TripPlanner

Semesterarbeit NDS HF Applikationsentwicklung

Reto Kaufmann, Dieter Biedermann

Inhalt

- Projektidee
- Projektvorgehen
- Demo
- Technische Aspekte
- Reflexion

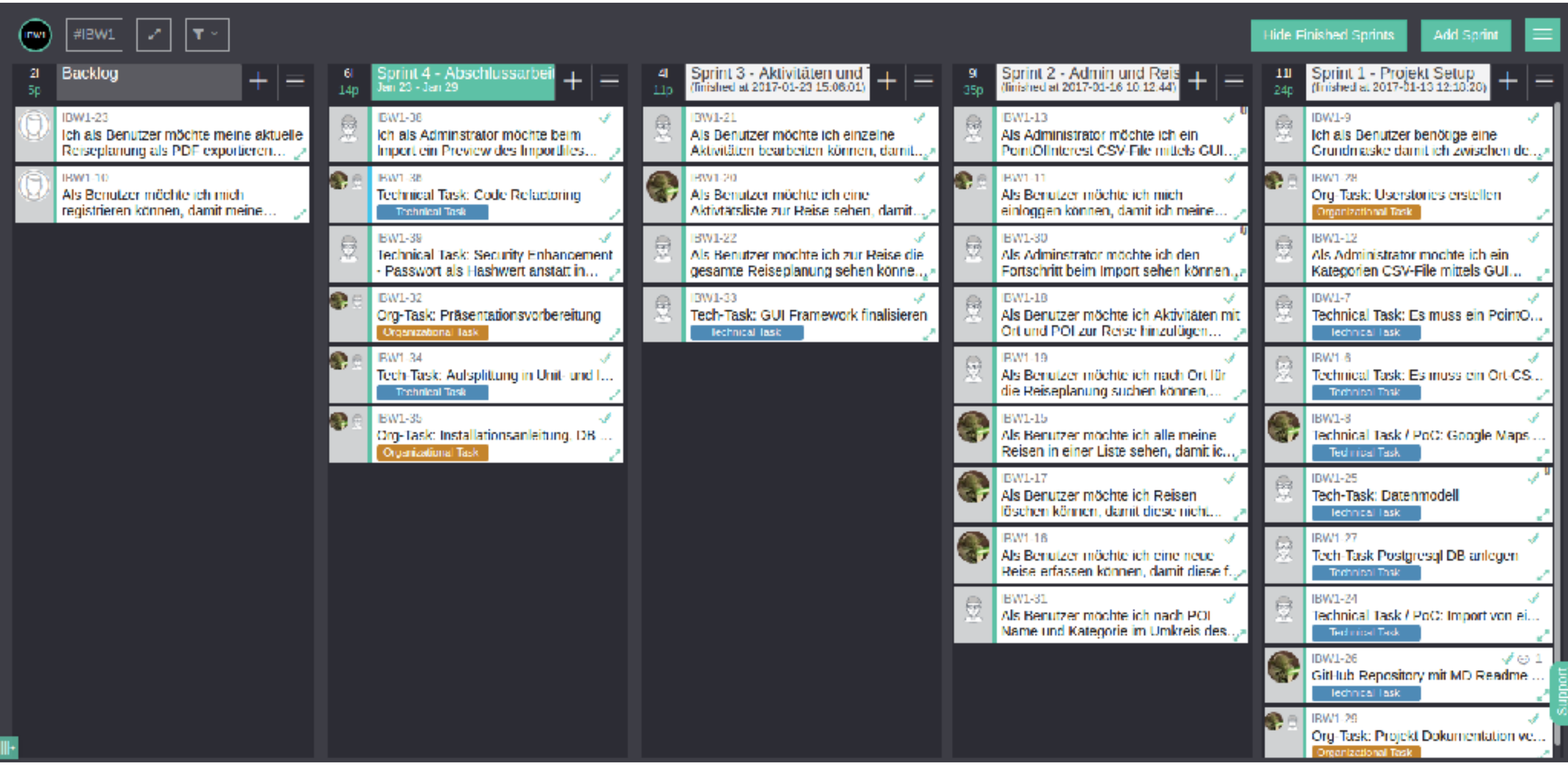
Projektidee

- Alle Lernthemen sollten angewandt werden
- Applikation zur Planung einer Reise
- Admin Funktionen: Datei Upload / DB Import
- Benutzer Funktionen: Reiseverwaltung, Poi Suche, Google Maps anzeige

