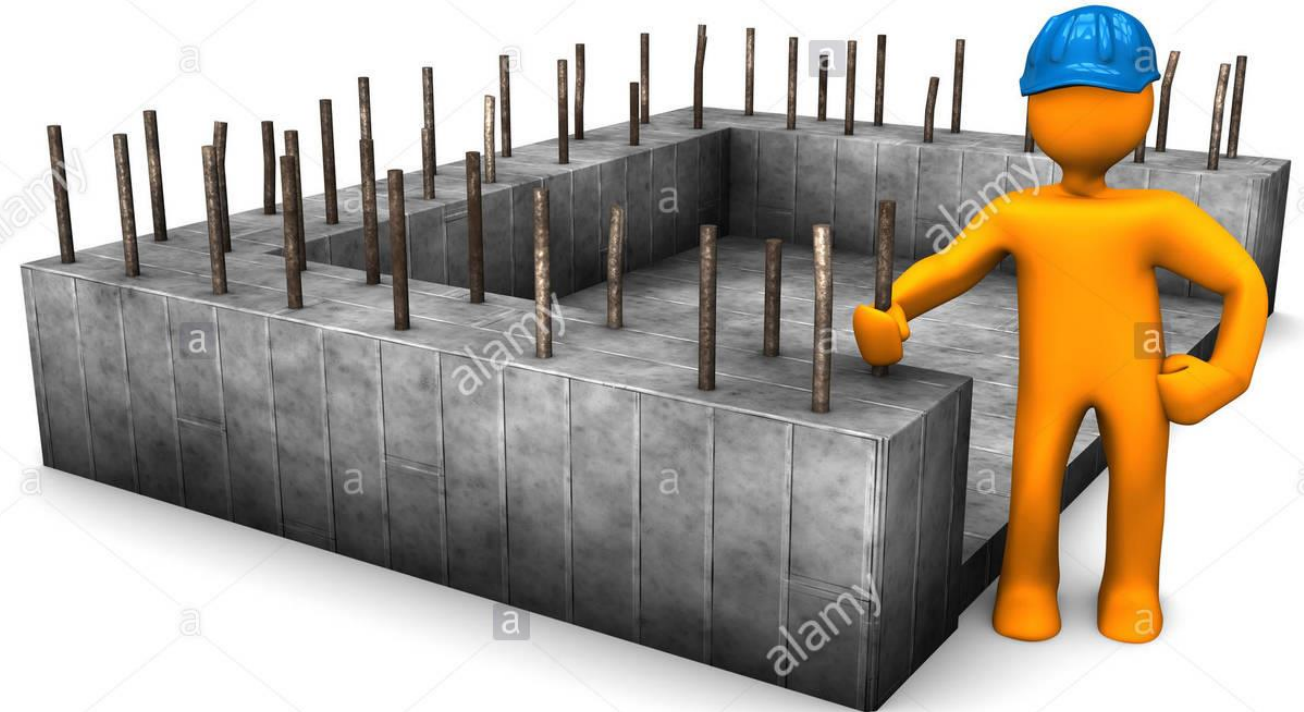


# Das Basis-System

Vorstellung und Einrichtung des SWP Basissystems

... und ganz viele andere Dinge.



