Einheit 7

Fullstack - Ausblick

FH University of Applied Sciences TECHNIKUM WIEN

Agenda

- Überprüfung zur Hausübung
- Feedback zu BEB
- next Steps für einen Fullstack Developer:
  - Backend: Laravel als PhP Framework
  - Frontend: Unterschiede Vue.js, react.js und angular
  - Clean Architecture Pattern von Robert C. Martin

Laravel

technologische Basis

- Namespaces und Autoloader
- Composer
- Blade

Laravel

- Hintergrund
- Aufbau und Ablauf
- Tutorials
- Alternativen

Namespaces und Autoloader in PhP

- Namespaces wie in anderen Programmiersprachen auch

- Autoloading in PhP selber organisieren: (Autoloading - Manual)

```php
<?php
define("appRoot", filter_input(INPUT_SERVER, 'DOCUMENT_ROOT', FILTER_SANITIZE_URL).API_PATH);

ini_set('unserialize_callback_func', '__autoload');

function __autoload($className) {

    // convert namespace to full file path
    $class = "/" . str_replace("\\", "/", $className) . '.php';
    //remove project name (it is root folder name and inside the app)
    $classPath = appRoot . str_replace("            /", "", $class);

    if(file_exists($classPath)){
        require_once($classPath);
    } else {
        error_log("trying to load ".$classPath.", but file not found");
    }
}
```

Composer Packetverwaltung

- ein s.g. anwendungsorientierter Paketmanager
- Installation als Service am System (unter windows *.bat)
  Introduction
- Aufruf auf der Command Line (CLI)
- Definition der Pakete in JSON File
- Einbinden via Namespaces (autoloader)

Blade: Templating Engine

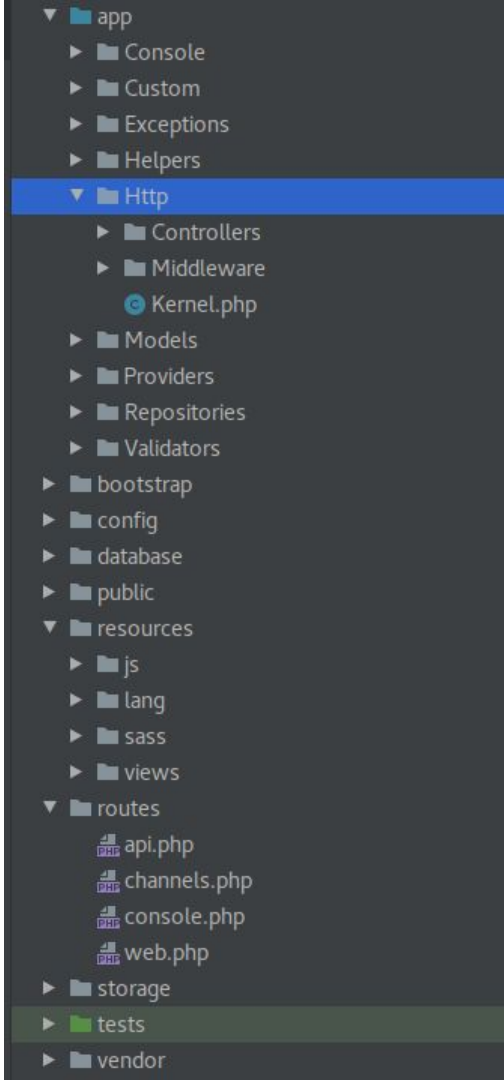
- Von Laravel eingesetzt (https://laravel.com/docs/7.x/blade#introduction)
- um HTML Templates mit Variablen zu bestücken
- Templates können gecached werden
- Nachteil: Stärkere Koppelung Client und Server
- standalone Alternative: Home - Twig - The flexible, fast, and secure PHP template engine

Laravel 7 - Hintergrund

- besteht seit 2011 (Erfinder: Taylor Otwell)
- derzeit in Version 7.x
- API fähig
- Ortwell kommt aus dem Symfony Umfeld
- Ziel: Rasches und solides erstellen von Webanwendungen