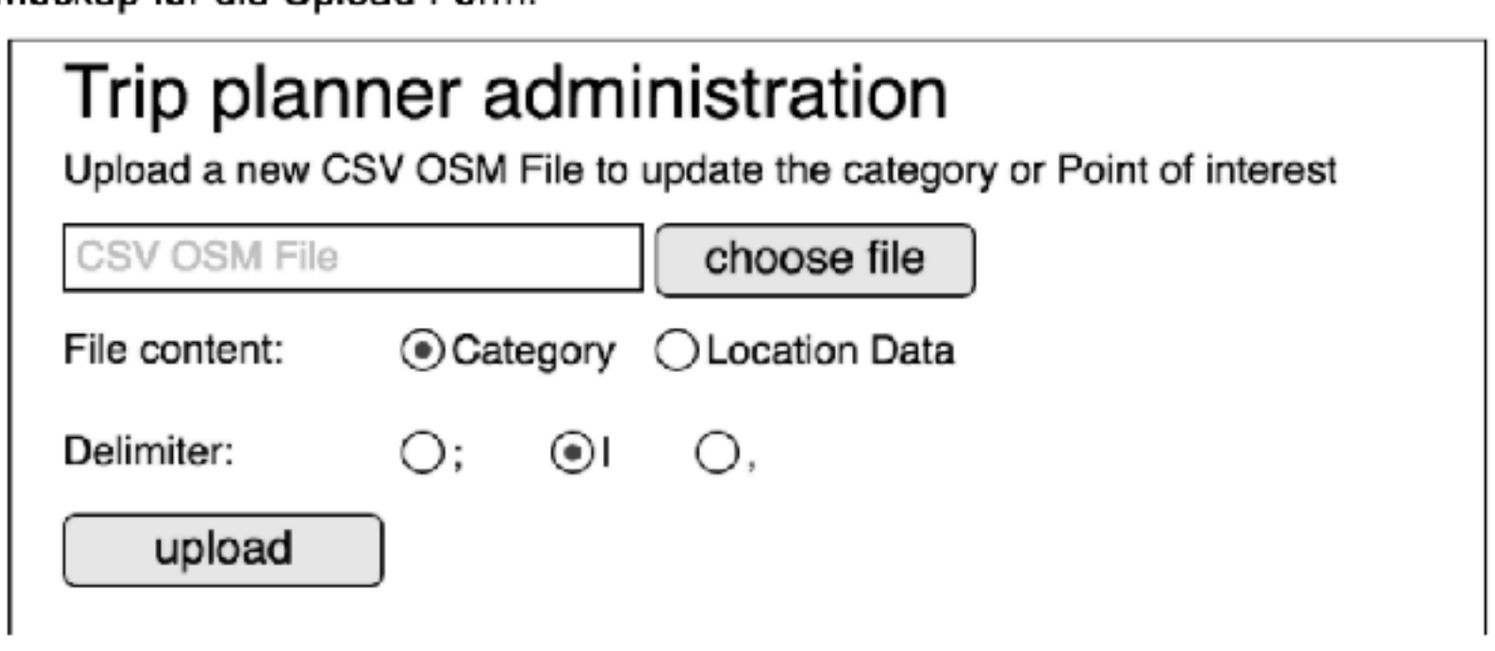
Projektvorgehen

- Scrum, User Stories
- Git, Feature-Branches, Pull-Request
- Komponentendiagramm
- Klassendiagramm