 alamy stock photo

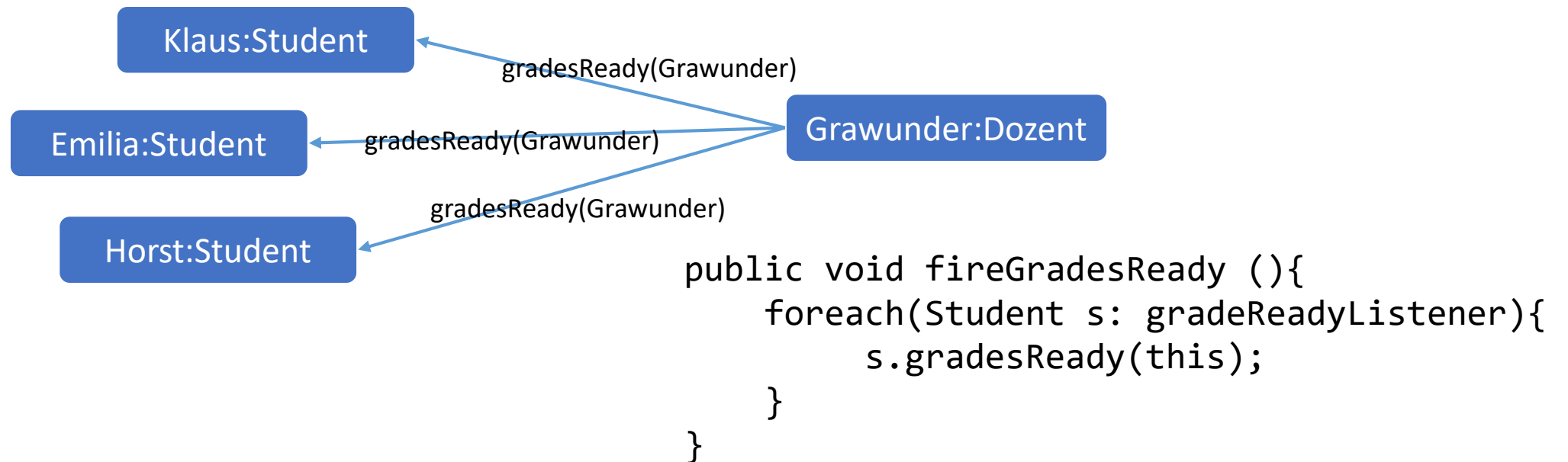
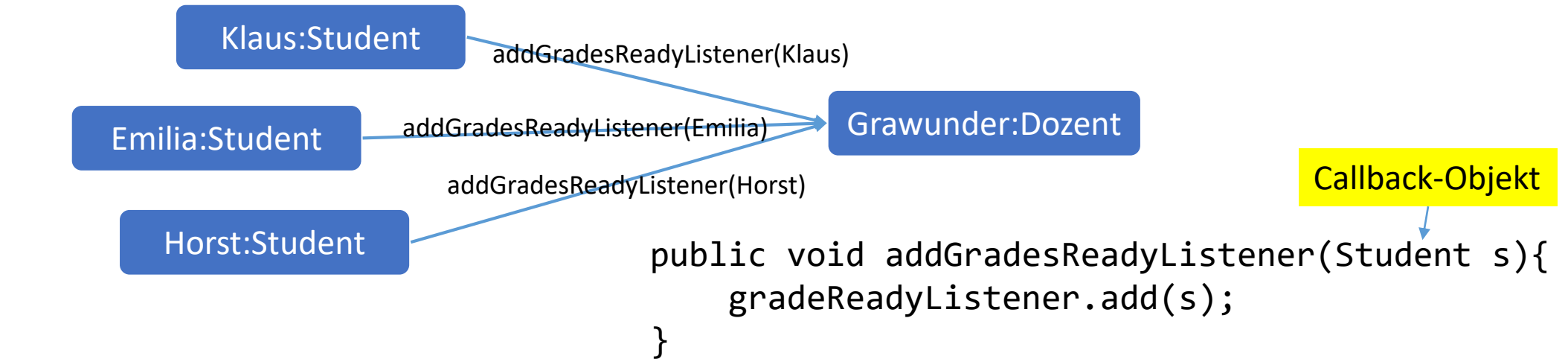
H6TPCR  
www.alamy.com

- Muss von **JEDEM einzeln** durchgeführt werden
- Richten Sie bei sich IntelliJ ein.
  - Sie können bei JetBrains einen Studierenden-Account bekommen und können die Profi-Version (Ultimate) verwenden.
- Clonen Sie das Basissystem (aus Ihrem **Gruppen-Repo**)
- Richten Sie Ihr Basissystem so ein, dass es mit IntelliJ und ohne Maven läuft.
- Starten Sie den Server und starten Sie zweimal den Client
  - Testen Sie das Einloggen: Nutzer: test1, Password: test1 bzw. test2/test2
- Hinweise:  
<https://confluence.swl.informatik.uni-oldenburg.de/display/SWP/SWP+Basisprojekt+mit+IntelliJ>

- Software Entwicklung faktisch nie mehr „auf der grünen Wiese“
- Einstieg in das SWP für (hoffentlich) viele erleichtern
- Architekturvorgaben (über Architektur kann man sich in der Bachelorarbeit und im Masterstudium Gedanken machen)
- Basissystem zeigt an vielen Stellen Beispiele, wie etwas konform zu den Anforderungen gemacht werden kann
- Das Basissystem ist nicht perfekt! Es enthält Fehler und Dinge, die man besser machen kann.
- Es ist erlaubt,
  - hier Anpassungen vorzunehmen, die notwendig erscheinen
  - das Basissystem ohne Anpassungen direkt für das eigene Projekt zu verwenden
- Jede Gruppe hat bereits einen Fork des Basissystems bekommen (ohne Historie)
- Übungsaufgaben für Einzelpersonen (!) auf dem Basissystem
- Das Basissystem ist **ereignisgetrieben** konzipiert.