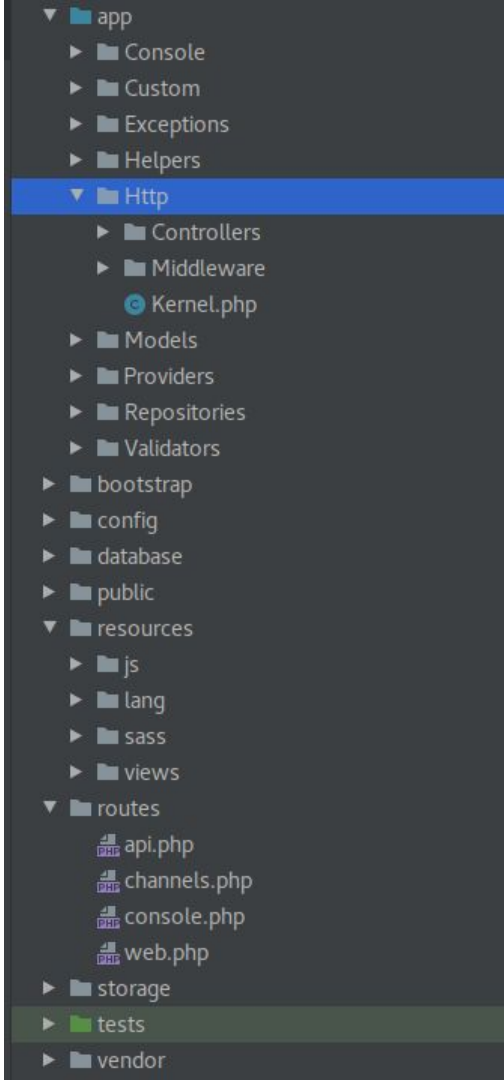
Laravel - Aufbau

- wird via Composer aufgesetzt oder Homestead
- baut auf MVC Pattern
- Trennt Services in vendor lib verzeichnis
- JS Integration oft via npm oder node,
  CSS mittels Less oder SASS



Laravel - Ablauf

- index.php in /public ist Ausgangspunkt der Anwendung
- routes geben durch eigene Syntax Wege zu Controllern vor
- Controller bedient sich der Repositories und erzeugt HTML oder JSON Views



Tutorials

- Installation via Homestead: Homestead - Laravel - The PHP Framework For Web Artisans
- Dokumentation: Installation
- Basic Task List (Leider keine Tutorial für 7.x, aber ausreichend)
- Topic: Laravel (kostenpflichtige Tutorials, sehr viel freier Community Support)

Alternativen

- Symfony

  Symfony ist ein in PHP geschriebenes Webframework und eine Menge von wiederverwendbaren PHP-Komponenten/Bibliotheken. (Elegant und Modern !! Gute Alternative zu Laravel) **2005**

- Zend

  Das Zend Framework ist ein komponenten-orientiertes Webframework für PHP. Klassen und Pakete können unabhängig voneinander und auch in Kombination mit den Lösungen anderer Hersteller genutzt werden. (Eher Large Scale) **2005**

- YII

  Yii ist ein freies, objektorientiertes, komponentenbasiertes Webframework, das in PHP geschrieben ist. **2008**

- CodeIgniter

  CodeIgniter ist ein in PHP geschriebenes quelloffenes leichtgewichtiges Webframework. **2014**

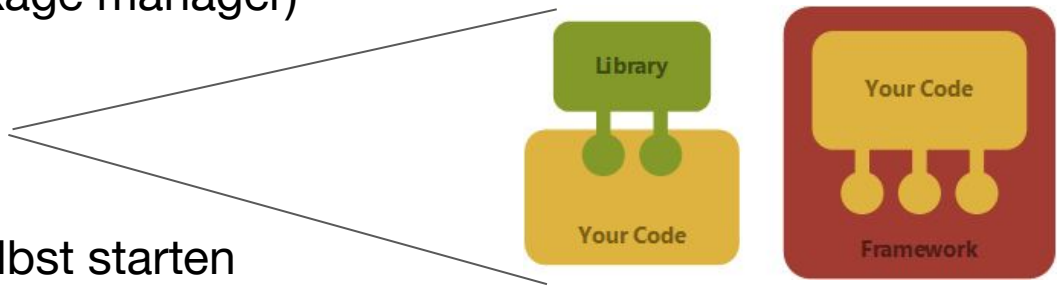
FH University of Applied Sciences TECHNIKUM WIEN

