

# Programmierpraktikum

## 3. Semester Medizininformatik

Markus Degen, Dominique Brodbeck

Fachhochschule Nordwestschweiz

# Beyond Programmieren: Software Engineering

---

Anforderungsanalyse

SRS

Kollaboration

gitlab



**Programmieren**

Konstruktion

maven

Versionsverwaltung

git

Qualität

unit testing  
coding conventions

### *Die Studierenden ....*

- ▶ *... sind in der Lage aus einer unscharfen Projektbeschreibung Anforderungen abzuleiten, diese zu priorisieren und auf der Zeitachse zu planen*
- ▶ *... können in einem Team ein robustes und dokumentiertes Software-System entwickeln, welches die zuvor erarbeiteten Anforderungen erfüllt und nutzen dabei gängige Software-Tools zur Unterstützung des Software-Lebenszyklus*

## Methode: Durchführen eines Software-Projekts

---

- ▶ Durchführen eines Software-Entwicklungsprojekts in Teams mit allen Facetten:
  - Anforderungsanalyse
  - Einarbeiten in neue Technologien
  - Erstellen und Testen von Software
  - Arbeit / Abstimmung in Teams
  - Präsentation der (Zwischen-) Ergebnisse

## Weitere Projekte in der Medizininformatik

---

- ▶ 5. Semester: Praxisprojekt
  - Meist mit Industriepartner, im Team
- ▶ 6. Semester: Grösseres Projekt, führt Inhalte von mehreren Fächern zusammen
  - Interne Aufgabenstellung, im Team
- ▶ Bachelorthesis
  - Meist mit Industriepartner, Einzelarbeit

# The Story

# Was ist die Distanz zwischen...? Wie gross ist...?

---



Photoshop?  
kennt nur Pixel...

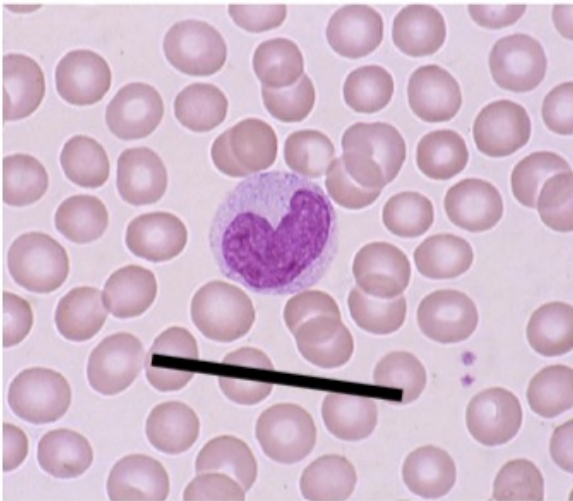
Excel?  
kennt keine Bilder...

Matlab?  
kostet etwas...

Google Maps?  
kennt nur die Erde...

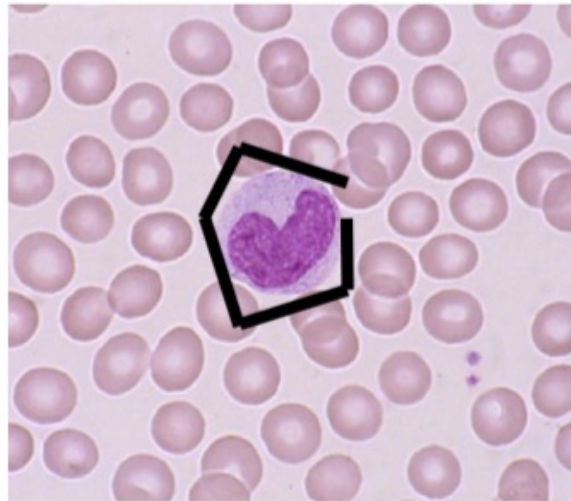
# Eine interaktive App mit GUI

Messung von Linien:



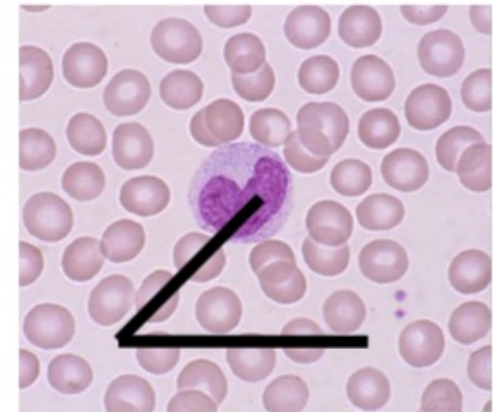
Length: 34mm

Messung von Pfaden:



Length: 98mm

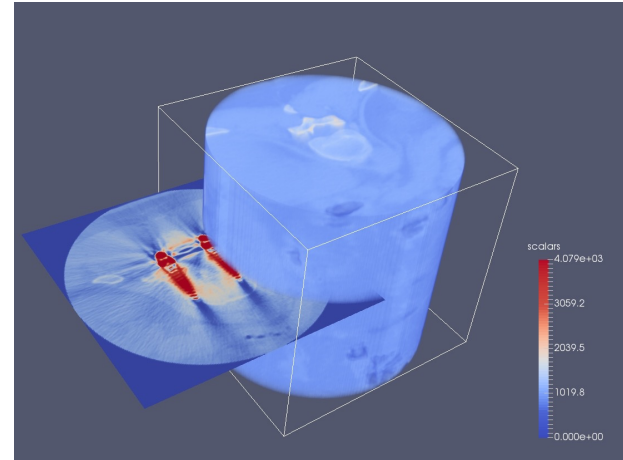
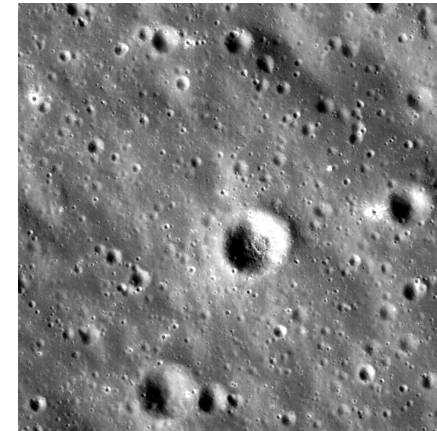
Messung von Winkeln:



Angle: 40°



# Typische Daten



# Linienfarbe



Farbe  
Dicke

“Weisser Adler auf weissem Grund”-Problematik



# Das Datenformat

## Meta-Datei

```
description: Blutausstrich (Mensch)  
image-file: image01.jpg  
resolution: 0.002 mm
```

resolution = Abstand zwischen  
zwei Bildpunkten

test-image-01.txt

## Bild-Datei

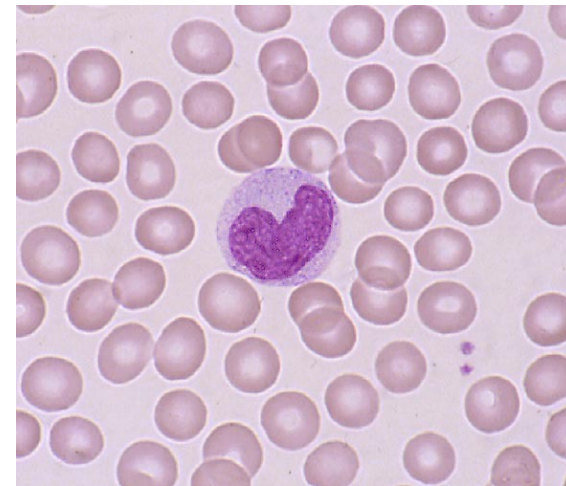
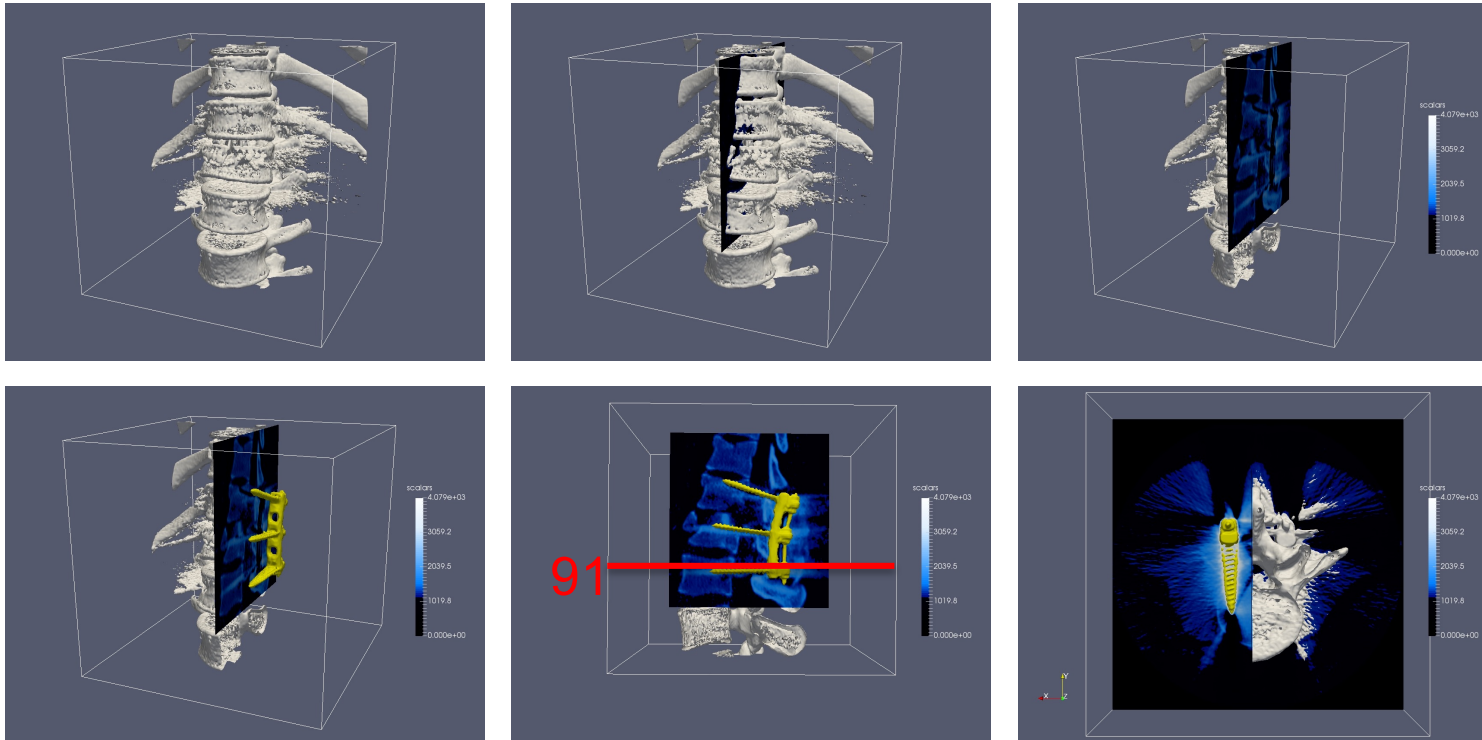
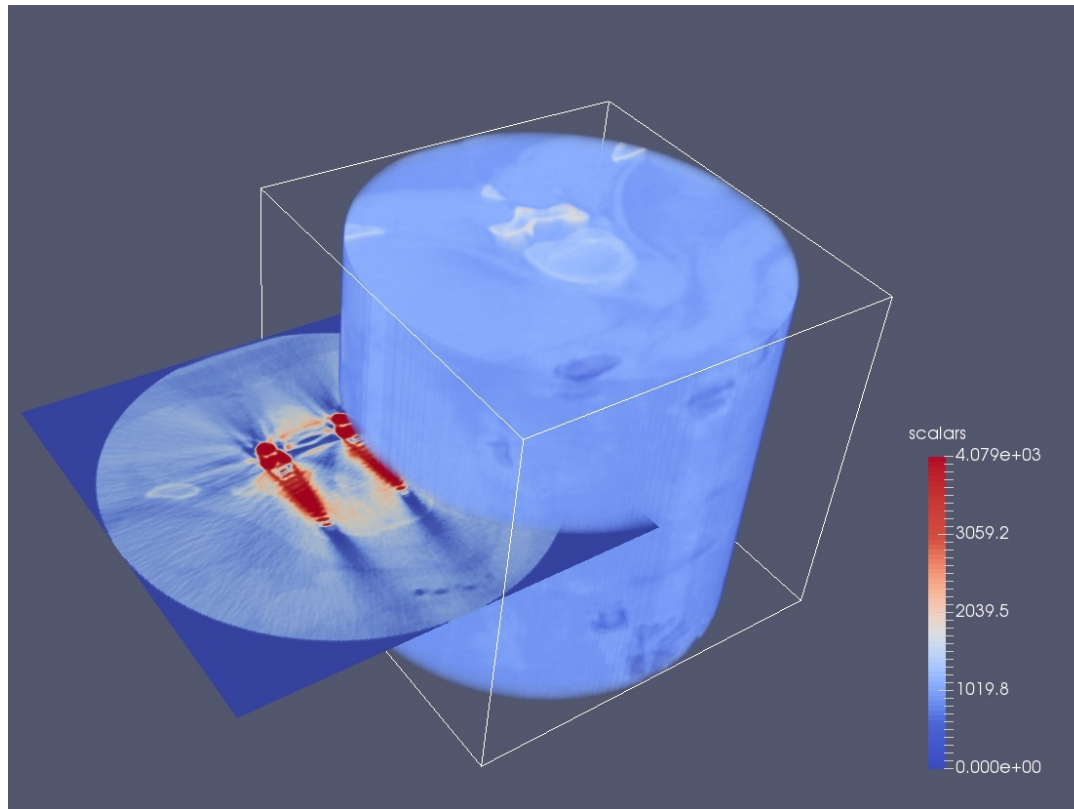