- **Szenario:**
  - Student hat **Klausur** geschrieben
  - möchte vom Dozenten informiert werden, wenn die **Ergebnisse** vorliegen
- **Ansatz:**
  - Student gibt Dozent seine E-Mail-Adresse mit dem Benachrichtigungswunsch
  - Prüfer sammelt alle E-Mail-Adressen
  - Prüfer korrigiert die Arbeit
  - Prüfer schickt an alle hinterlegten E-Mail-Adressen, dass die Ergebnisse vorliegen
  - Studenten holen die Ergebnisse ab
- **Asynchron** → Niemand „blockiert“ hier, kein „Busy Waiting“
- **Ereignisgetrieben** → Wenn es etwas gibt, erfolgt eine Nachricht
- In Java statt der E-Mail ein **Callback-Objekt** bei dem Methode aufgerufen wird



```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class Dozent {
5
6     List<Student> gradeReadyListener = new ArrayList<>();
7
8     public void addGradesReadyListener(Student s) {
9         this.gradeReadyListener.add(s);
10    }
11
12    public void work() {
13        //
14        fireGradesReady();
15    }
16
17    private void fireGradesReady() {
18        for(Student s: gradeReadyListener) {
19            s.gradeReady(this);
20        }
21    }
22 }
```

Student möchte informiert werden

Informieren aller Studenten

- Klasse Dozent bekommt public-Methode:
  - addGradesReadyListener(Student s)
- Klasse Student bekommt public-Methode:
  - gradesReady(Dozent d)
- Zwei **Callback-Objekte**
  - Student (will informiert werden) und
  - Dozent (die Noten welches Prüfers sind eigentlich fertig) werden jeweils übergeben
- Das kann man so machen, aber: Warum ist dies so noch keine „schöne“ Lösung?
- Was, wenn das P-Amt auch wissen möchte, wann die Noten vorliegen?
  - addGradesReadyListener(P-Amt p) ???

```
3
4 public class Dozent {
5
6     List<Student> studentGradeReadyListener = new ArrayList<>();
7     List<PAmt> pamtGradeReadyListener = new ArrayList<>();
8
9     public void addGradesReadyListener(Student s) {
10         this.studentGradeReadyListener.add(s);
11     }
12
13     public void addGradesReadyListener(PAmt s) {
14         this.pamtGradeReadyListener.add(s);
15     }
16
17
18     public void work() {
19         //
20         fireGradesReady();
21     }
22
23     private void fireGradesReady() {
24         for(Student s:studentGradeReadyListener) {
25             s.gradeReady(this);
26         }
27         for(PAmt p:pamtGradeReadyListener) {
28             p.gradeReady(this);
29         }
30     }
31 }
32
```



- und jetzt wollen auch noch die Eltern informiert werden ...



- ist (in diesem Beispiel) dem Dozent doch egal, wer sich für die Fertigstellung der Noten interessiert
- Ansatz: Interface einführen

```
1
2 public interface GradeReadyListener {
3     void gradeReady(Dozent dozent);
4 }
5
6 public class Student implements GradeReadyListener{
7
8     @Override
9     public void gradeReady(Dozent dozent) {
10
11     }
12 }
13
14 public class PAmt implements GradeReadyListener {
15
16     @Override
17     public void gradeReady(Dozent dozent) {
18
19     }
20 }
21 }
```

