**Zusammenfassung "Webbasierte Datenbankanwendungen"**

1. Vorlesung "Versionsmanagement"
   1. **Änderungsmanagement**

beschäftigt sich damit, die Änderungsanträge ( change requests) der Kunden und Entwickler für die Software im Auge zu behalten, die Kosten und Auswirkungen dieser Änderungen zu ermitteln und zu entscheiden, ob und wann diese Änderungen implementiert und ausgeliefert werden sollen.

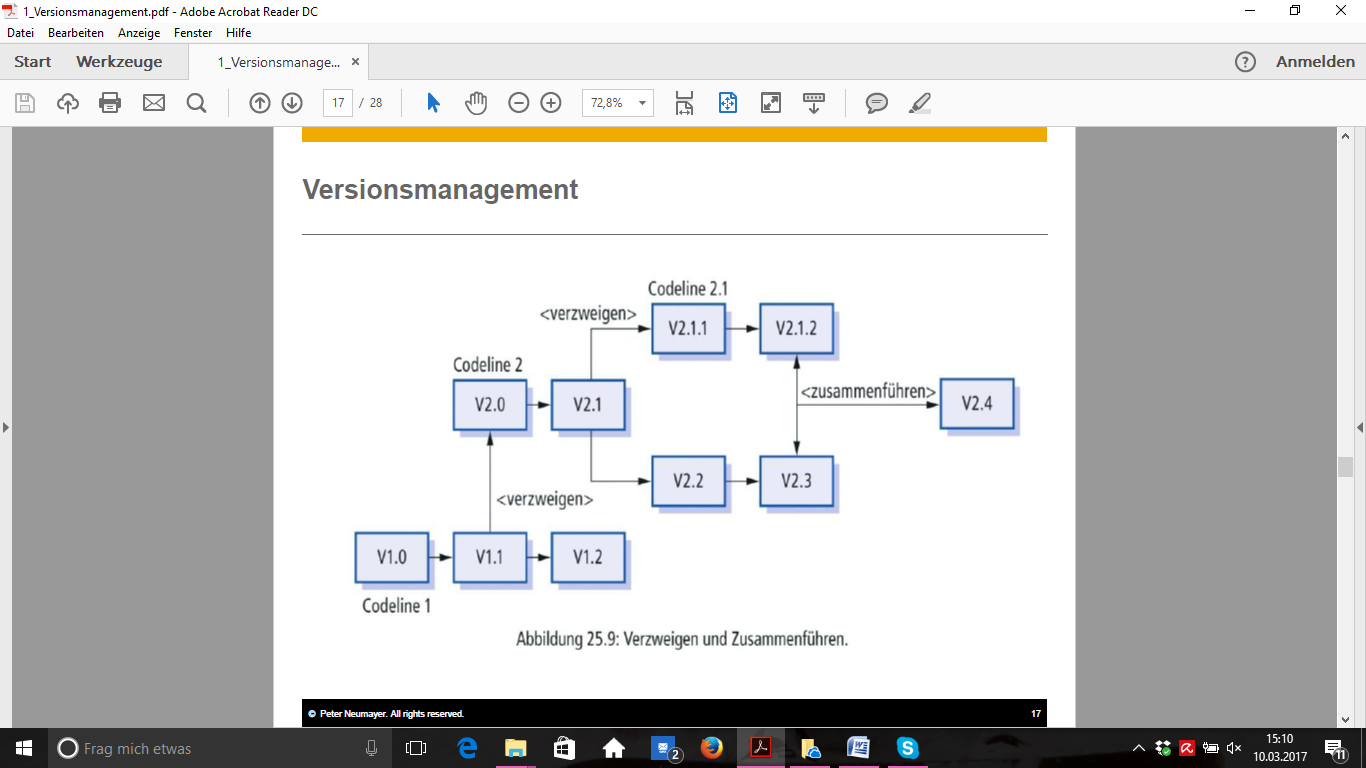
Bsp. Erweiterung der Software um eine Funktionalität oder beheben eines Fehlers