image01.jpg

# Länge der Schraube in Slice 91?

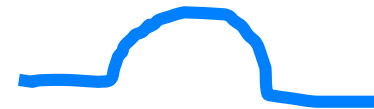
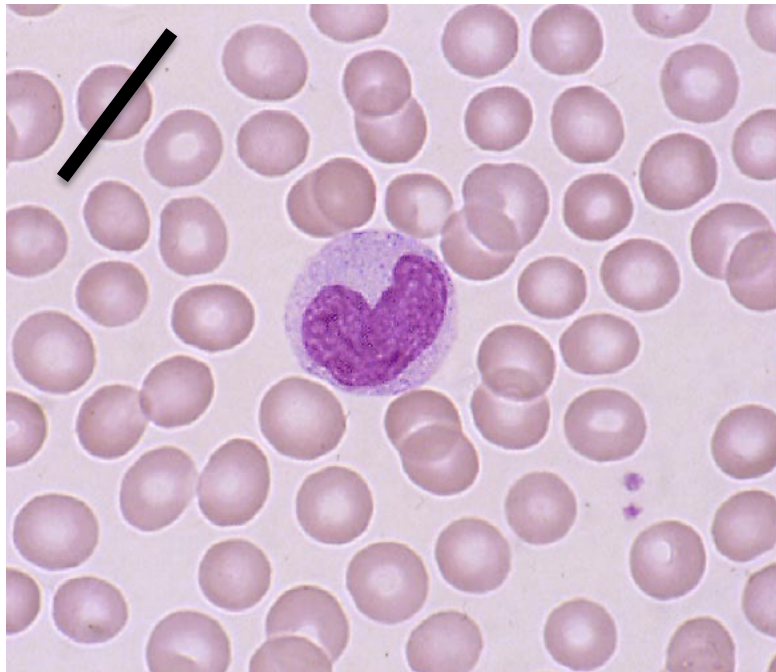


## Länge der Schraube in Slice 91?



Datei: wirbel.vtk, Auflösung lateral 0.58mm

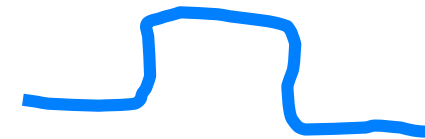
# Intensitätsprofile



?