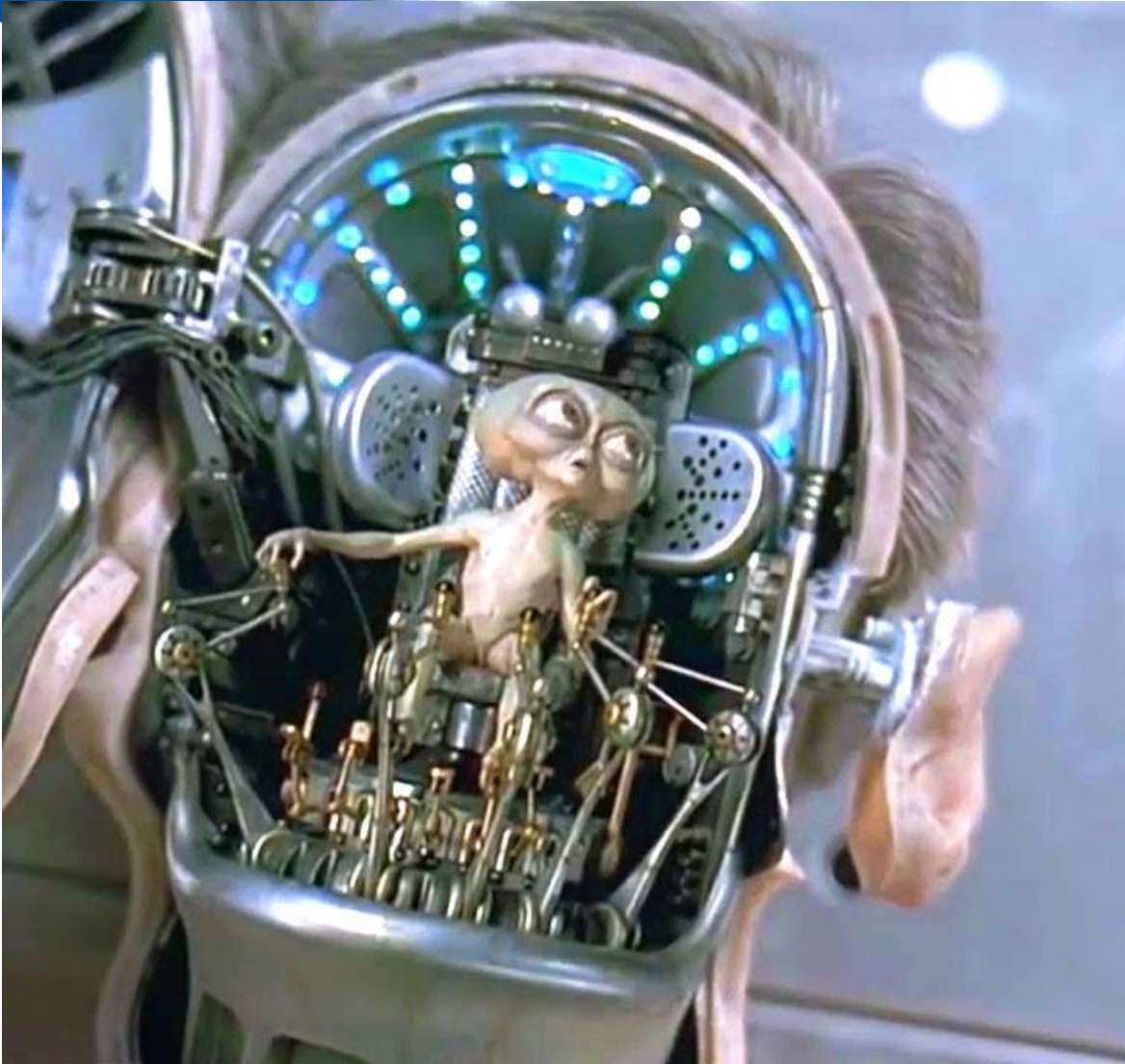
```
public class Dozent {  
  
    List< GradeReadyListener> gradeReadyListener = new ArrayList<>();  
  
    public void addGradesReadyListener( GradeReadyListener s) {  
        this.gradeReadyListener.add(s);  
    }  
  
    public void work() {  
        //  
        fireGradesReady();  
    }  
  
    private void fireGradesReady() {  
        for ( GradeReadyListener l : gradeReadyListener) {  
            l.gradeReady(this);  
        }  
    }  
}
```