* 1. **Versionsmanagement (Dateien und Objekte zusammenbauen)**

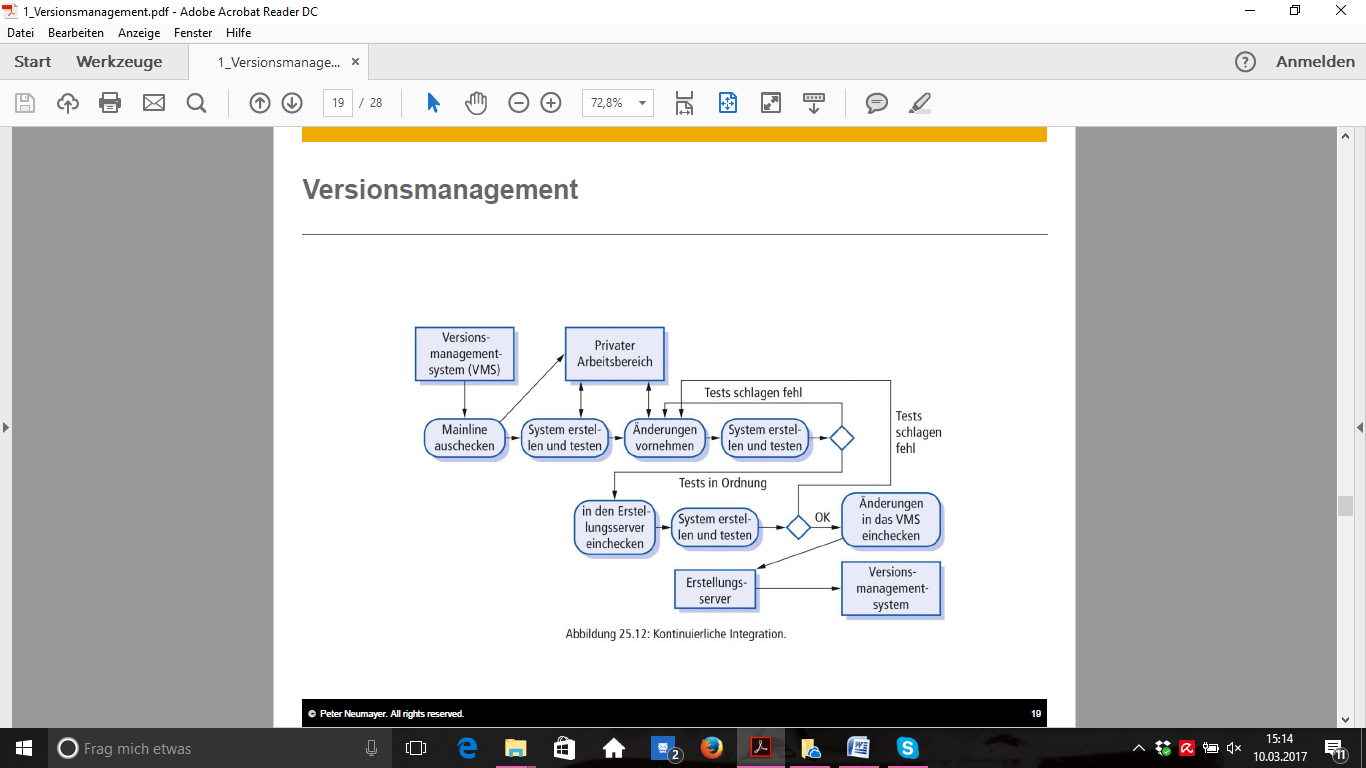
verwalten mehrere Versionen von Systemkomponenten und sicherstellen, dass die Änderungen verschiedener Entwickler an den Komponenten sich nicht gegenseitig beeinträchtigen.

Hier sollte darauf geachtet werden dass man die aktuellsten Versionen der einzelnen Komponenten nutzt.

Eine **Version** ist eindeutig benannt und mit einer Versionsnummer versehen.

****

**Codeline:** eine Folge von Quellcodeversionen, die aufeinander aufbauen (entsteht z.B. durch parallele Programmierung an einer Komponente)

**Baseline:** spezifiziert die Komponentenversionen, die zu einem System gehören mit Angabe zu Bibliotheken und Konfigurationsdateien.

Unser **Versionskontrollsystem** = Github

Jeder hat einen privaten Arbeitsbereich, um ihn mit allen im öffentlichen Arbeitsbereich zu teilen checken wir ein (Push) und aus (Pull).

**Systemerstellung**, durch Kompilieren und Linken der Komponenten und Bibliotheken zu einem volllauffähigen Programm

* 1. **Fragen**
* Unterschied zwischen Änderungsmanagement und Versionsmanagement erklären
* Erklärung der Begriffe Versionsmanagement und Version
* Arbeitsweise und typischen Arbeitsschritte bei der Verwendung eines lokalen Arbeitsbereichs beschreiben
* 3 verschiedene Versionsmanagementsysteme nennen
  + GitHub,Perforce, Bitkeeper, ClearCase, und Trac, RedmineOpenProject
  1. **Begriffe**

**Repository**: kann bei GitHub einfach als „Projekt“ verstanden werden. Die Dateien für ein Software-Projekt werden in einem Repository abgelegt.

**Branch**: Innerhalb eines Repositories kann es mehrere Versionen einer Software geben. Z.B. eine experimentelle Beta-Version und eine stabile Version für den Produktiv-Einsatz. Jede Version stellt dabei einen Branch („Ast“) des Repositories dar.