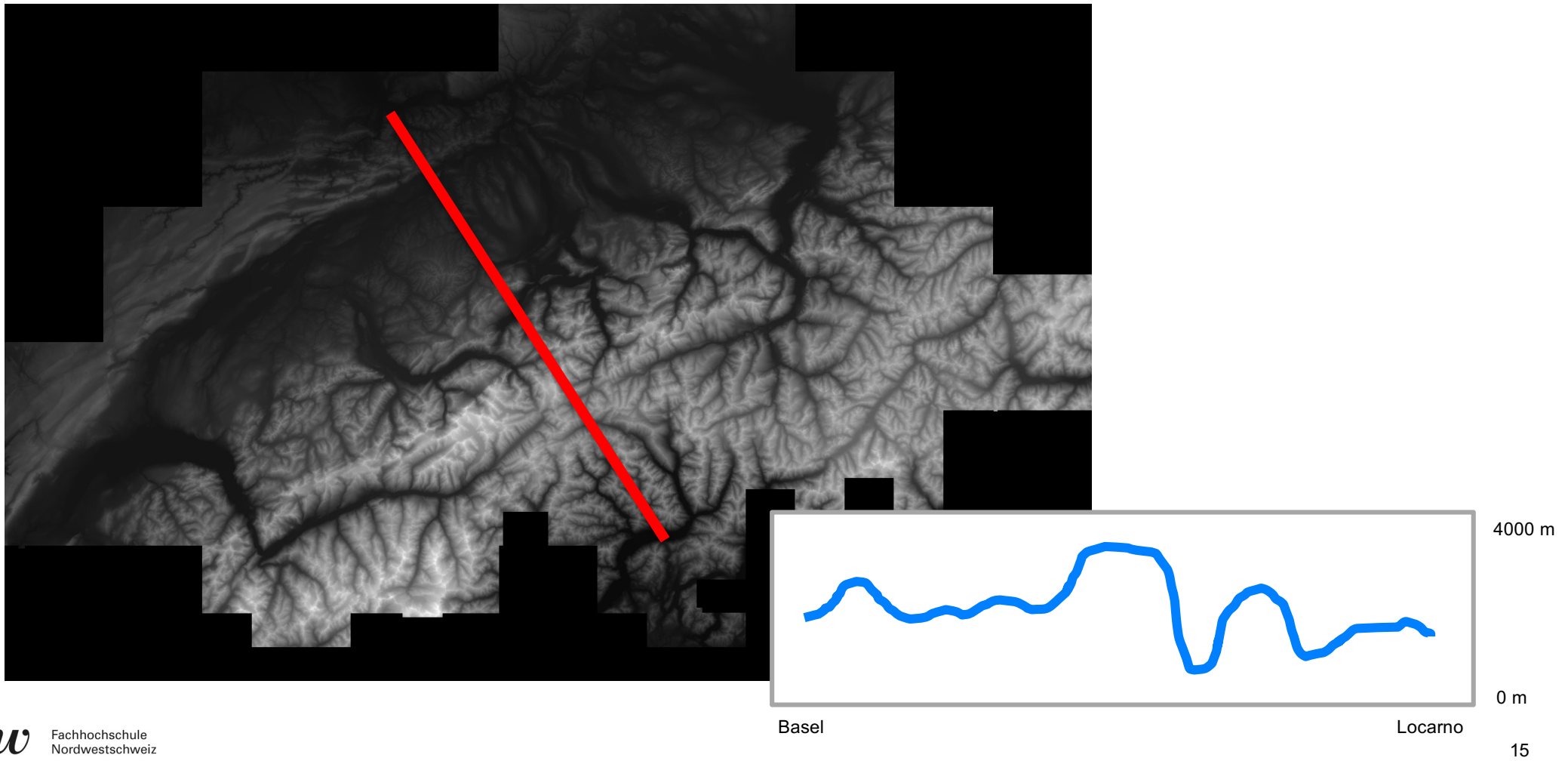
?



?

