

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-Branche, Pull-Request
- Komponentendiagramm
- Klassendiagramm

Scrum, User Stories

The screenshot displays a Jira Scrum board for a project named 'IBW1'. The board is organized into five columns representing different stages of the Scrum process:

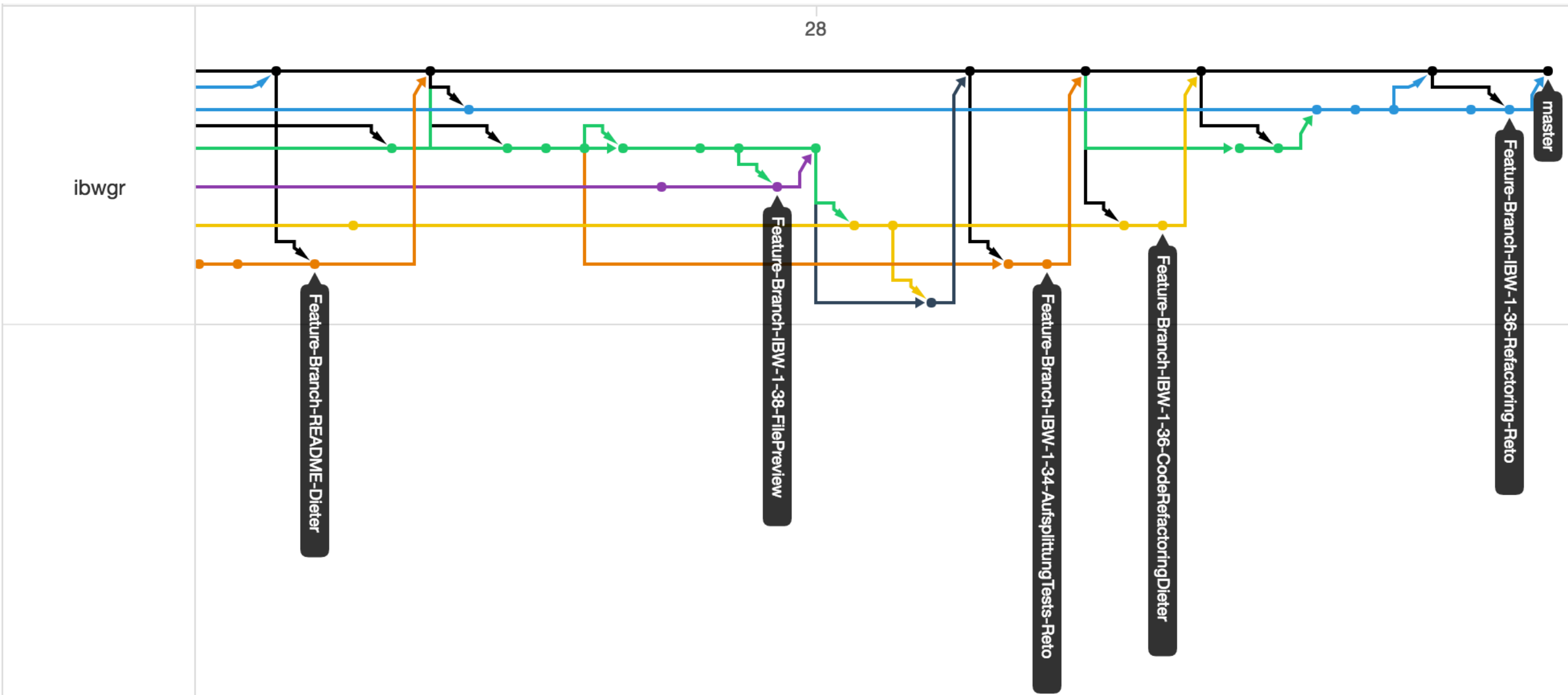
- Backlog (2 items, 5 points):**
 - IBW1-23: Ich als Benutzer möchte meine aktuelle Reiseplanung als PDF exportieren...
 - IBW1-10: Als Benutzer möchte ich mich registrieren können, damit meine...
- Sprint 4 - Abschlussarbeit (6 items, 14 points, Jan 23 - Jan 29):**
 - IBW1-38: Ich als Administrator möchte beim Import ein Preview des Importfiles...
 - IBW1-36: Technical Task: Code Refactoring
 - IBW1-39: Technical Task: Security Enhancement - Passwort als Hashwert anstatt in...
 - IBW1-32: Org-Task: Präsentationsvorbereitung
 - IBW1-34: Tech-Task: Aufsplittung in Unit- und I...
 - IBW1-35: Org-Task: Installationsanleitung, DB ...
- Sprint 3 - Aktivitäten und (4 items, 11 points, finished at 2017-01-23 15:06:01):**
 - IBW1-21: Als Benutzer möchte ich einzelne Aktivitäten bearbeiten können, damit...
 - IBW1-20: Als Benutzer möchte ich eine Aktivitätsliste zur Reise sehen, damit...
 - IBW1-22: Als Benutzer möchte ich zur Reise die gesamte Reiseplanung sehen könne...
 - IBW1-33: Tech-Task: GUI Framework finalisieren
- Sprint 2 - Admin und Reis (9 items, 35 points, finished at 2017-01-16 10:12:44):**