**Commit**: nennt sich der Vorgang, wenn eine neue Version eines Branches bei der Versionsverwaltung eingereicht wird. Das heißt, der Entwickler „committed“ eine Änderung.

**Pull Request**: ein Entwickler möchte eine Änderung pullen uns stellt einen „Pull Request“ an den Administrator. Dieser kann sich die Änderungen dann ansehen und über den Pull entscheiden.

**Fork**: wenn jemand einen eigenen Ableger eines Repositories hat. Hier kann man privat vor sich hin entwickeln und seine Version am Ende wieder dem ursprünglichen Projekt zuführen oder eine eigenständige Variante verbreiten.

1. Vorlesung "UI Design Prozess"
   1. **Benutzbarkeit**

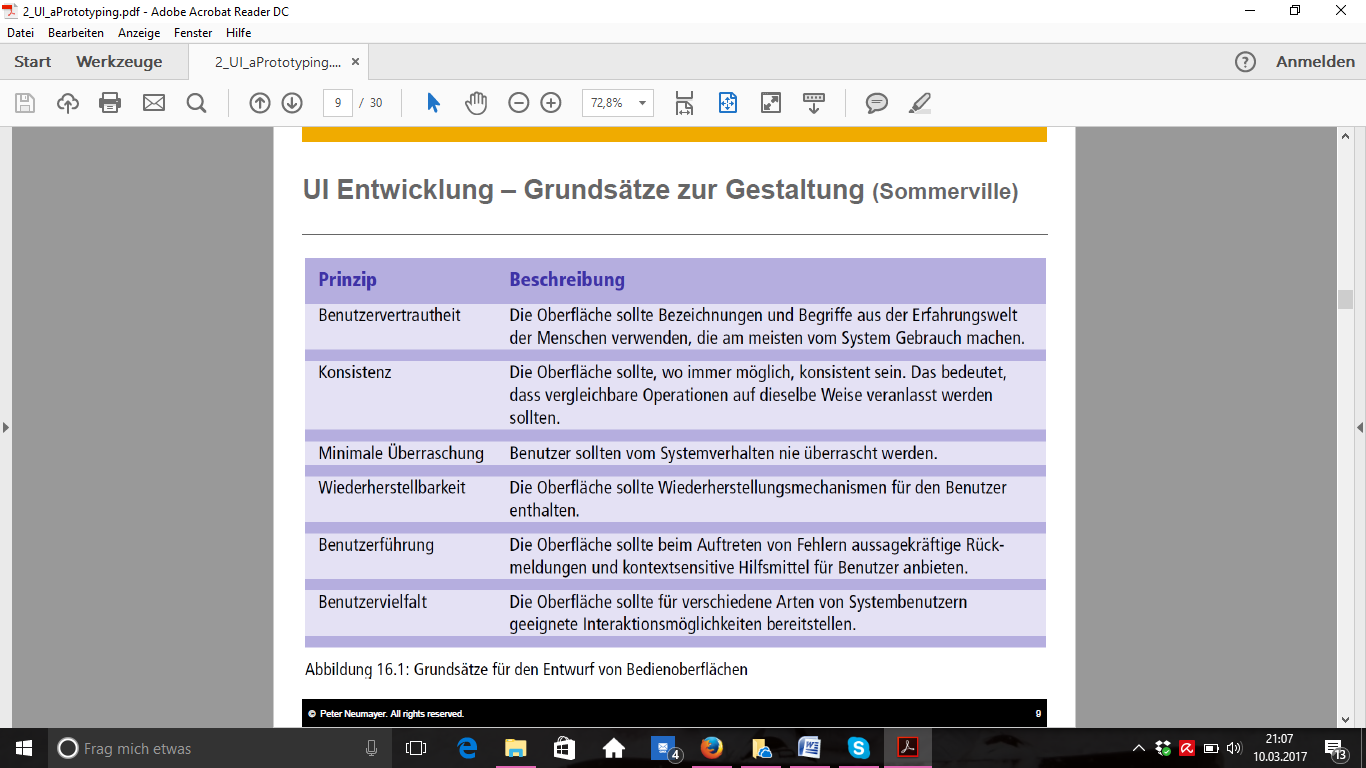
Fähigkeit des Softwareprodukts, vom Benutzer verstanden und benutzt zu werden sowie für den Benutzer erlernbar und ‚attraktiv‘ zu sein.