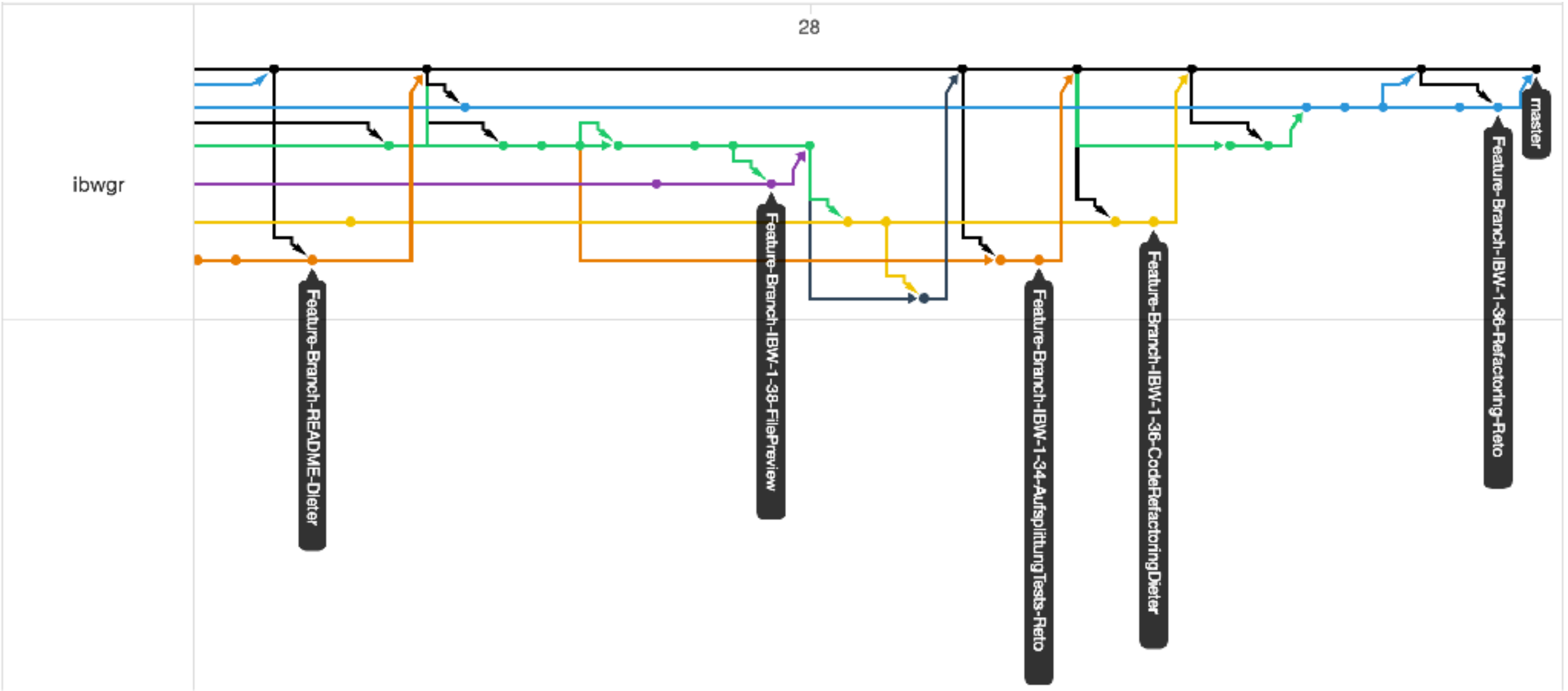
Scrum, User Stories



Beispiel einer User Story

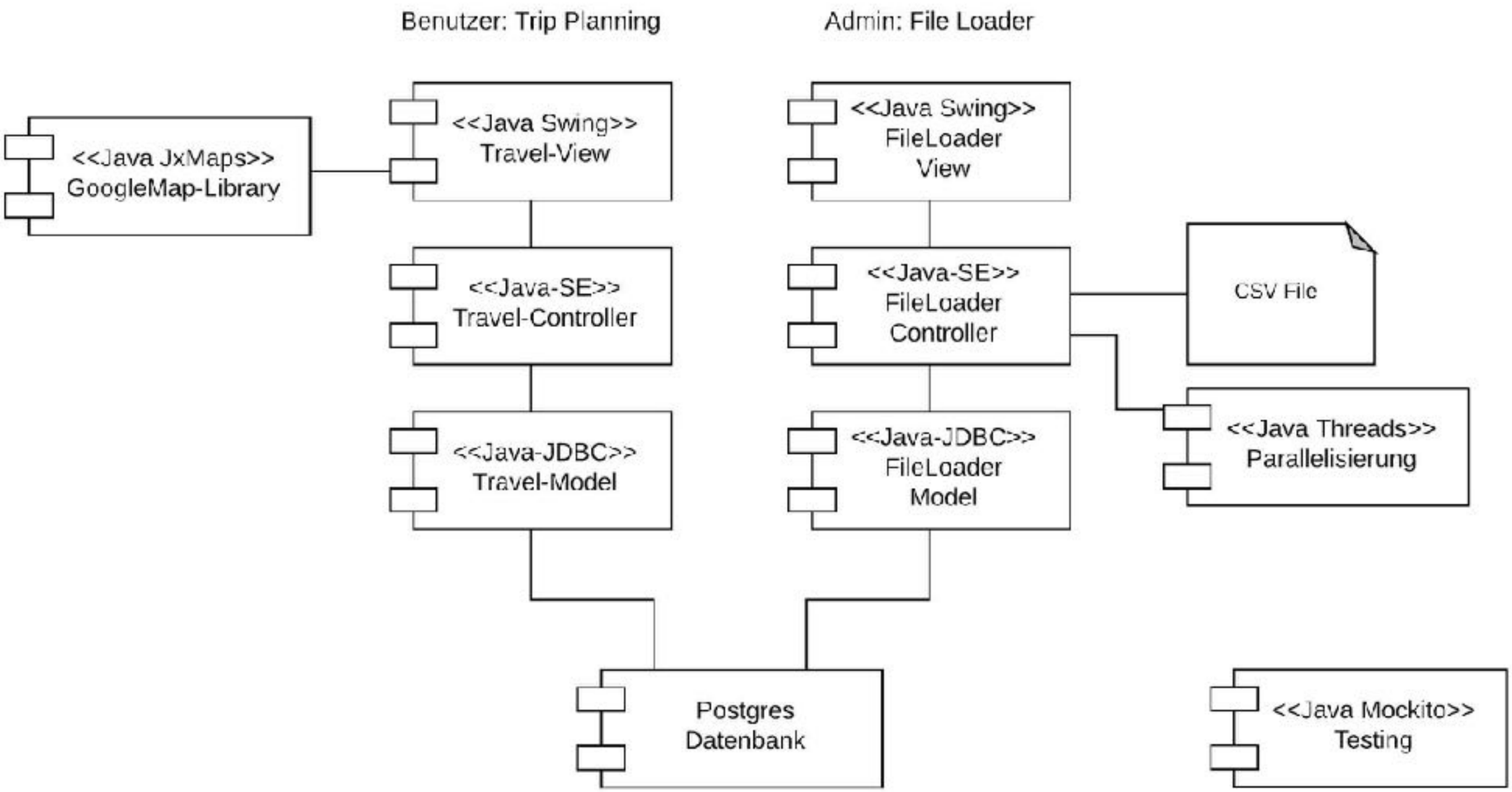
ID	Titel	Story Points
	Beschreibung	
	Akzeptanzkriterien	
IBW1-12	Als Administrator möchte ich ein Kategorien CSV-File mittels GUI hochladen können, damit die Point of Interest (POI) Kategorien importiert werden.	5
	Es braucht ein GUI, in welchem das CSV File ausgewählt und das CSV File importiert werden kann. Der Import soll neue Zeilen in der Tabelle CATEGORY einfügen und bereits vorhandene Zeilen mutieren. Für den Upload braucht es eine File Auswahl, Auswahl des File Typs (Category oder Location Data) und die Auswahl des Trennzeichens. Mockup für die Upload Form: 	
	* Auswahl eines CSV Files in einem "File öffnen"-Dialog * Alle Zeilen des Files werden in die Tabelle CATEGORY geschrieben.	