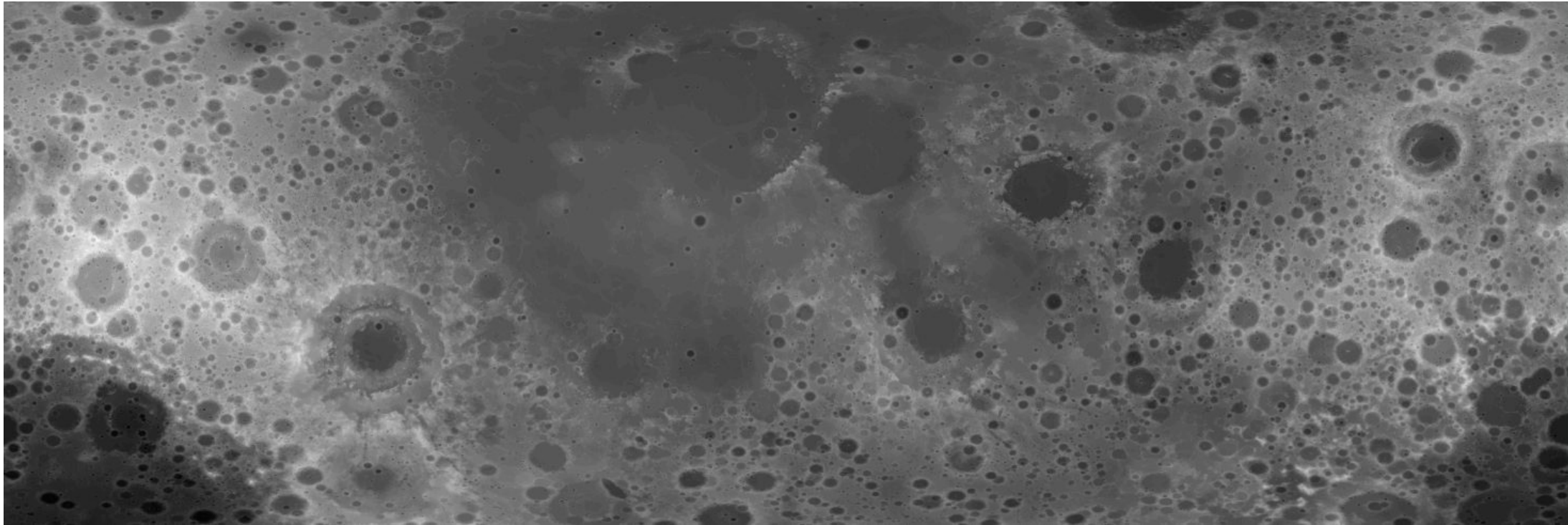


# Höhenprofil aus DEM (digital elevation model)



# The Moon - of course

---





# Was braucht es?

Wir brauchen ein GUI, mit dem man Bilder vermessen kann. Die Distanz zwischen zwei Punkten ist wichtig, aber auch der Umfang (z.B. einer Zelle) oder allgemein die Länge entlang einem Pfad (z.B. Länge der Wirbelsäule vom Sakrum bis zum Brustkorb). Manchmal sind auch noch Winkel praktisch, z.B. zwischen zwei Fingern oder Wirbelachsen.

Für die zusätzlichen Informationen (Metrik, Beschreibung, etc.) haben wir einfach zu jedem Bild noch eine Textdatei erstellt. Als Einheiten müssen mindestens mm, cm, m, und km unterstützt werden. Im GUI sollte dann immer ersichtlich sein, mit welchen Einheiten man es zu tun hat, also z.B. in der Anschrift der Seitenlängen. Die Beschreibung soll auch angezeigt werden.

Wir hatten einmal ein Tool, aber das hat die Bilder immer quadratisch angezeigt. Unsere Bilder können aber ganz verschiedene Seitenverhältnisse haben, manchmal im Breitformat aber auch oft eher Porträt, und die dürfen nicht verzerrt werden. Manchmal fehlt übrigens die Auflösung, z.B. bei normalen Fotos, dann können die Längenangaben ja aber einfach in Pixeln erfolgen.

Es gibt einige Testbilder. Die sind allerdings alle verschieden gross. Man müsste die einfach skaliert anzeigen, je nach dem, wie gross das Fenster der App ist. Die Testbilder zeigen ganz verschiedene Anwendungen, von Mikroskopiebildern bis zu Mondkratern. Auch in CT- und MRI-Bildern kann es nützlich sein, Messungen zu machen, dazu müssen diese aber zuerst ins richtige Format gebracht werden. Es kam auch einmal noch die Idee auf, entlang einer gemessenen Linie nicht nur die Distanz anzuzeigen, sondern auch das Profil der Bilddaten. Damit könnte man z.B. Höhenmodelle analysieren.

Der Praktikant hat kürzlich gemeint, das Dateiformat sei etwas altmodisch und hat eine Variante im JSON Format erstellt. Das sei flexibler bezüglich zusätzlichen Feldern, oder wenn einmal eines fehlt oder die Reihenfolge der Felder anders ist. Jetzt müssen halt beide Formate unterstützt werden. Auswählen tut man die Beschreibungsdatei und da drin steht neben den Bildinformationen, welche Bilddatei geladen werden muss. Der Praktikant vertritt auch die Entwicklungsleiterin, welche ferienhalber abwesend ist. Das User Interface muss mit JavaFX programmatisch (d.h. ohne visuelle Layout Tools wie Scene Builder) implementiert werden. Es sollen keine zusätzlichen Bibliotheken verwendet werden (ausser zum Lesen der JSON Daten, falls das das Java SDK nicht schon kann).

Abteilungsleiter Hugentobler möchte noch einen Nachtmodus, wie in seiner Lieblings-IDE.

Die Entwicklungsleiterin Sophie Meier besteht auf einer 3-Schichten Architektur, damit die Logik-Klassen auch unabhängig vom GUI wiederverwendet werden können. Sie hat auch bereits eine Anwendung dafür. Sie benötigt ein simples Tool, welches auf der Kommandozeile ausgeführt werden kann und alle in einem Ordner befindlichen Bilddateien auflistet, eine Zeile pro Bild mit Namen und den Dimensionen in physikalischen Einheiten.

# Aufgabenstellung

---

- ▶ „Entwicklung einer Applikation zur Vermessung von Bildern“
  - Erarbeiten, Abstimmen und Priorisieren von Anforderungen, ausgehend von «unscharfen» Beschreibungen
  - Planung, Umsetzung und Test der Applikation
  - Projektmanagement-Tools für Software-Entwicklung einsetzen:
    - Git (Source Code Repository, Dokumentation, Tasks/Issues)
    - Agiles Vorgehen (Planen von Iterationen, Führen eines Backlog, Daily Meetings)
    - Reproduzierbares Software Engineering: Verwenden von Software Management Tools (maven)
- ▶ **Wichtig:** Das Ziel ist nicht nur, am Schluss eine funktionierende Applikation zu haben, sondern den ganzen (professionellen) Entwicklungspfad einer Applikation durchlaufen zu haben