**Merkmale**

* **Verständlichkeit** (einfache Gestaltung, Benutzerfreundlichkeit)
* **Erlernbarkeit** (intuitive Nutzung oder Erklärungen/Handbücher für den Nutzer)
* **Bedienbarkeit** (angepasst an den Nutzer und dessen Umgebung)
* **Attraktivität** (ansprechende Gestaltung, Interesse wecken)
* **Konformität** (Standards und Stilvorgaben werden eingehalten)

**Prinzipien**

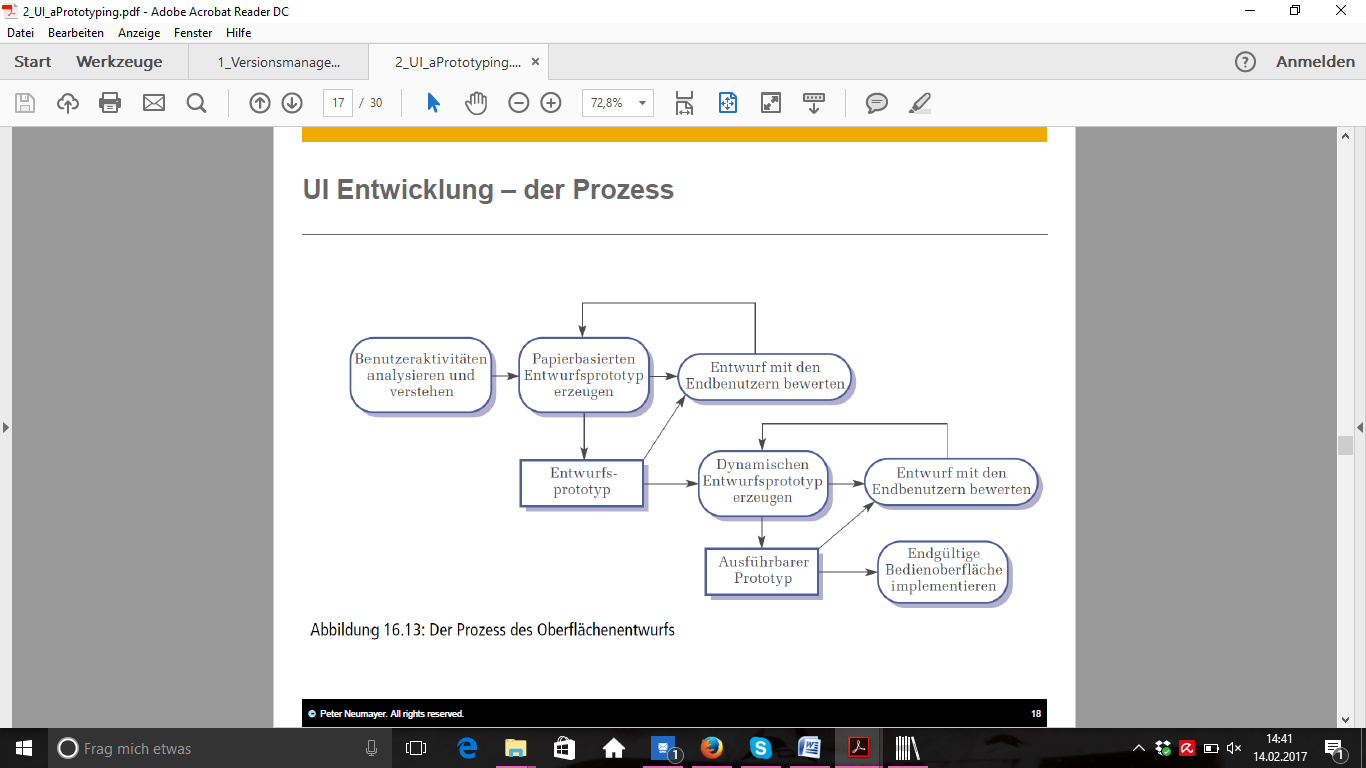
* **Prinzip der Verbalisierung**: Aussagekräftige Namensgebung, geeignete Kommentare
* **Architekturprinzip Ökonomie**: nur die Funktionalität, die der Benutzer zur Erledigung seiner Aufgaben benötigt. = bessere Benutzbarkeit
* **Architekturprinzip** **Trennung von Zuständigkeiten**: kein HTML in JavaScript und kein Stylesheet in HTML



* 1. **Ergonomie**

wissenschaftliche Sicht zur Erforschung den Dialog zwischen User und Software

* **Aufgabenangemessenheit** unterstützt den User bei der Aufgabenerledigung
* **Selbstbeschreibungsfähigkeit** Der User weiß wo er sich befindet und wo er wie zu seinem Ergebnis kommt (Tooltips, Schaltflächen erkennbar)
* **Erwartungskonformität** Die Anwendung verhält sich, wie vom User erwartet wird, entspricht Standards
* **Lernförderlichkeit** Unterstützt und leitet bei Erlernung der Nutzung
* **Steuerbarkeit** Der User kann selbst entscheiden wie er von A nach B kommt
* **Fehlertoleranz** Fehlererkennung und Korrektur
* **Individualisierbarkeit** an individuelle Fähigkeiten und Bedürfnisse des Users anpassbar