Git, Feature-Branches, Pull-Request



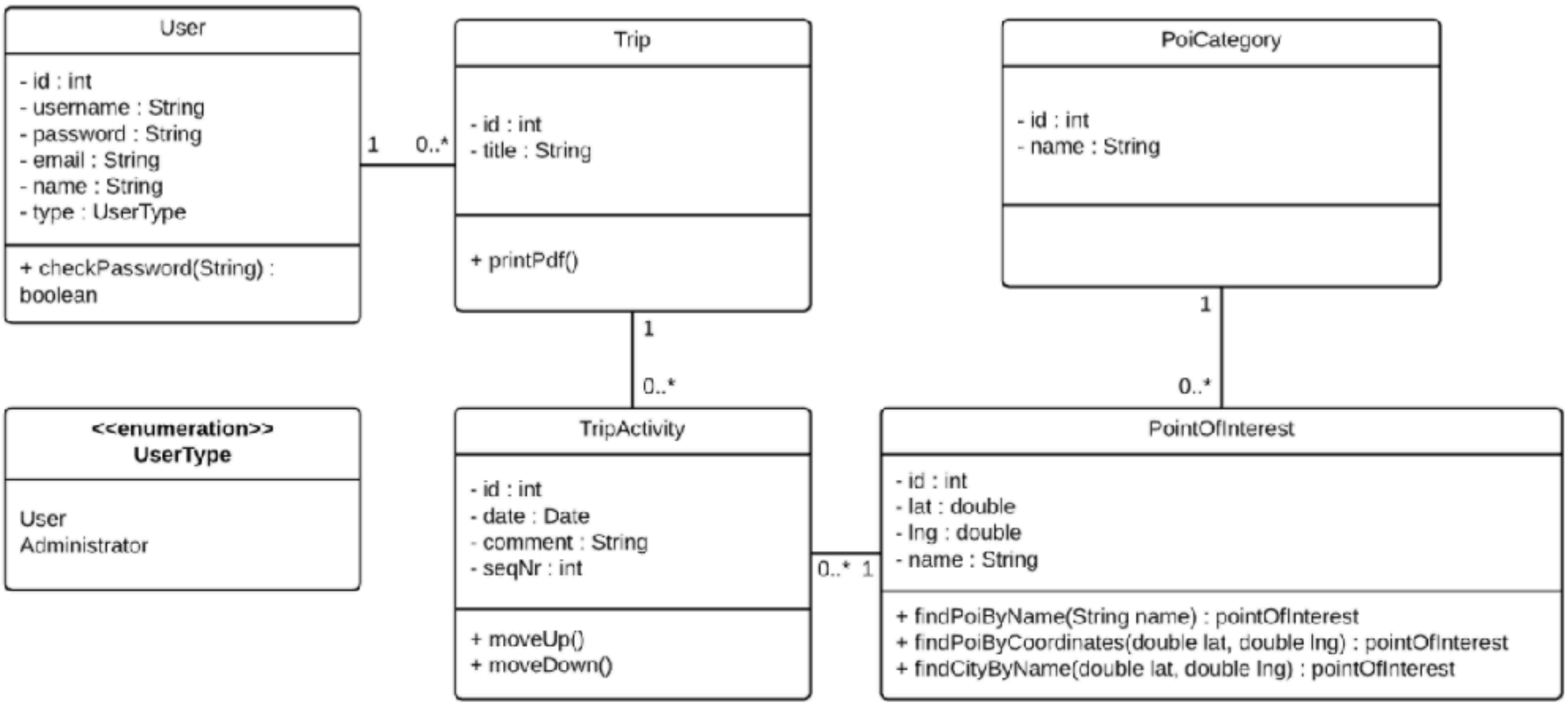
19 Branches, 84 Pull Requests in den Master
Ein Featurebranch jeweils pro Scrum Userstory

