chumm-uffa 23.12.2017

Entscheide

- Frontend und Backend werden separiert. Sie werden in unterschiedlichen Container ausgeführt.
- Das Styling wird erst am Schluss entschieden. Zuerst werden Routing und Funktionalität erstellt. (Design follow function)
- Schnittstelle zwischen Frontend und Backend werden via API-Projekt zur Verfügung gestellt. Somit wird sicher gestellt, dass die übertragenen Daten auf beiden Seiten verstanden werden.
- Rest Service URI beginnt immer mit /api. Somit kann das Routing von Frontend zu Backend sicher gestellt werden.
- Die Suche muss separat nochmals beschrieben werden, da diese zu kompliziert ist.
- Jede Seite wird als separates Modul umgesetzt.
- Pullrequest werden gegenseitig gemerged.
- Im Commit immer Issue nummer vermerken. Somit wir die Verbindung von Commit zu Issue hergestellt.
- Entwicklungsstrategie ist, möglichst dumme Komponenten zu haben und möglichst alles in Service aus zu lagern. Unittest werden nur für Services erstellt, aber nicht für Komponenten.
- HTTP Zugriffe mit Angula HTTP Module implementieren.

Main UseCase

- Einfacher Ansatz. Ein Benutzer registriert sich. Anschliessend kann er eigene Meetups erfassen oder sich auf bereits vorhandene Meetups registrieren. Der Owner des Meetup kann dann entscheiden, ob er einen anderen Benutzer mitnehmen möchte oder nicht.
- Die auf ein Meetup registrierten Benutzer können via Chat Informationen austauschen.

Vorgehen

- Alle Seiten mit Routing erstellen
- Seiten mit HTML Elementen füllen
- Services erstellen

Fragen

- Welche DB wollen wir nehmen. NoSql oder SQL
- NgBootstrap Framework oder anderes Styling Framework verwenden erlaubt? => Responsive!!

Antwort Silvan vom 28.11

- Eine relationale DB mit node hmmm könnte ungemütlich werden.
- Mongo mit Mongoose wäre ok , zur Not auch Nedb aus dem Projekt 1, performt halt überhaupt nicht.email
- responsiv wäre gewünscht -> er hats nicht explizit gefordert.
- boostrap Framework wäre ok er findets aber nicht sooo toll. Grundsätzlich können wir soviele Frameworks nehmen was wir wollen.

- Empfehlung : wir sollen im Angular *.scss und nicht *.css Files einsetzen. Css Code ist im Scss kompatibel.

Entscheide vom 02.12

- Wir verwenden Mongo mit Mongoose im Backend als Datenbank. Bei Problemen kann Silvan sicher Unterstützung geben.
- Design wird erst im Januar entschieden. Wir konzentrieren uns weiter auf Funktionalität und Inhalte. Mit dem Design-Entscheid wird auch CSS-Framework/Responsive mit entschieden.
- In Angular wird ausschliesslich SCSS Datei-Format verwendet.
- Weiteres vorgehen:
  - Christoph macht weiter im Frontend Bereich mit der Detail Ansichten und Routing.
  - Dani kümmert sich um Backend/Datenbank und setzt Login/User Verwaltung mit Backend Anbindung um.
- Generelles Ziel ist, das wir bis ende Januar 2018 die Funktionalität auf Frontend/Backend umgesetzt haben und uns anschliessend nur noch um das Design kümmern müssen.

Entscheide vom 23.12

- Model Klassen werden in Interface verschoben
- Angular 5 Material wird als GUI Bibliothek verwendet.
- Error sollen mit Error Popup in Komponente gelöst werden.
- Das Rest Response Observable wird vorläufig bis zur Component zurückgereicht. Ist im Moment noch einfacher.
  Alternative wäre das Response Observable z. B. Im Business Service auszuwerten, dann müsste aber gegebenenfalls ein neues erzeugt und zur Component gereicht werden. Dass würde dann in etwa so aussehen:

```
  register(request: IRegisterRequest): Observable<any> {
  const resourceTemp = this.resourceService;
  return Observable.create(function (observer) {
    resourceTemp.register(request).subscribe(response => {
      // mach die Hausaufgaben
      // und mach danach ein neues nettes Obsi für die Component
      observer.next({id: 'Halleluja!!', message: 'geht auch so'});
    });
  });
}
```

-
-