****

* 1. **Oberflächenentwurf**

* **Benutzeranalyse** Welche Aufgaben müssen in welchem Arbeitsumfeld erfüllt werden?
* **Erstellen eines Systemprototyps,** um herauszufinden was der User wirklich braucht.
* **Oberflächenbewertung** Bewertung der User-Erfahrungen sammeln
  1. **User-Centered Design**

**Prozess**

1. **Plan**: Zeit, Budget, Personen planen

2. **Research**: Marktchancen und User-Bedürfnisse untersuchen

3. **Design**: Use-Case schreiben UI-Prototypen

4. **Adapt**: Bei der Entwicklung auf Design und Richtlinien achten

5. **Measure**: Die Benutzbarkeit messen

**Prinzipien**

* Focus on real end-users
* Validate requirements & designs
* Design, prototype, & develop products iteratively
* Understand & design for a holistic user experience

**Use-Case, User Story und Persona** (ich denke ihr wisst was das ist)

* 1. **Lernziele / Fragen**
* Benutzbarkeit erklären können, deren Merkmale und Prinzipien nennen können
* Benutzbarkeit und Ergonomie erklären können
* Ergonomie und Grundsätze der Dialoggestaltung erklären können
* Schritte im UI-Entwicklungsprozess beschreiben
* Prinzipien von User Centered Design nennen
* Einen agilen Entwicklungsprozess für die Benutzerschnittstelle beschreiben und anwenden können

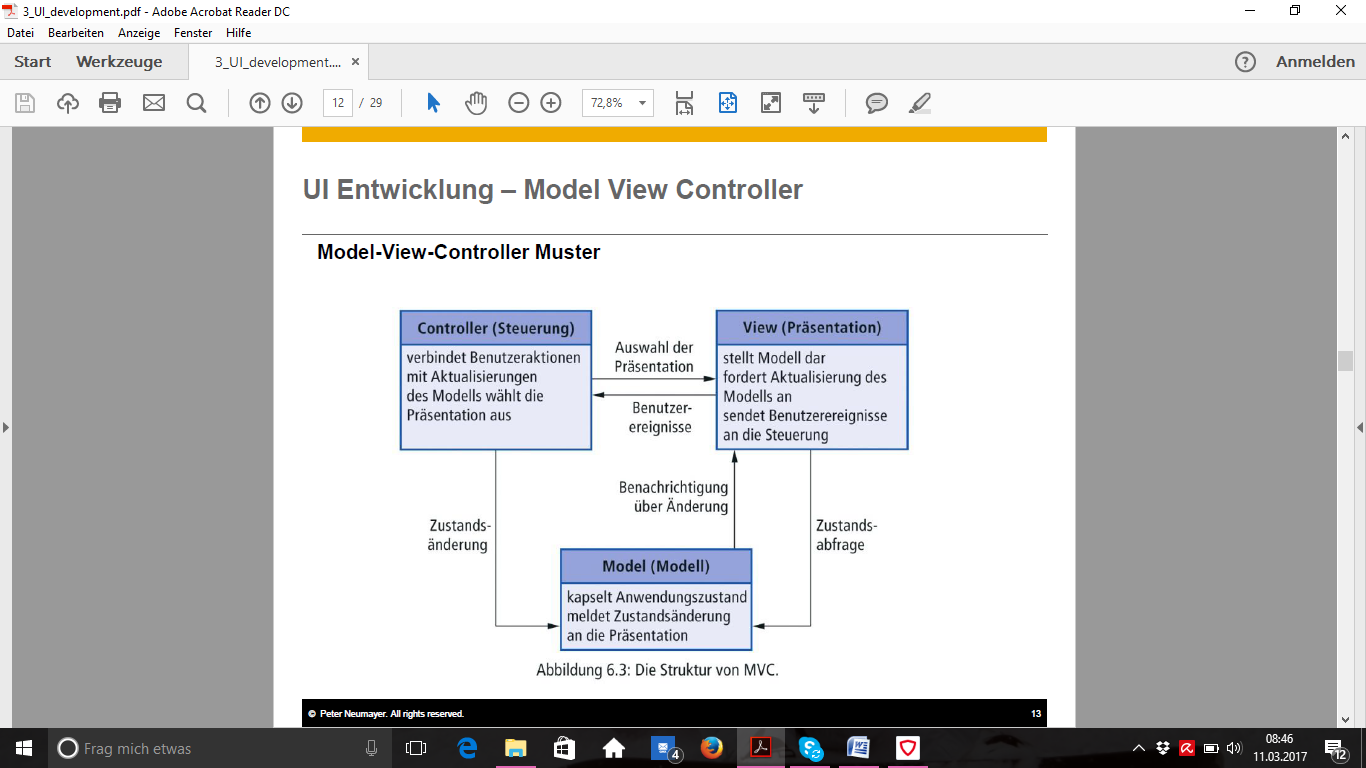
1. Vorlesung

**Unser Framework: AJAX** bezeichnet die asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden.

* 1. UI Entwicklung / Ereignisgesteuerte Programmierung

**Inversion of Control**, das Programm wartet nicht auf ein Ereignis, sondern ein eigenständiger Kontrollfluss wird durch das Ereignis aktiv (z.B. durch Threads). Wir nutzen dies im Rahmen des Servlets.