Komponenten

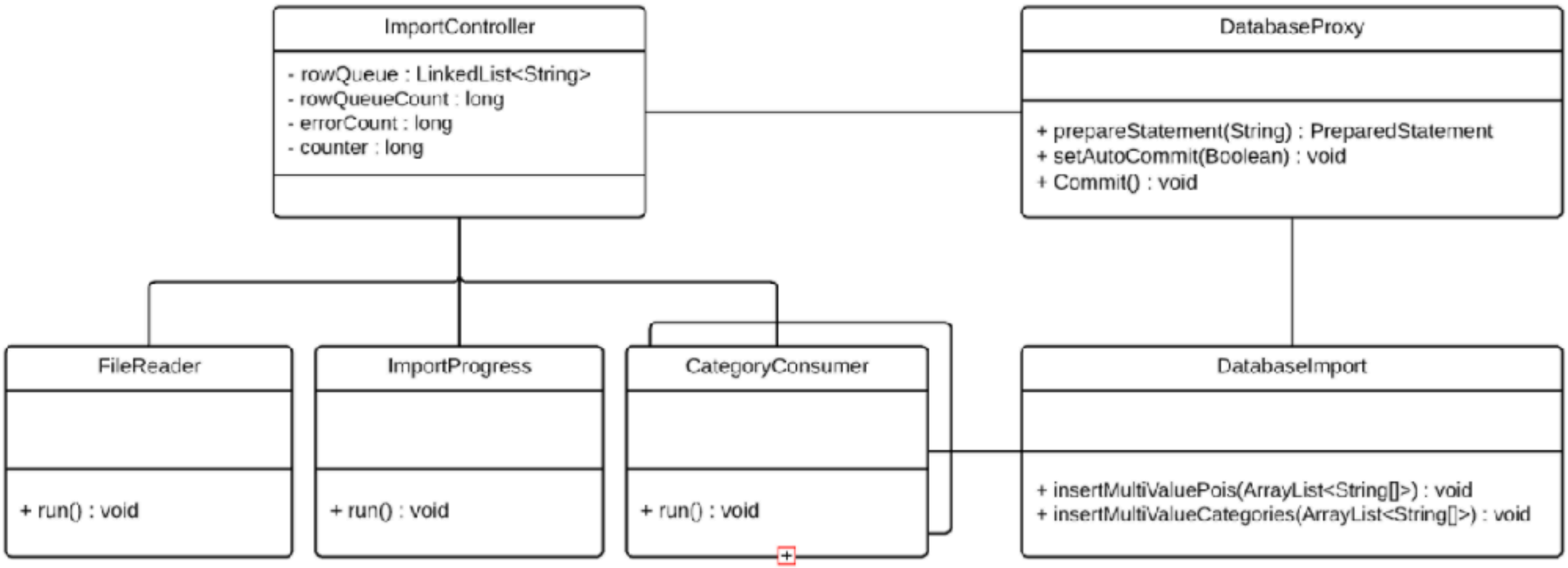


Klassen

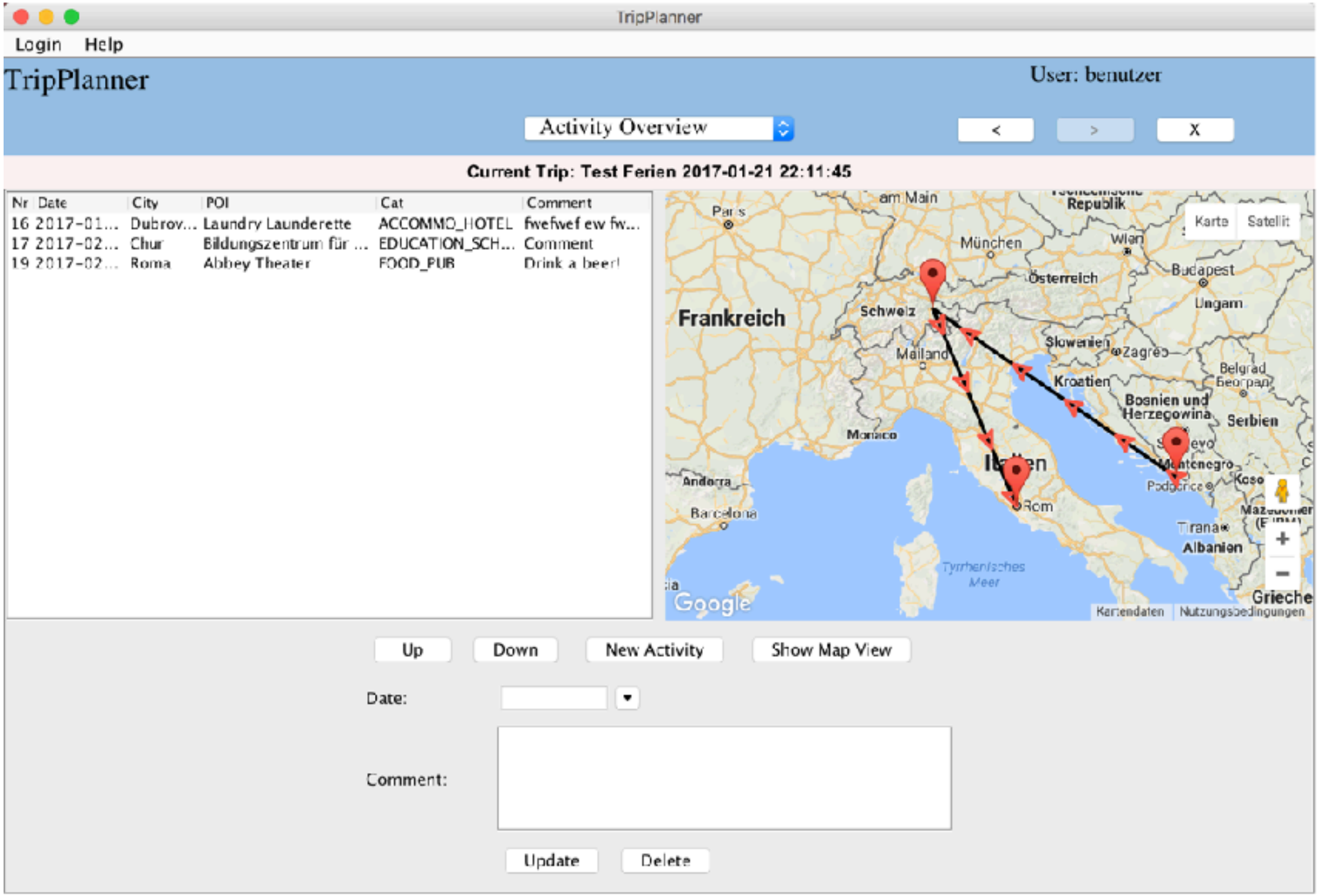
Travel:



Admin:


ImportController
- rowQueue : LinkedList<String>
- rowQueueCount : long
- errorCount : long
- counter : long
DatabaseProxy
+ prepareStatement(String) : PreparedStatement
+ setAutoCommit(Boolean) : void
+ Commit() : void
FileReader
+ run() : void
ImportProgress
+ run() : void
CategoryConsumer
+ run() : void
DatabaseImport
+ insertMultiValuePois(ArrayList<String[]>) : void
+ insertMultiValueCategories(ArrayList<String[]>) : void

Demo


TripPlanner
Login
Help
TripPlanner
User: benutzer
Activity Overview
<
>
X
Current Trip: Test Ferien 2017-01-21 22:11:45
Nr Date City POI Cat Comment
16 2017-01... Dubrov... Laundry Launderette ACCOMMO_HOTEL fwefwef ew fw...
17 2017-02... Chur Bildungszentrum für ... EDUCATION_SCH... Comment
19 2017-02... Roma Abbey Theater FOOD_PUB Drink a beer!
Karte
Satellit
Paris
München
Wien
Budapest
Österreich
Ungarn
Frankreich
Schweiz
Mailand
Slowenien
Zagreb
Kroatien
Belgrad
Bosnien und Herzegowina
Serbien
Monaco
Andorra
Barcelona
Rom
Tirana
Albanien
Tyrrhenisches Meer
Google
Kartendaten
Nutzungsbedingungen
Up
Down
New Activity
Show Map View
Date:
Comment:
Update
Delete