JavaScript Frameworks

Single Page Applications

Wichtige Begriffe vorab

- SPA: single-page-application
- typescript: Aufwertung von JS, kompiliert nach ECMAScript
- npm (the Node.js package manager)
- Framework vs. Library
- bootstrapping: sich selbst starten
- compiling to JS



vue.js

versus

react.js

versus

angular

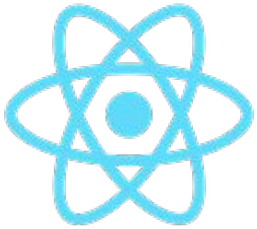
angular (ab V.2)

- 2010 von Google entwickelt (derzeit V9)
- typeScript basiertes JS Framework
- Komponenten als Marker auf DOM Elemente
- Diesen s.g. Direktiven kann man Verhalten beibringen
- Zeitweise gilt es als langsam und schwer zu debuggen
- Erste Schritte

Syntax Beispiel

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    {{title}}!!
  </h1>
  <img width="300" src="
```

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Hello World';
}
```



react.js

- Seit 2013 von Facebook (derzeit V. 16.x)
- ist eher eine Bibliothek als ein Framework
- Bekannt für die starke Performance dank Shadow DOM
- UI und Interaktion werden im selben Code geschrieben
- Erste Schritte

Syntax Beispiel

```
ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```

vue.js

- Seit 2014 von Evan You (derzeit in V.2.6)
- kann auch mit typescript arbeiten
- vereint shadow DOM und Komponenten Ansatz der beiden anderen
  Also sehr schnell, sehr flexibel
- leichter erlernbar als react und angular
- Achtung: vue funktioniert am besten, wenn man sauber implementiert
- Erste Schritte

Syntax Beispiel

HTML

```
<!-- development version, includes helpful console warnings -->
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

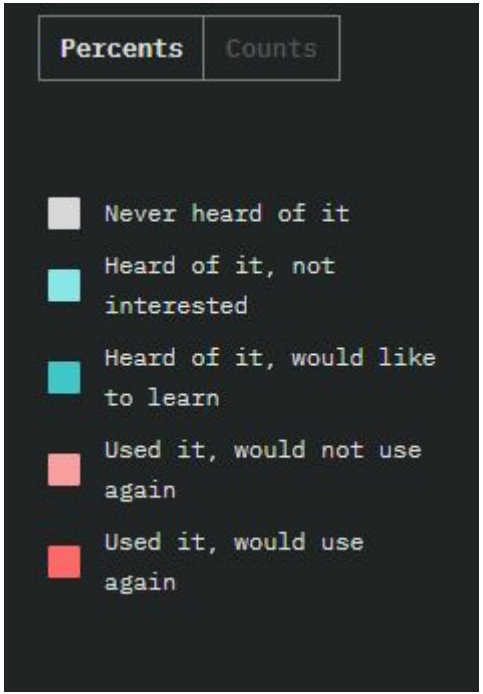
HTML

```
<div id="app">
  {{ message }}
</div>
```

JS

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Lernkurven


Percents
Counts
Never heard of it
Heard of it, not interested
Heard of it, would like to learn
Used it, would not use again
Used it, would use again