**Nebenläufigkeit** parallel Prozessornutzung, hierbei muss die Unabhängigkeit der Programmteile **sichergestellt** werden.



* 1. MVC

Einteilung des Systems in 3 logische Komponenten.

(+) Unabhängigkeit der Elemente.

(-) Kann aufwendiger und komplexer sein

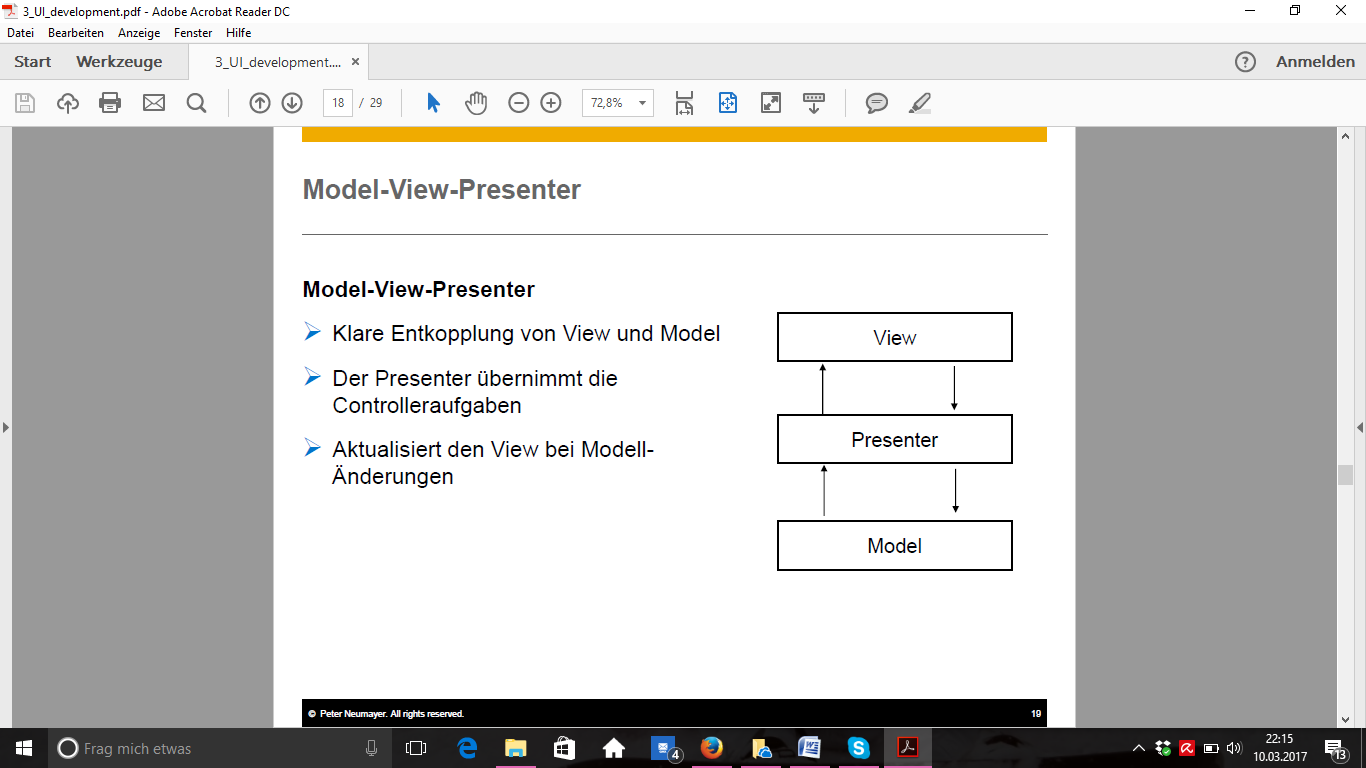
**Model** -Erzeugt die anzuzeigenden Daten