## Bewertung

---

- ▶ Die Schlussbewertung dieses Moduls erfolgt in Form eines Testats: «**bestanden**» (6 ECTS) / «**nicht bestanden**» (0 ECTS)
- ▶ Das Testat kann erreicht werden, in dem eine Menge von Aufgaben erfolgreich (Beurteilung durch Betreuende) abgeschlossen werden. Die dafür zur Verfügung stehende Zeit beträgt maximal 16 Tage (4 Wochen x 4 Tage). Der Zeitplan und damit die Geschwindigkeit kann aber gruppenindividuell bestimmt werden (d.h. das Praktikum kann auch in kürzerer Zeit absolviert werden)

## Kommunikation: In der Projektgruppe

---

- ▶ Angelehnt an „agile Vorgehensmodelle“ in der Software-Entwicklung
  - Häufige Abstimmungstreffen (min. täglich). Das Ziel ist, dass alle in der Gruppe immer genau wissen, wer gerade an was arbeitet. Planen Sie diese (virtuellen) Treffen schon zu Beginn des Projektes fix ein (Ein Treffen kann im Extremfall auch nur 1 Min dauern, falls nichts Neues vorliegt)
  - Seien Sie für einander «ansprechbar» durch den Einsatz eines Instant-Messengers (z.B. Slack, Teams, Skype)
  - Führen Sie in gitlab eine Taskliste (Issues) und aktualisieren Sie diese ständig
    - Ein „Task“ sollte nicht länger als ca. 2 - 4h (<1/2 Tag) Arbeitsaufwand entsprechen

## Kommunikation: Mit Betreuern

- ▶ Es gibt nur drei obligatorische Besprechungen (on-line) mit den Betreuern (Einladungen folgen):
  - Ende erste Woche: Requirements, GUI, Architektur (Als Gruppe)
  - Nach der 1. Iteration der Entwicklung
  - Ende des Projekts: 10' einzeln Code erklären (pro Person)
- ▶ Sie dürfen jedoch jederzeit auch um weitere Treffen anfragen

	Programmierpraktikum HS2022																			
	KW48					KW49					KW50					KW51				
	M	D	M	D	F	M	D	M	D	F	M	D	M	D	F	M	D	M	D	F
	28.11.22	29.11.22	30.11.22	01.12.22	02.12.22	05.12.22	06.12.22	07.12.22	08.12.22	09.12.22	12.12.22	13.12.22	14.12.22	15.12.22	16.12.22	19.12.22	20.12.22	21.12.22	22.12.22	23.12.22
Kickoff																				
Zwischentreffen																				
Schlusspräsentation																				

## Kommunikation: Mit Betreuern

- ▶ Wenn Sie glauben, einen Task erledigt zu haben, erstellen Sie in GitLab ein entsprechendes Ticket («Issue») und weisen dieses beiden Betreuern zu. Wenn das Ticket von uns «geschlossen» wird, heisst das, wir sind ebenfalls der Meinung, dass der Task erledigt ist (Sonst Rückweisung mit Kommentar)

### New Issue

Title


T09: Code bereinigt


Add [description templates](#) to help your c

Type

Issue

Assignee(s)

✓  Dominique Brodbeck  
@dominique.brodbeck

✓  Markus Degen  
@markus.degen

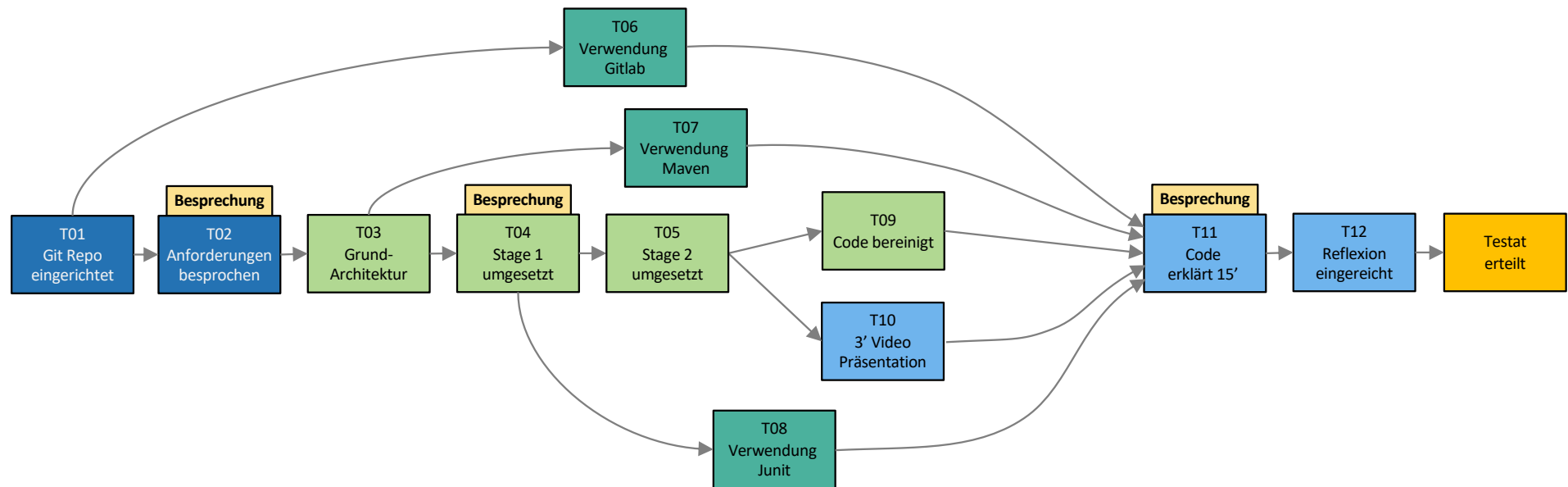
Assignee

Dominique Brodbeck + 1 more

# Tasks

ID	Text
T01	Gitlab Repository ist eingerichtet
T02	Anforderungen spezifiziert und bereinigt
T03	Grundarchitektur der Anwendung erstellt und «genehmigt»
T04	Stage 1 der Anwendung umgesetzt
T05	Stage 2 der Anwendung umgesetzt
T06	Einsatz von Gitlab (Tickets, Verwendung Code-Repository, Commit-Kommentare, Tags)
T07	Einsatz von Maven
T08	Exemplarischer JUnit Test
T09	Code bereinigt
T10	3' Video-Präsentation der Ergebnisse
T11	Code mündlich erklärt (15' pro Person)
T12	Reflexion eingereicht