 - IBW1-13: Als Administrator möchte ich ein PointOfInterest CSV-File mittels GUI...
 - IBW1-11: Als Benutzer möchte ich mich einloggen können, damit ich meine...
 - IBW1-30: Als Administrator möchte ich den Fortschritt beim Import sehen können...
 - IBW1-18: Als Benutzer möchte ich Aktivitäten mit Ort und POI zur Reise hinzufügen...
 - IBW1-19: Als Benutzer möchte ich nach Ort für die Reiseplanung suchen können...
 - IBW1-15: Als Benutzer möchte ich alle meine Reisen in einer Liste sehen, damit ic...
 - IBW1-17: Als Benutzer möchte ich Reisen löschen können, damit diese nicht...
 - IBW1-16: Als Benutzer möchte ich eine neue Reise erfassen können, damit diese f...
 - IBW1-31: Als Benutzer möchte ich nach POI Name und Kategorie im Umkreis des...
- Sprint 1 - Projekt Setup (11 items, 24 points, finished at 2017-01-13 12:18:28):**
 - IBW1-9: Ich als Benutzer benötige eine Grundmaske damit ich zwischen de...
 - IBW1-28: Org-Task: Userstories erstellen
 - IBW1-12: Als Administrator möchte ich ein Kategorien CSV-File mittels GUI...
 - IBW1-7: Technical Task: Es muss ein PointO...
 - IBW1-6: Technical Task: Es muss ein Ort-CS...
 - IBW1-8: Technical Task / PoC: Google Maps ...
 - IBW1-25: Tech-Task: Datenmodell
 - IBW1-27: Tech-Task Postgresql DB anlegen
 - IBW1-24: Technical Task / PoC: Import von ei...
 - IBW1-26: GitHub Repository mit MD Readme ...
 - IBW1-29: Org-Task: Projekt Dokumentation ve...