Kann Student, P-Amt ...  
oder wer auch immer sein

Oder auch mit Lambda

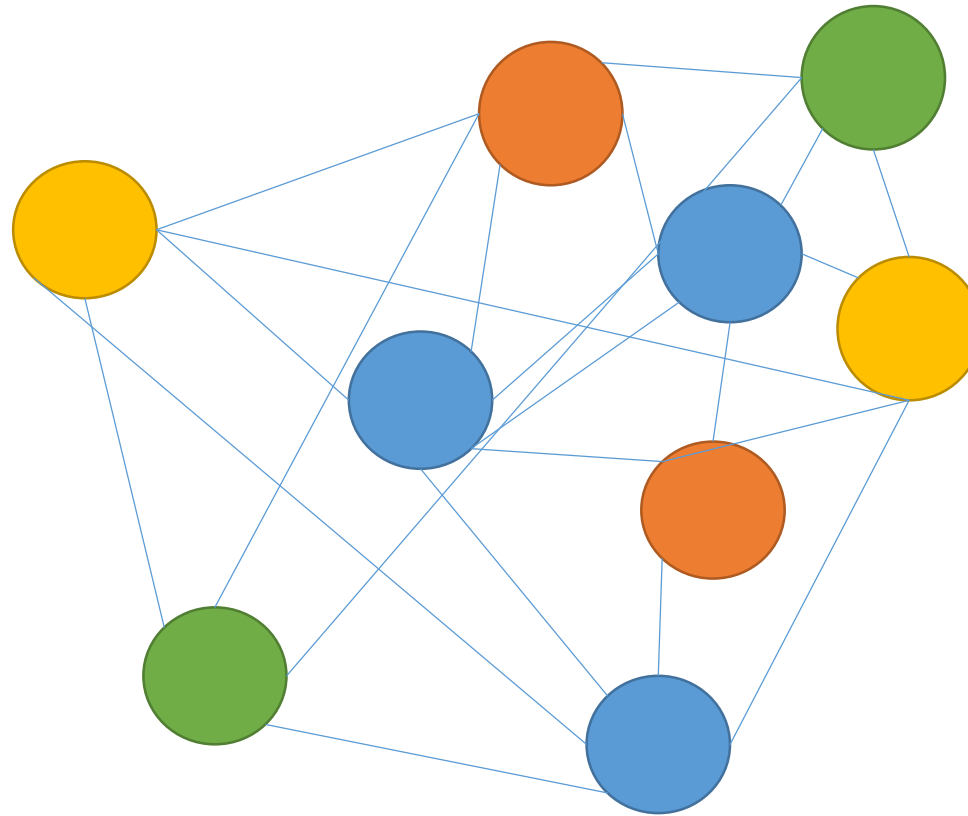
- Das Konzept nennt sich **Observer Pattern**
- Immer dann, wenn nicht aktiv gewartet werden soll/kann
- Oft im Kontext von GUIs, aber (hier) auch in der grundlegenden Kommunikation



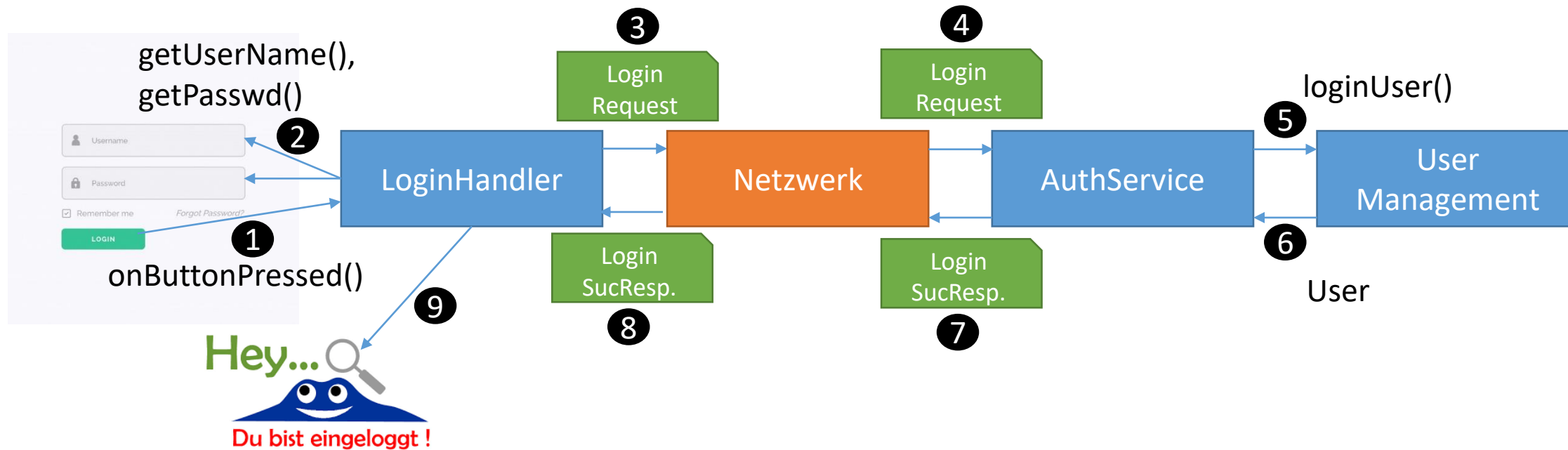


Dann ist man auch nicht  
so abhängig,  
wenn sich etwas ändert

- Immer wichtiges Ziel: Abhängigkeit reduzieren, damit man etwas ändern kann!