# «Spielfeld»





# Tasks, erweitert

---

- ▶ **T01 Gitlab Repository eingerichtet**

Dies ist ein einfacher Task. Sobald Sie Ihr Repository erhalten haben, initialisieren Sie es mit einer ersten Version eines README.md Files (den Beispielttext durch etwas ersetzen, das für das konkrete Projekt sinnvoll ist) und erstellen einen Issue für "T01".

- ▶ **T02 Anforderungen spezifiziert und bereinigt**

Leiten Sie aus den Beschreibungen in den Kickoff-Folien die Anforderungen ab, dokumentieren Sie diese in einem zweckmässigen SRS und bereiten Sie eine Besprechung mit den Betreuenden vor, um die Anforderungen zu besprechen und priorisieren. Nach erfolgter Bereinigung der Anforderungen im Anschluss an die Besprechung, bitte einen Issue für "T02" erstellen.

- ▶ **T03 Grundarchitektur der Anwendung erstellt und «genehmigt»**

Erstellen Sie das Grundgerüst für die Anwendung mit der Möglichkeit, eine Datei zu wählen, zu laden und anzuzeigen. Verwenden Sie dafür eine "Layered Architecture" wie Sie sie aus dem Modul "Programmieren 2" kennen, so dass die Zuständigkeiten in verschiedene Schichten aufgeteilt sind. Wenn Sie das funktionsfähige Grundgerüst in Git eingchecked haben erstellen Sie bitte einen Issue für "T03".

- ▶ **T04 Stage 1 der Anwendung umgesetzt**

Nachdem Sie die als "Priorität 1" bezeichneten Leistungsmerkmale umgesetzt und in Gitlab eingchecked haben, erstellen Sie bitte einen Issue für "T04".

- ▶ **T05 Stage 2 der Anwendung umgesetzt**

Nachdem Sie die als "Priorität 2" bezeichneten Leistungsmerkmale umgesetzt und in Gitlab eingchecked haben, erstellen Sie bitte einen Issue für "T05".

- ▶ **T06 Einsatz von Gitlab gezeigt**

Den Issue "T06" können Sie erstellen, nachdem Sie einige Issues erfasst und bearbeitet haben, sowie einige Code-Commits mit "vernünftigen" Commit-Kommentaren gemacht haben.

# Tasks, erweitert (cont.)

- ▶ **T07 Einsatz von Maven gezeigt**  
Das Projekt soll komplett ohne IDE gebaut werden können, setzen Sie dafür deshalb "Maven" ein und erstellen Sie ein entsprechendes "pom.xml" File, welches die Abhängigkeiten enthält und zum Bauen der Applikation von der Kommandozeile aus verwendet werden kann. Starten der Anwendung über 'javafx:run' muss möglich sein. Den Issue "T07" können Sie auch gleichzeitig mit T03/T04/T05/T08 erstellen.
- ▶ **T08 Exemplarischer JUnit Test erstellt**  
Nachdem Sie einen (sinnvollen und funktionierenden, z.B. Test des Einlesens eines Bildes) JUnit Test zum Testen eines Aspekts Ihrer Business-Logic erstellt haben und welcher von Maven aus ausgeführt werden kann, erstellen Sie bitte einen Issue für "T08".
- ▶ **T09 Code bereinigt**  
Beim Ende einer Entwicklungsphase sollte der Code jeweils bereinigt ("refactored") werden, damit die folgenden Entwicklungsphasen auf einer sauberen Grundlage beruhen. Führen Sie eine Codebereinigung durch (Vernünftiges Benennen von Variablen, Methoden, Klassen, Verlagern von Code an den "richtigen Ort", Löschen von nicht verwendetem / auskommentiertem Code, Bereinigung von Kommentaren etc.). Erstellen Sie im Anschluss einen Issue für "T09".
- ▶ **T10 3' Video-Präsentation der Ergebnisse erstellt**  
Zeigen Sie die Funktionen Ihrer Applikation in einem rund 3 Minuten dauerenden Video (Screencast), stellen Sie dieses zur Verfügung und erstellen Sie einen Issue für "T10".
- ▶ **T11 Code mündlich erklärt (15' pro Person)**  
Jede Person der Gruppe soll während rund 15' Code aus der erstellten Anwendung erklären und Fragen beantworten - vereinbaren Sie mit den Betreuenden dafür für jede Person eine 15' WebEx Konferenz und erstellen Sie einen Issue für "T11". Dieser Issue wird geschlossen, wenn alle Mitglieder der Gruppe das Gespräch erfolgreich absolviert haben.
- ▶ **T12 Reflexion eingereicht**  
Am Schluss des Projekts reflektieren Sie noch einmal Ihren persönlichen Lernfortschritt, was Sie gelernt haben, was gut gelaufen ist, was weniger, was Sie nächstes Mal anders machen würden etc. Diese Reflexion senden Sie individuell per Mail an die Betreuenden und erstellen gleichzeitig einen Issue für "T12". Dieser Issue wird geschlossen wenn alle Reflexionen eingetroffen sind und keine Issues (T01-T11) mehr offen sind. Wenn der Issue geschlossen ist, ist der obligatorische Teil des Projekts für Sie abgeschlossen und sie erhalten das Testat "Bestanden" für das Praktikum. Sie können frei über die allfällige Restzeit des Praktikums verfügen oder können sich auf die (freiwillige) Karma/Bonus Lane begeben.