Each item includes a user story icon, a title, a description, and a progress indicator (checkmark and arrow). Tasks are categorized as 'Technical Task' or 'Organizational Task'.

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
	<p>Es braucht ein GUI, in welchem das CSV File ausgewählt und das CSV File <u>importiert</u> 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 <u>Typs</u> (Category oder Location Data) und die Auswahl des Trennzeichens.</p> <p>Mockup für die Upload Form:</p> <div data-bbox="263 1101 1822 1733" data-label="Form"> <p>Trip planner administration</p> <p>Upload a new CSV OSM File to update the category or Point of interest</p> <div> <input type="text" value="CSV OSM File"/> <input type="button" value="choose file"/> </div> <p>File content: <input checked="" type="radio"/> Category <input type="radio"/> Location Data</p> <p>Delimiter: <input type="radio"/> ; <input checked="" type="radio"/> <input type="radio"/> ,</p> <div><input type="button" value="upload"/></div> </div>	
	<p>* Auswahl eines CSV Files in einem "File öffnen"-Dialog</p> <p>* Alle Zeilen des Files werden in die Tabelle CATEGORY geschrieben.</p>	

Git, Feature-Branches, Pull-Request

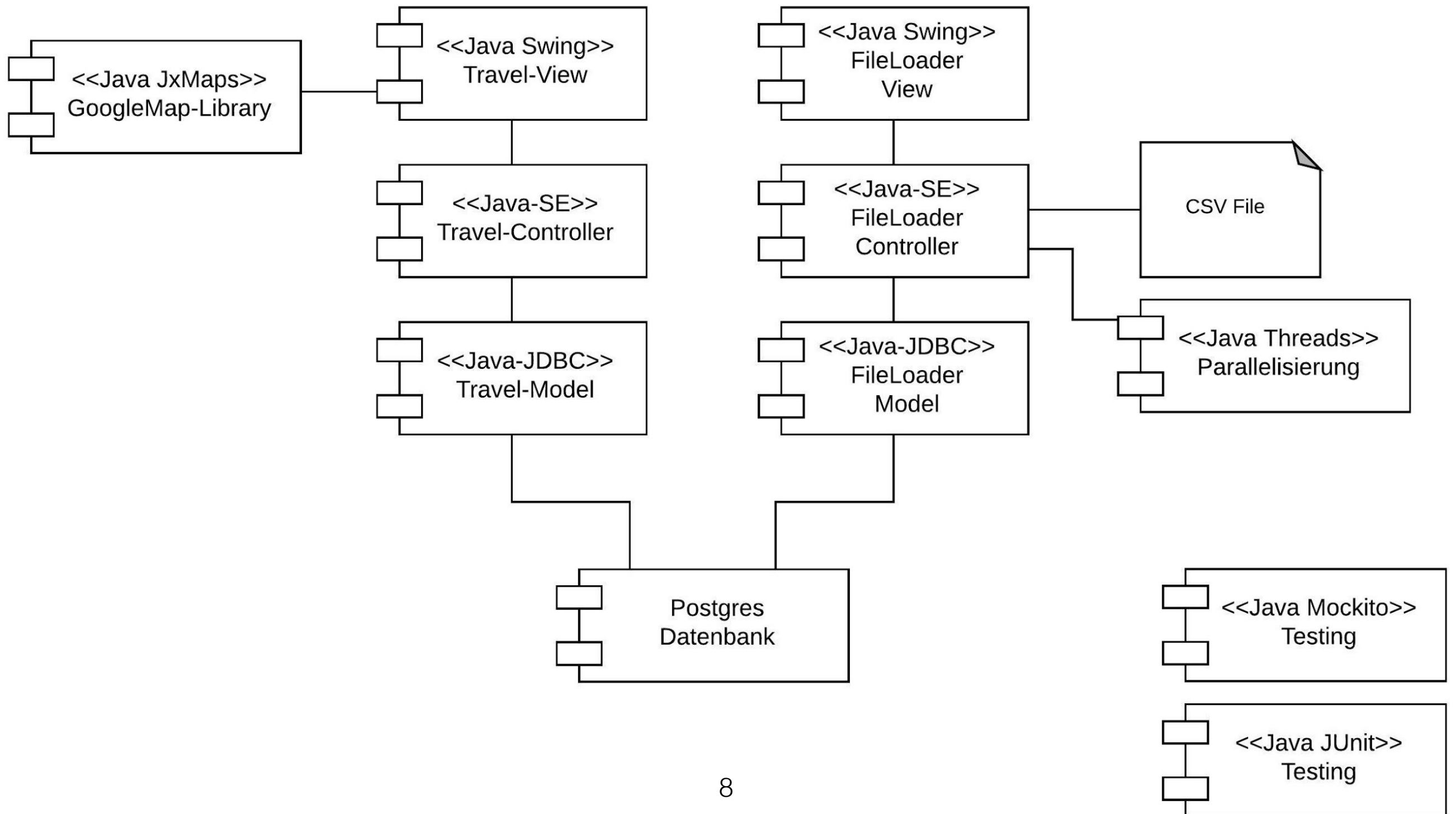


19 Branches, 84 Pull Requests in den Master
Ein Featurebranch jeweils pro Scrum Userstory

Komponenten

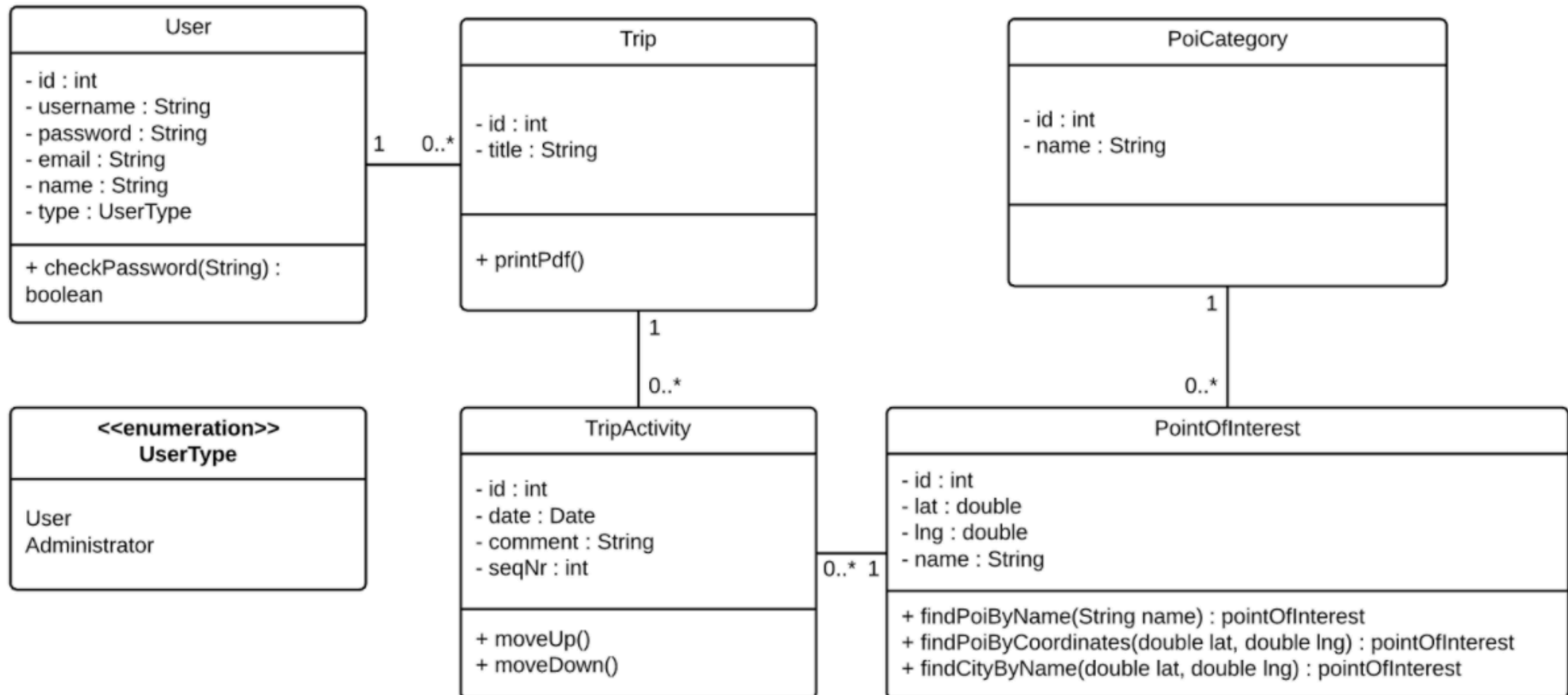
Benutzer: Trip Planning

Admin: File Loader

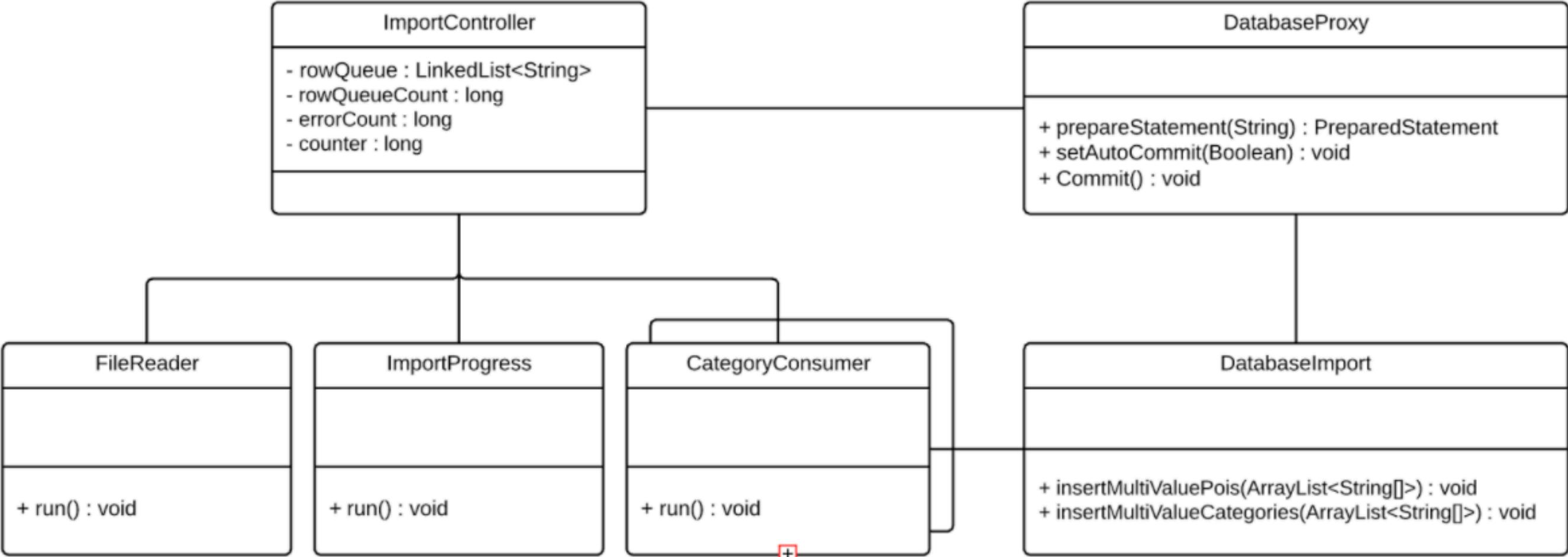


Klassen

Travel:



Admin:



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!