- Der grundlegende Ablauf ist durch **Ereignisse** gesteuert (reaktiv)
- Man schickt eine **Nachricht** an den Server: z.B. **LoginRequest** (mit Name und Passwort)
- Der Server empfängt die Nachricht und leitet sie an die Komponente weiter, die sich um das Einloggen kümmert (z.B. **AuthenticationService**)
- Der Service nutzt i.d.R. andere Komponenten, um seine Aufgabe zu erfüllen (z.B. das **UserManagement** um die Login-Daten zu überprüfen) und schickt eine Antwortnachricht an den Aufrufenden (z.B. **LoginSuccessfulResponse**, **LoginFailedResponse**) und an alle anderen (z.B. **UserLoggedInMessage**)



Als Nutzer möchte ich mich beim Spiel einloggen können, damit ich mit anderen Nutzen chatten und spielen kann.

Der Nutzer ist bereits im System registriert.

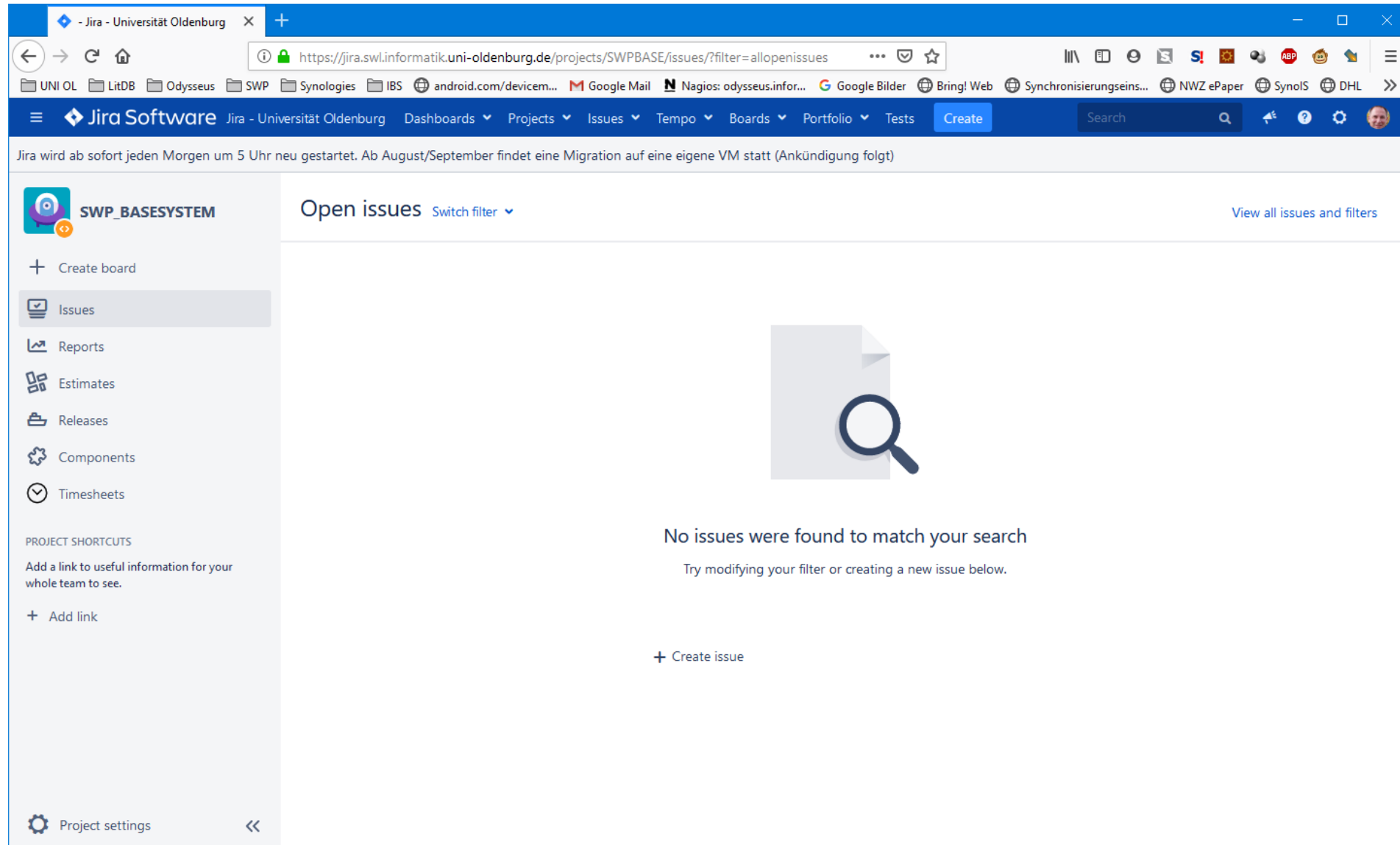
Der Nutzer soll informiert werden, wenn der Login fehlschlägt