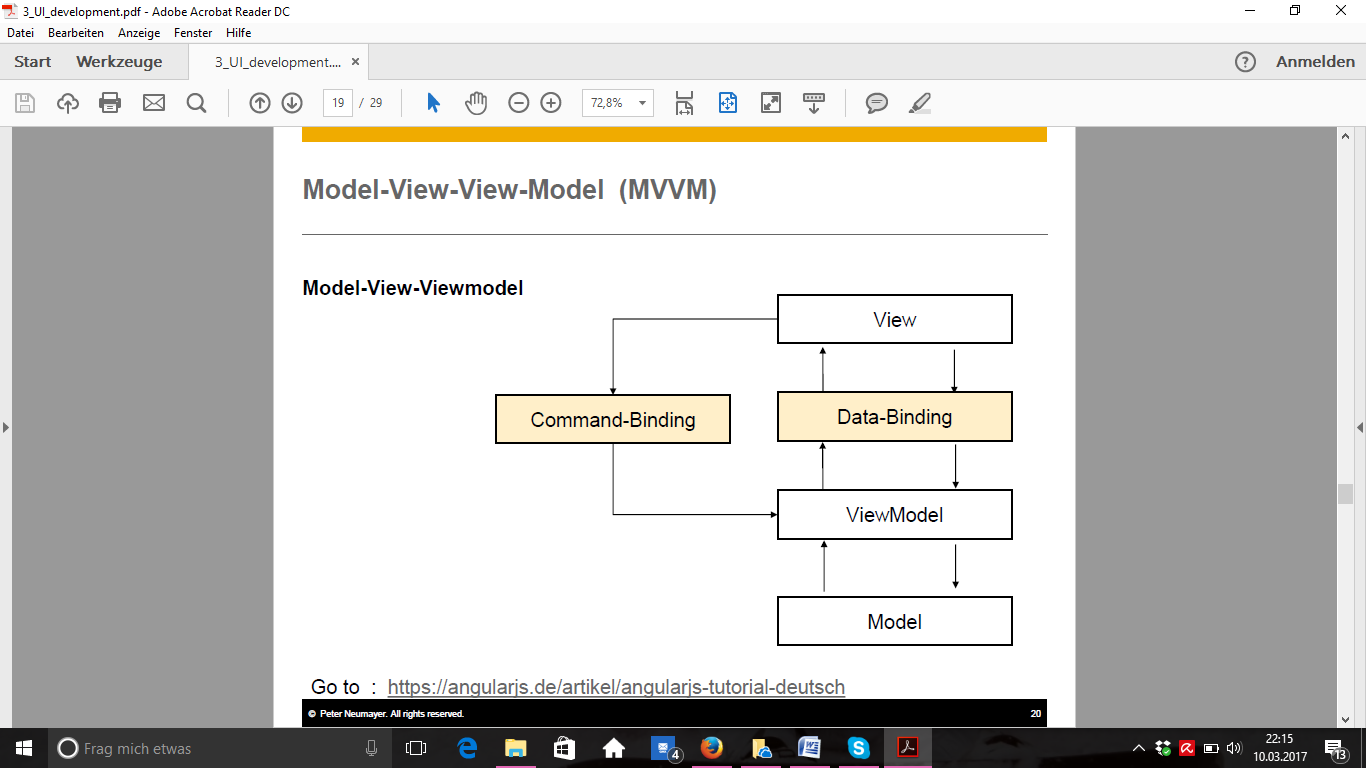
- unsere Datenbank

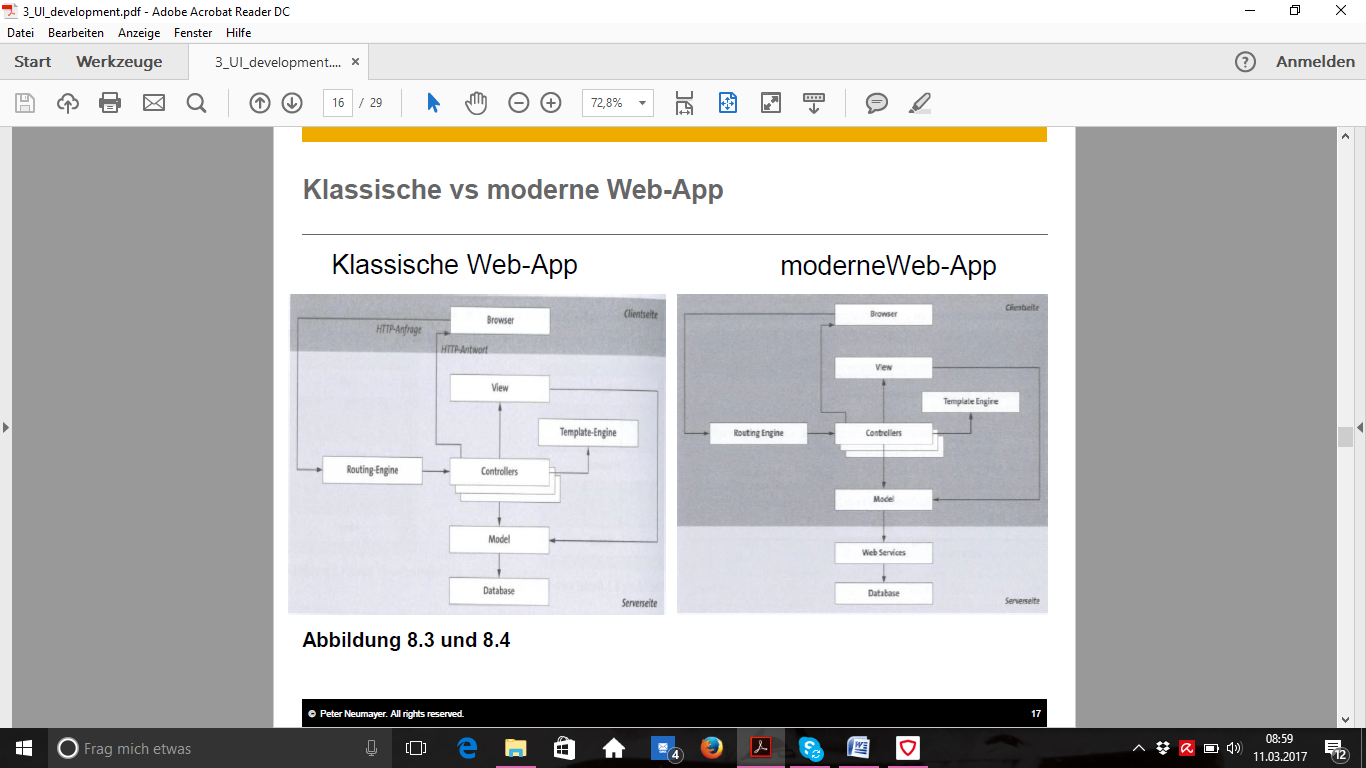
**View** - Kümmert sich um das Anzeigen der Daten

- Weboberfläche (HML und CSS)

**Controller** - Definiert die Systemreaktionen auf User-Eingaben

 - unser JavaScript und die Java-Klassen

* 1. Model-View-Presenter
* Entkoppelung von View und Model
* Pressenter = Controller
  1. Model-View-Viewmodel
* **View**: die Oberfläche
* **ViewModel**: Das Modell hat alle Daten, die im View zu sehen sind.
* **Model**: Das Programm mit Methoden und Diensten.
* Soll im View etwas geändert werden muss es zunächst im ViewModel geändert werden
  1. klassische und moderne Web-App

In der modernen Web-Anwendung, spielt sich vieles auf der Clientseite in JavaScript ab. Aus XML wird JSON

* 1. **Lernziele / Fragen**
* MVC Pattern und MVVM unterscheiden und erklären können
* MVC an unserem Projekt
* Unterschied einer klassischen und modernen Webapplikation

1. Vorlesung
   1. **REpresentational State Transfer**

HTTP-Datenübertragung ohne eine Sitzungsverwaltung über Cookies (stateless)

Identifikation einer Ressource (URI)

**Prinzipien**

* Unterstützung der **Client –Server** Architektur