## Weitere Anforderungen

---

- ▶ Dokumente im ASCIIDoc Format
- ▶ Java >15
- ▶ Aktuelle Version muss auf “main” Branch sein
- ▶ Starten der Anwendung über das maven-Target 'javafx:run' muss möglich sein

# Die Anforderungs-Spezifikation (SRS) muss im Minimum folgende Fragen beantworten

---

## ▶ Hintergrund

- Welches Problem wird durch dieses Projekt gelöst und für wen?
- Was sind die wichtigsten Bedürfnisse der Benutzenden und warum?
- Welche Ziele sollen erreicht werden?
- Wie wird das Problem gelöst und welcher Ansatz wird gewählt?
- Was ist der Unterschied zu bestehenden Lösungen?

## ▶ Allgemeine Beschreibung

- Benutzende und deren Eigenschaften
  - Wer wird das System benutzen und wie sind diese charakterisiert?

## ▶ Funktionale Anforderungen

- Welche Aufgaben wollen die Benutzenden mit dem System durchführen können?
- Weshalb? (Welche Bedürfnisse haben sie? Welche Ziele wollen sie erreichen?)

## ▶ Weitere Anforderungen

- Nicht-funktionale Anforderungen
  - Welche Qualitäten soll das System aufweisen und wie werden diese überprüft/gemessen?
- Externe Schnittstellen
  - Mit welchen externen technischen Systemen (Hardware, Software, Kommunikation) muss das zu bauende System Informationen austauschen?
  - Wie sind diese Schnittstellen spezifiziert?

## Erzeugte Artefakte (Auf [gitlab.fhnw.ch](https://gitlab.fhnw.ch) abgelegt)

---

- ▶ Source-Code (Java)
- ▶ Anforderungs-Spezifikation (SRS)
- ▶ GUI-Entwürfe
- ▶ Projektplanung
- ▶ Architektur der erstellten Software
- ▶ Ständig aktualisierte Task-Liste
- ▶ Reflexion (ca. eine halbe Seite, am Schluss des Projekts, Einzeln pro Person, kann auch per Mail abgegeben werden)

# Projektplanung

- ▶ Strukturieren Sie das Projekt zu Beginn mit Hilfe von «Meilensteinen» und ordnen Sie dann die Tasks (Issues) den Meilensteinen zu
- ▶ Für die Grobplanung können Sie ein beliebiges anderes Tool verwenden, gut eignet sich dafür z.B. auch ein einfaches Spreadsheet:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
1		11/23	11/24	11/25	11/26	11/27	11/28	11/29	11/30	12/1	12/2	12/3	12/4	12/5	12/6	12/7	12/8	1
2	Planung																	
3	RequirementSpec																	
4	Review																	
5	Sprint 1																	
6	Sprint 2																	
7	Besprechungen																	
8																		
9																		

...in diesem Fall bitte ein PDF im Repository ablegen

## Dokumentenablage

---

- ▶ Auf [gitlab.fhnw.ch](https://gitlab.fhnw.ch) (In Software Engineering eingeführt)
- ▶ Bitte auf der obersten Ebene die folgende Ordnerstruktur einhalten (Sie können bei Bedarf natürlich weitere Unterordner erzeugen):
  - [src](#) (Java Source Code)
  - [pom.xml](#) (Maven File mit Dependencies und Build-Instruktionen)
  - [doc](#) (Gui-Sketches, Anforderungen, Dokumentation, Reflexion)
  - [README.md](#) (Text Datei, welche den Inhalt des Repositories und den Start des Programms beschreibt)

## Projektgruppen / Gitlab-Repository

---

- ▶ Organisieren Sie sich in 5 Gruppen (3 \* 4er, 2 \* 3er) Gruppen
- ▶ Melden Sie den Betreuern die Namen Ihrer Gruppe
- ▶ Die Betreuer erstellen dann ein Gitlab-Repository mit einem entsprechenden Projektnamen und legen diese Präsentation und Testdaten darin ab



## Konkrete erste/nächste Schritte

---

- ▶ Gruppen bilden, melden und auf Repository warten
- ▶ README.md hinzufügen und Issue für T01 erstellen und den Betreuern zuweisen
- ▶ Beispieldaten sichten, Aufgabenstellung lesen und verstehen
- ▶ Erstellung eines zweckmässigen SRS (Vision, Benutzende, Rahmenbedingungen, Anforderungen zur Priorisierung)
- ▶ Erstellung einer ersten Version eines Projektplans
- ▶ Skizzen für ein mögliches GUI erstellen
- ▶ Projekt initialisieren, Applikationsgerüst und initiales Maven-File erstellen
- ▶ Vorbereitung auf Besprechung
- ▶ Themen der ersten Besprechung: Anforderungen (funktional, nicht-funktional) und deren Priorisierung, Zeitplan, GUI-Entwürfe, Fragen