1
R
React
2
V
Vue.js
3
Ng
Angular
9.2%
19.1%
64.8%
20.5%
46.6%
28.8%
31.8%
10.4%
33.8%
23.9%
React
Vue.js
Angular

Fazit

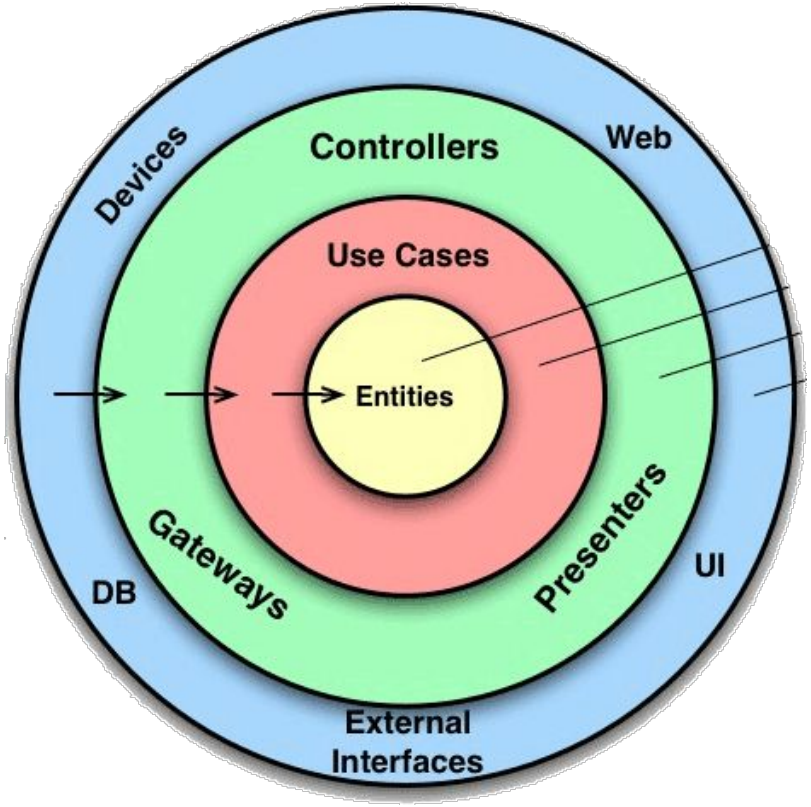
- Alle drei Frameworks haben ihre Daseinsberechtigung
- je nach Anwendungsfall fällt Entscheidung anders
- Mein Fazit:
  - Karriere- (insb. Gehalts-) technisch (dann aber im BE): react.js
  - Rasche Weiterentwicklung als Developer: vue.js
  - verlässliche large scale Anwendungen: angular.js

FH University of Applied Sciences TECHNIKUM WIEN

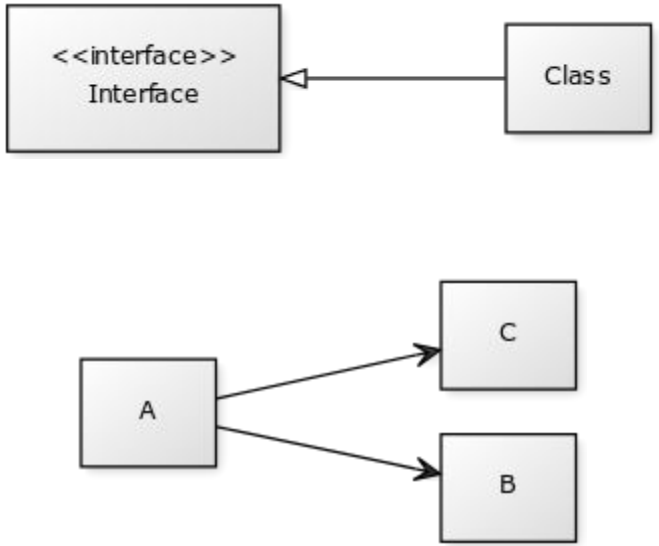
Clean Architecture
nach Robert C. Martin