Wenn der Login erfolgreich ist, soll der Nutzer auf die Hauptmenüseite weitergeleitet werden.

Alle anderen Nutzer sollen sehen können, dass sich jemand Neues angemeldet hat.



- Bevor man irgendetwas am Code macht, muss man zunächst ein **Ticket in Jira** erstellen
  - Dokumentation der Stunden
  - Nachvollziehbarkeit für die anderen: Wer arbeitet eigentlich gerade an welchen Aufgaben
  - Möglichkeiten schaffen, dass mehrere Personen gleichzeitig am System arbeiten können, ohne sich dabei zu sehr in die Quere zu kommen
- **Anmerkung:**
  - Später wird es die Tickets vorher geben.
  - NIEMALS: Irgendetwas am Code ändern, ohne dafür ein Ticket zu haben!



Jira - Universität Oldenburg

https://jira.swl.informatik.uni-oldenburg.de/projects/SWPBASE/issues/?filter=allopenissues

Jira Software Jira - Universität Oldenburg Dashboards Projects Issues Tempo Boards Portfolio Tests Create Search

Jira wird ab sofort jeden Morgen um 5 Uhr neu gestartet. Ab August/September findet eine Migration auf eine eigene VM statt (Ankündigung folgt)

**SWP\_BASESYSTEM**

+ Create board

Issues

Reports

Estimates

Releases

Components

Timesheets

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

+ Add link

Project settings

Open issues Switch filter

View all issues and filters

No issues were found to match your search

Try modifying your filter or creating a new issue below.

+ Create issue



## Create an Agile board

- ☒ Board from an existing project  
Boards can contain one or more projects.
- ☐ Board from an existing Saved Filter  
An advanced option using a JQL filter.

### Name this board

Board name\* SWP\_BASESYSTEM

Project(s)\* SWP\_BASESYSTEM (SWPBASE) x

Select one or more projects to be included in this board.

Name wie das  
Projekt!

Back

Create board

Cancel

SWP\_BASESYSTEM - Agile Board

https://jira.swl.informatik.uni-oldenburg.de/secure/RapidBoard.jspa?rapidView=3082&view

Jira Software Jira - Universität Oldenburg Dashboards Projects Issues Tempo Boards Portfolio Tests Create Search

Jira wird ab sofort jeden Morgen um 5 Uhr neu gestartet. Ab August/September findet eine Migration auf eine eigene VM statt (Ankündigung folgt)

**SWP\_BASESYSTEM**

Backlog

Agile Poker Board

QUICK FILTERS: Only My Issues Recently Updated

FILL YOUR BACKLOG WITH ISSUES

This is your team backlog. Create and estimate new issues, and prioritize the backlog using drag and drop.

Backlog 0 issues Create Sprint

What needs to be done?

New Bug in Backlog Open in dialog Cancel

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

+ Add link

Project settings

Anmerkung: Später ist das Backlog bereits mehr gefüllt

# Backlog

QUICK FILTERS: [Only My Issues](#) [Recently Updated](#)

VERSIONS

EPICS

Bug
 Impediment
 Improvement
 New Feature
 Story
 Task
 Waiting for...
 Meeting

FILL YOUR BACKLOG WITH ISSUES

This is your team backlog. Create and estimate new issues, and prioritize the backlog using drag and drop.