* **Stateless** = Zustandslosigkeit (es werden keine Zustandsinfos zwischen zwei Nachrichten gespeichert, z.B. keine Cookies oder Sessions) ( gut für die Skalierbarkeit )
* **Caching** (Ein Puffer für die Daten fördert die Performance)
* **Einheitliche Schnittstelle** (Hauptmerkmal):
  + **Adressierbarkeit von Ressourcen** (Jede Information per URI ist eine Ressource)
  + **Repräsentationen zur Veränderung von Ressourcen** (Eine Veränderung der Ressource ist immer auch in der Repräsentation darzustellen: HTML,JSON, XML, etc)
  + **Selbstbeschreibende Nachrichten** ( HTTP Methoden)

**Hypermedia as the Engine of Application State (Entwurfsprinzip)**

der Client navigiert ausschließlich über URLs (Zustandsübergänge)

Zustände sind durch Ressourcen dargestellt

* 1. **Open Data Protocol**

standardisiertes Protokoll auf Basis von HTTP und REST ist. Unterstützt Create, Read, Update und Delete, um Operationen auf Daten zu ermöglichen

* 1. **Lernziele / Fragen**
* Prinzipien und Vorteile von RESTful Webservice
* RESTful Services auf Basis des oData Standards aufrufen können

Lernziele

Sie sollen am Beispiel einen Änderungsmanagement-Prozess kennen und erläutern können

Sie sollen den typischen Versionsmanagementprozess kennen und erläutern können

Sie sollen GitHubals ein bekanntes Versionsmanagementsystem benutzen können