Grundidee

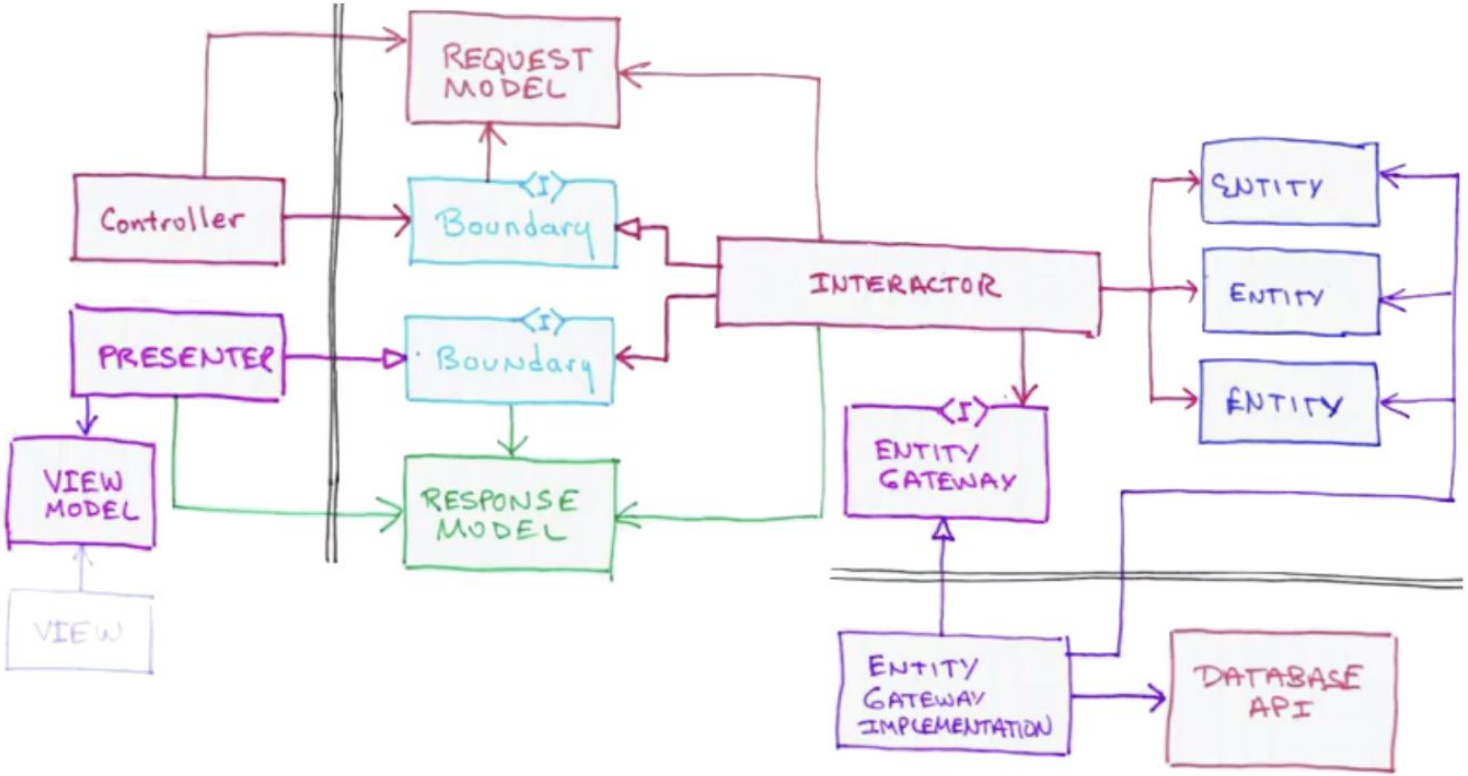
- alles wird rund um die Kernlogik aufgebaut
- nach Außen werden Objekte zunehmend
    - Abstrakter
    - Austauschbarer
- Abhängigkeiten verlaufen von außen nach innen
- Ziel: Austauschbarkeit, Testbarkeit
- Naming von R. Martin etwas gewöhnungsbedürftig



