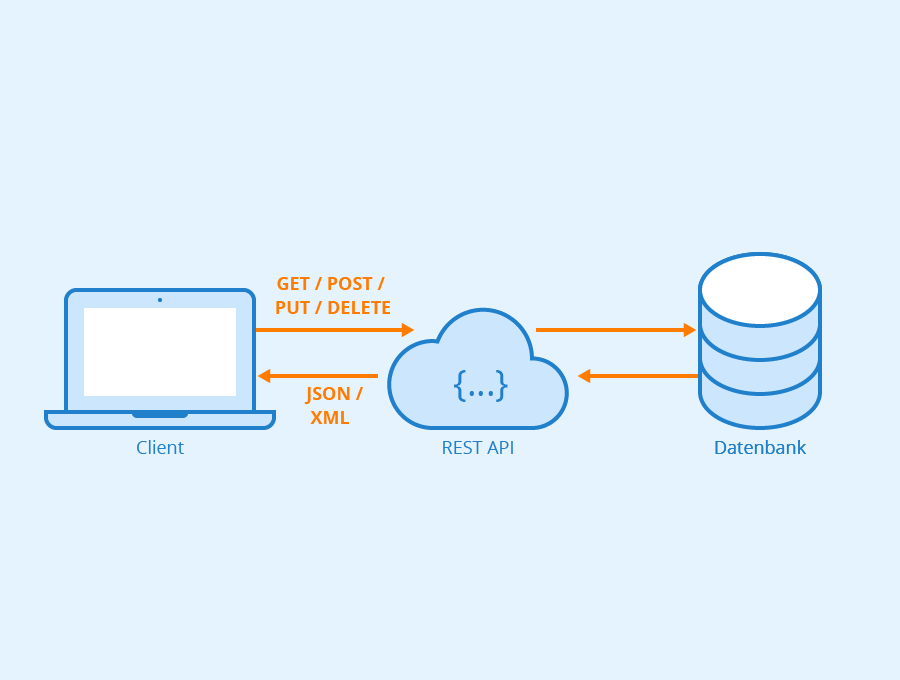


Abb. 4 REST/API als Zugriffsvariante auf Infromationen

|  |  |  |
| --- | --- | --- |
|  | **Vorteile** | **Nachteile** |
| **Technische**: | Kein direkten Zugriff auf die Daten; | Website wird benötigt;  Offene Schnittstelle; |
| **Organisatorische**: | Eigen Kontrolle über die ausgegeben Daten;  Eigene Verantwortung; Schnell und einfach; |  |
| **Rechtliche**: |  | Fehlende Sicherheit;  Kontrolle der Verarbeitung ist schwierig; |

**Drittapplikation mit Direktzugriff:**

In einigen Fällen ist der Zugriff auf die Datenbank des Server auch von ‚aussen‘ gewünscht. Das bedeutet, der Zugriff soll von extern direkt auf die Datenbank erolfgen. Dies ist z.B. nötigt wenn man von einem PC aus mit einem Administrationstool wie phpMyAdmin oder pgAdmin zugreifen möchte.

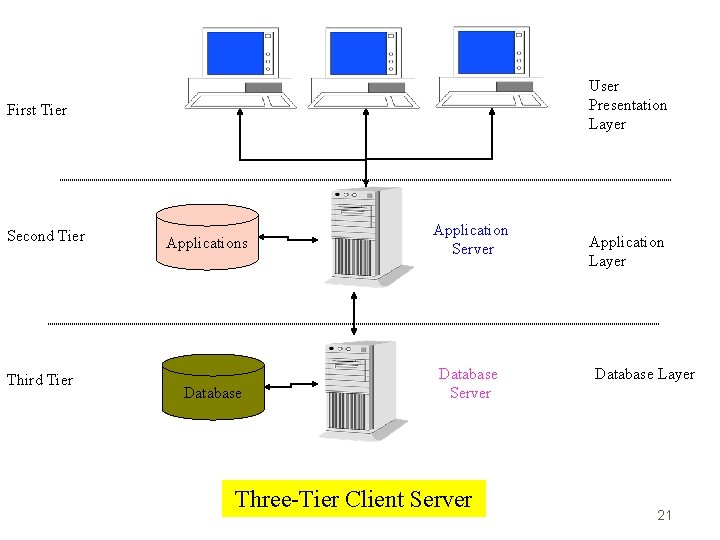


Abb. 5 Drittapplikation als Zugriffsvariante mit Direktzugriff auf Informationen

|  |  |  |
| --- | --- | --- |
|  | **Vorteile** | **Nachteile** |
| **Technische**: | Kein Aufwand; | Beschränkte Funktionalität;  Abhängigkeit von anderen Programmen; Client kommuniziert direkt mit der Datenbank; |
| **Organisatorische**: | Schnellen Zugriff; Benutzerfreundliche Anwendung; einfache Inbetriebnahme; |  |
| **Rechtliche**: | Eigene Verantwortung; | Fehlende Sicherheit;  Kontrolle der Verarbeitung ist schwierig;  Folgen der Richtlinien; Kosten; |

**Drittapplikation mit View:**

In der Regel greift man bei einer Datenbank direkt auf eine Tabelle zu, um Daten zu ändern, hinzuzufügen oder zu löschen. Manchmal ist aber der Zugriff direkt auf die Tabelle nicht erwünscht, sondern es steht eine View dazwischen, die die Daten aufbereitet. VIEWs sind Abfragen, die in der Datenbank als Objekt fest gespeichert sind. Sie können als virtuelle Tabellen verstanden werden und können anstelle einer „echten“ Tabelle verwendet werden.

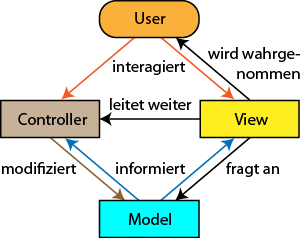


Abb. 6 Drittapplikation mit View als Zugriffsvariante auf Informationen

|  |  |  |
| --- | --- | --- |
|  | **Vorteile** | **Nachteile** |
| **Technische**: | Kein Aufwand; Direkt an der Datenbank angebunden; | Beschränkte Funktionalität;  Abhängigkeit von anderen Programmen; |
| **Organisatorische**: | Geregelter Datenverarbeitung;  Kontrolle der Zugriffsfreigabe; | Sicht auf alle Daten; |
| **Rechtliche**: | Eigene Verantwortung; | Fehlende Sicherheit; |