Ideen für die Projekt-Präsentation

möglicher Spin

“Herausforderungen” chronologisch:

- Zuerst mit sich selbst (Framework lernen, Aufgleisung)
- Danach innerhalb des Teams (Im Team einen Sprint fertig bekommen)
- Danach zwischen den Teams (Integration)
- Inzwischen zwischen Teams und Kunden (Kein Feedback)

Vorgehen

KEIN BDUF

- Wir haben **nicht** zuerst ein Datenmodell gebaut und das dann umgesetzt.
- Unsere Arbeitsweise war das Gegenteil, unterstützt durch Tools wie Hibernate, durch die Änderungen einfach umsetzbar sind.

Das Framework (aus-)nutzen

Das Rad nicht neu erfinden, lieber radeln lernen.

- Unser Projekt ist in der Aufgabenstellung ziemlich *straight forward*; (CRUD mit ein bisschen Businesslogik).
- Spring Boot...
  - erledigt eigentlich alle *Standardprobleme*, erfordert aber Einarbeitung.
  - Hat eine klare Vorstellung, wie gute Software aussehen sollte
    - Unit Tests
    - ORM
    - Zentrale Exceptionhandler
  - erlaubt es nicht, direkt loszuprogrammieren, aber man programmiert auch nicht Scheiße.
  - tauscht den Schmerz des Refactorings gegen den der Einarbeitung.
    - Hier tut's eher *am Anfang* weh als *am Ende*.
    - Ich les lieber manchmal spärliche Doku als Spaghetti-Code.

ScrumBut

Sinngemäß nach scrum.org:

Man sollte sich an's Framework halten, damit Probleme ans Tageslicht kommen. Bei ScrumBut kommt ein Problem zutage, ist aber nicht lösbar. Deshalb wird Scrum angepasst, so dass das Problem nicht gelöst, sondern nur unsichtbar wird.

ScrumBut

ScrumBut	Reason	Workaround
Wir machen Scrum, ABER	wir arbeiten nicht täglich	deshalb sind die Dailys nicht *daily*.
Wir machen Scrum, ABER	keiner Interessiert sich für unsere Software	daher liefern wir nichts.
Wir machen Scrum, ABER	wir bekommen kein Feedback vom Kunden	daher entwickeln wir das, was unser PO sich ausdenkt.

Der Github Workflow

- PR's all the way
- Wir sind vielleicht eher ein *verteiltes* Team als ein *Scrum*-Team.

Agil? Scrum?

Oder vielleicht eher:

- Kein Wasserfall sondern inkrementell
- (fast) keine Silos
  - 2 Stück, aber gekoppelt über die API-Spec
  - Wachstumsschmerzen: das *"Backend im Frontend"*
- breite Aktivierung durch gemeinsame Planung
  - Backend: Code von ~7
  - Frontend: Code von ~8

Verwendete Technologien

Spring Boot

ORM	Hibernate + Spring Data JPA
Webserver	(embedded) Tomcat

Teamstruktur (Mitglieder, Rollen, Zusammenarbeit)

- Statistiken ausm git

Journey (Entwicklung über Zeit, Veränderungen, Experimente)

Semester 1

- Tutorial: *Building a RESTful Web Service (in 15 min.)*
- Spring Boot Referenz
- Addy: Server

Semester 2

- Weiteres Aufgleisen
- ein, zwei Sprints vernünftig entwicklen

Challenges

- Security
  - Hohe Komplexität in der Umsetzung
- JPA / Hibernate
  - Man läuft zwar mal in ihre Faust, aber sie liebt einen wirklich.

Highlights

Fuckups

- Simon's PR die nie Umgesetzt wurde
- Member Kontaktdetails

Learnings (was nehmt Ihr für euch aus dem Projekt mit? Was bedeutet „agile“ für euch?)

- First and foremost: Wir haben ’n vernünftiges Softwareprojekt gemacht.