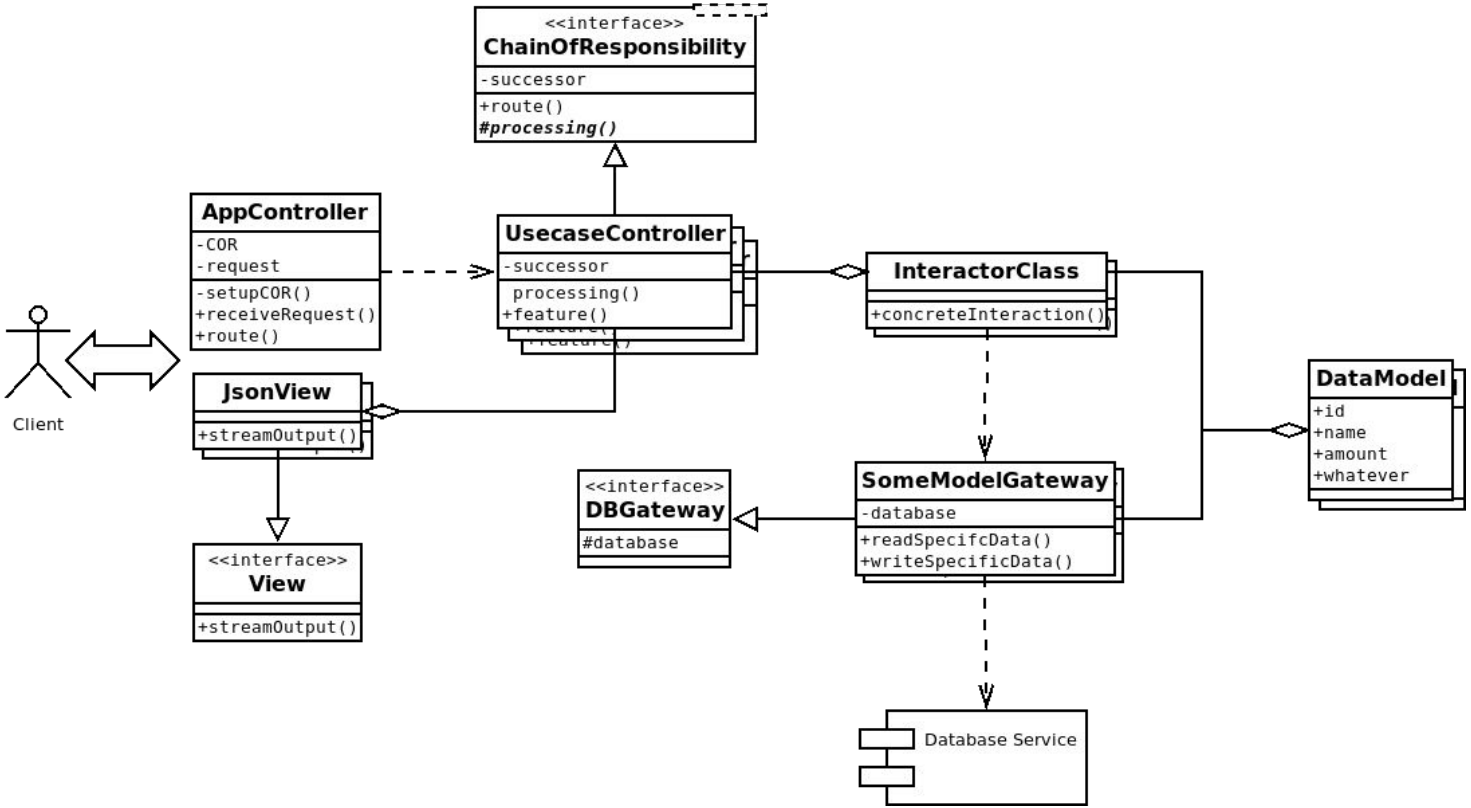
UML Grundlagen



Since a picture worth a thousand words here is Uncle Bob's UML class diagram representing the structure (and the data flow between the concerned components) of the *Clean Architecture*:



In PhP herunter gebrochen



FH University of Applied Sciences TECHNIKUM WIEN

Danke für die Aufmerksamkeit