Technische Aspekte

- Producer / Consumer Pattern
- kleines Swing Framework
- JxMaps
- MVC
- Unit- / Integrations-Test

Producer / Consumer Pattern

- Producer: FileReader liest File und füllt Queue
- Consumer: CategoryConsumer / PoiConsumer lesen die Queue und schreiben die Daten mit Hilfe der DatabaseImport Klasse in die Datenbank

```
for (int i = 0; i < threadNo; i++) {
    DatabaseImport databaseImport = new DatabaseImport( importController: this, databaseProxy);
    consumers[i] = new PoiConsumer( importController: this, databaseImport, adminView.getFileDelimiter());
    consumers[i].start();
}
```

- DatabaseImport verwendet Multi Value Insert Statements, welche dynamisch erstellt werden.
- Für das Update wird ein Trigger verwendet, damit im Programm keine zusätzliche Abfrage auf die Datenbank notwendig ist und beim Schreiben eine Unterscheidung zwischen Update/Insert gemacht werden muss. (Geschwindigkeit Optimierung)

kleines Swing Framework

- MainTripPlanner Klasse: Ein kleines Swing Framework, damit GUIs schneller und einfacher entwickelt werden können.
- ViewInfo Klasse speichert die Informationen/Optionen zu einer View (Content) ab.
- Mit der FormPanel Klasse haben wir ein einheitliches Design für Formulare bereitgestellt. Diese Klasse enthält auch Hilfsmethoden zum schnellen erstellen von Buttons, RadioButtons, TextFields, etc.
- Beispiel Button:

```
addComponentToPanel(createButton("Open file", "open_file",
adminController));

addPanelWithLabel("File:", true);
```

- Beispiel RadioButton:

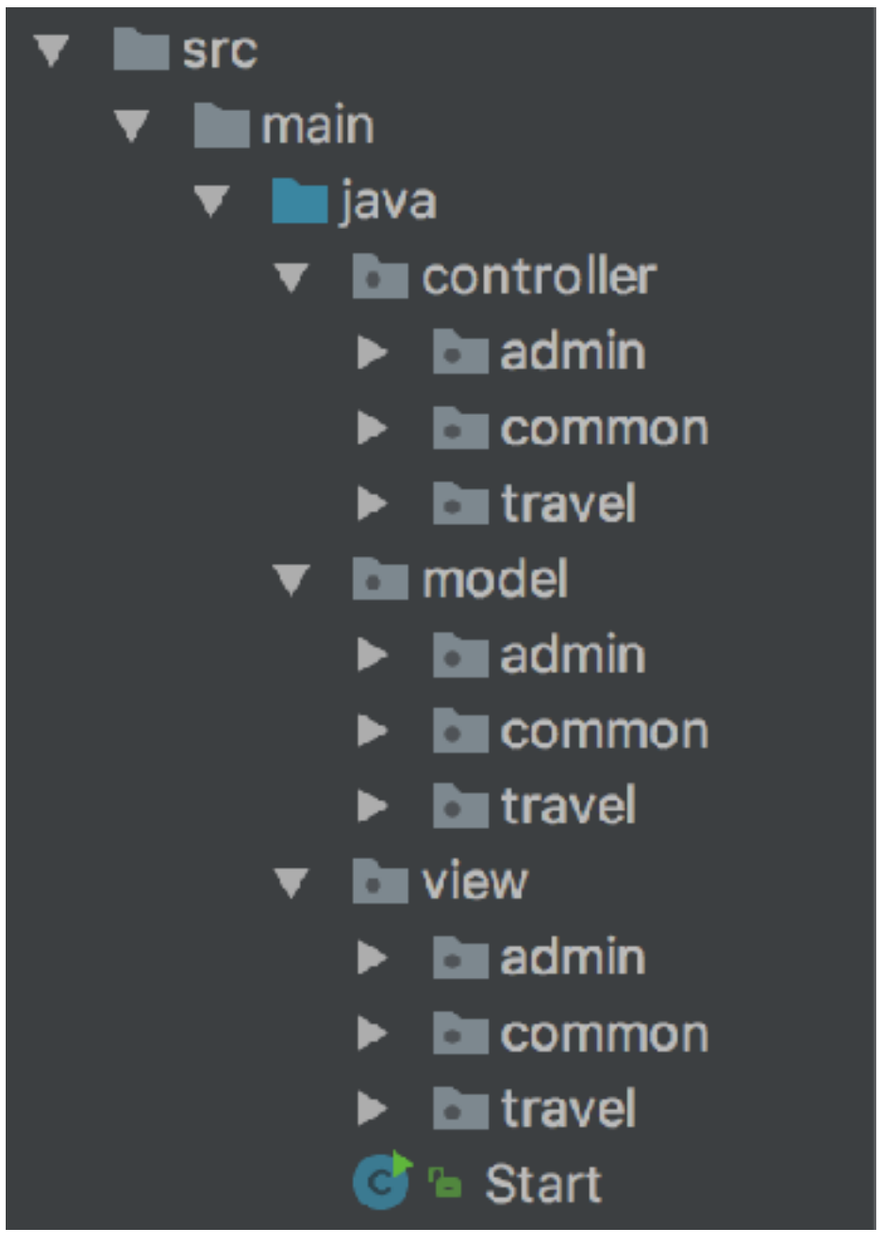
```
addComponentToPanel(fileTypeCategory = createRadioButton("Category",
"category", true, fileTypeGroup));

addPanelWithLabel("Type:", true);
```