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();  
}
```

- Geschwindigkeit Optimierungen:
 - 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.

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:

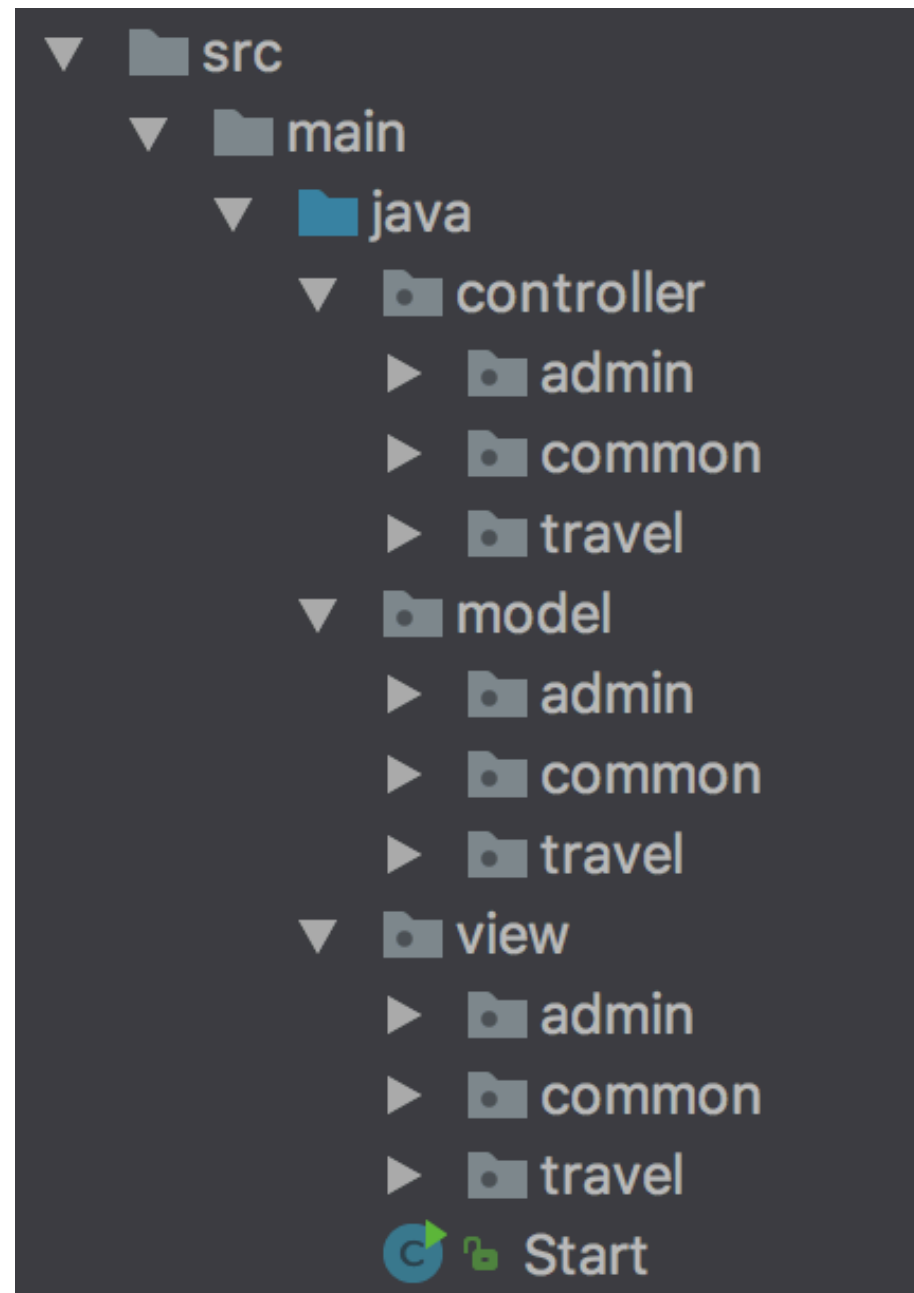
```
addComponentToPanel(fileTypeCategory = createRadioButton("Category",  
"category", true, fileTypeGroup));
```

```
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.addListener(eventId: "click", (MapMouseEvent) (mouseEvent) -> {
        closeAllWindows();
        activityView.setActivityInList(activity);
        setWindow(activity);
    });
    path[++i] = latLng;
}
```

MVC



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)
- Views schwierig zu testen

Reflexion - Statistik

Git "Code-Zeilen" Statistik per 27.01.2017:

Language	files	blank	comment	code
Java	68	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)