JxMaps

```
LatLng[] path = new LatLng[activityList.size()];
int i = -1;
for (Activity activity : activityList) {
    System.out.println("Reihenfolge " +(i+1) +" --> "+activity.getCity());
    // Marker erstellen
    Marker marker = new Marker(map);
    LatLng latLng = new LatLng(activity.getPoi().getLatitudeDouble(), activity.getPoi().getLongitudeDouble());
    marker.setPosition(latLng);
    marker.setTitle(activity.getPoi().getName());
    markerList.add(new Pair<>(marker, activity));
    map.setCenter(latLng);
    // Adding event listener that intercepts clicking on marker
    marker.addEventListener( eventId: "click", (MapMouseEvent) (mouseEvent) -> {
            closeAllWindows();
            activityView.setActivityInList(activity);
            setWindow(activity);
    });
    path[++i] = latLng;
}
```

MVC


src
main
java
controller
admin
common
travel
model
admin
common
travel
view
admin
common
travel
Start

Unit- / Integrations-Test

- Innerhalb IntelliJ werden mittels “Run all tests” alle Unit Tests und alle Integrationstests gestartet. Hierbei muss somit die DB gestartet sein.

- Laufen die Tests unter Maven, werden bei “mvn test” explizit nur die Unit-Tests durch laufen.

- Travis startet beim Push und nach dem Merge (Pull Request in Master) auch die Unit-Tests via Maven.

- Testing unterscheidet UnitTest (@Category({ UnitTest.class })) und Integrationstest

- Mockito Controller- und View-Test:

  https://github.com/ibwgr/TripPlanner/blob/master/src/test/java/controller/common/LoginControllerTest.java#L65

  https://github.com/ibwgr/TripPlanner/blob/master/src/test/java/model/admin/FileReaderTest.java#L27

Reflexion

- MVC Struktur besser zuerst nach Thema gruppiert
- Immer direkt Maven Projekt machen (Standard Ordner Struktur)
- Viele freie Werkzeuge (Google Docs, Vivify Scrum, Lucidcharts, GitHub, Travis) ideal für Collaboration
- Etwas über Lizenzen (Open Source)

Reflexion - Statistik

Git "Code-Zeilen" Statistik per 27.01.2017:

Language	files	blank	comment	code
Java	60	1189	941	5035
SQL	1	53	102	173
Maven	1	5	12	111
Markdown	1	13	0	60
XML	1	0	0	6
YAML	1	1	4	6
SUM:	73	1261	1059	5391

Test Coverage per 27.01.2017:

Package	Class, %	Method, %	Line, %
all classes	71.7% (38/ 53)	49.2% (174/ 354)	48.6% (1024/ 2107)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
<empty package name>	0% (0/ 1)	0% (0/ 2)	0% (0/ 3)
controller.admin	100% (3/ 3)	63.6% (14/ 22)	44.4% (56/ 126)
controller.common	100% (2/ 2)	67.9% (19/ 28)	62.2% (97/ 156)
controller.travel	66.7% (2/ 3)	22.6% (7/ 31)	17.3% (33/ 191)
model.admin	80% (4/ 5)	72.7% (8/ 11)	44.8% (69/ 154)
model.common	100% (10/ 10)	64.7% (55/ 85)	54.8% (218/ 398)
model.travel	100% (2/ 2)	48.8% (20/ 41)	50.8% (134/ 264)
view.admin	66.7% (2/ 3)	40% (6/ 15)	57.9% (44/ 76)
view.common	100% (6/ 6)	69.6% (32/ 46)	80.9% (225/ 278)
view.travel	38.9% (7/ 18)	17.8% (13/ 73)	32.1% (148/ 461)