Create Sprint

...

Als Nutzer möchte ich mich beim Spiel einloggen können, damit ich mit anderen Nutzen chatten und spielen kann.

New Story in Backlog

[Open in dialog](#)
[Cancel](#)


**SWP\_BASESYSTEM**

SWP\_BASESYSTEM

Backlog

Active sprints

Releases

Reports

Issues

Estimates

Components

Timesheets

PROJECT SHORTCUTS


Add a link to useful information for your whole team to see.

+ Add link

Project settings

## Backlog

QUICK FILTERS:
 [Only My Issues](#)
[Recently Updated](#)




FILL YOUR BACKLOG WITH ISSUES

This is your team backlog. Create and estimate new issues, and prioritize the backlog using drag and drop.


Backlog 1 issue

Create Sprint

...


[SWPBASE-1](#) Als Nutzer möchte ich mich beim Spiel einloggen können, da...

+ Create issue


 SWP\_BASESYSTEM / SWPBASE-1
 

...


X

Als Nutzer möchte ich mich beim Spiel einloggen können, damit ich mit anderen Nutzen chatten und spielen kann.

Estimate: Unestimated

Details

Status: **TO DO** [\(View Workflow\)](#)

Priority:  Major

Component/s: None


Labels: None


Affects Version/s: None

Fix Version/s: None

Epic Link: None

People

Reporter:  Marco Grawunder

Assignee:  Unassigned


[Assign to me](#)


Dates


Created: Just now


Updated: Just now


# Eine Story kann nicht direkt umgesetzt werden → Subtasks


**SWP\_BASESYSTEM**


 SWP\_BASESYSTEM


 Backlog


 Active sprints


 Releases

 Reports

 Issues

 Estimates


 Components


 Timesheets

PROJECT SHORTCUTS

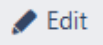
Add a link to useful information for your whole team to see.

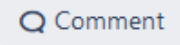
+ Add link


 Project settings


 SWP\_BASESYSTEM / SWPBASE-1


Als Nutzer möchte ich mich beim Spiel einloggen können, damit ich mit anderen Nutzen chatten und spielen kann.

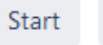
 Edit


 Comment

 Assign

 More

 Closed

 Start

 Admin

Details

Type: Story

Priority: Major

Labels: None

Description

Click to add description

Attachments

Tempo

Log Time

Log work

Add Expense

Plan Time

Start Tracker

View Worklogs

Agile Board

Rank to Top

Rank to Bottom

Attach files

Voters

Stop watching

Watchers

Create sub-task

Convert to sub-task

Create multiple Sub-Tasks

Log Time

Create sub-task for this issue

TO DO (View Workflow)

Unresolved

People

Assignee: Unassigned

Assign to me

Reporter: Marco Grawunder

Votes: 0

Watchers: 1 Stop watching this issue

Dates

Created: 2 minutes ago

Updated: 2 minutes ago


Collaborators

Development

Create branch

Agile

View on Board



THERE ARE NO OTHER ISSUES FOR THIS ISSUE.

You have no watchers for this issue.

Log Time


Create sub-task


Convert to sub-task

Create multiple Sub-Tasks

https://jira.swl.informatik.uni-oldenburg.de/secure/CreateSubTaskIssue!default.jspa?parentIssueId=63300



**SWP\_BASESYSTEM**

 SWP\_BASESYSTEM

 Backlog

 Active sprints

 Releases

 Reports

 Issues

 Estimates

 Components

 Timesheets

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

+ Add link

 Project settings


 SWP\_BASESYSTEM / SWPBASE-1

Als Nutzer möchte ich mich beim Spiel einloggen können, damit ich mit anderen Nutzen chatten und spielen kann.

 Edit

 Comment

 Assign

 More

 Closed


 Start

 Admin

 Share

 Export

**Details**

Type:  Story

Priority:  Major

Labels: None

Status: **TO DO** [\(View Workflow\)](#)

Resolution: Unresolved


**Description**

[Click to add description](#)

**Attachments**

 Drop files to attach, or browse.

**Sub-Tasks**

1. [Erstelle Login-View](#)  **TO DO** *Unassigned* [Create Sub-Task](#)

**Tempo**

**People**

Assignee:  Unassigned [Assign to me](#)

Reporter:  Marco Grawunder

Votes: 0

Watchers: [1 Stop watching this issue](#)

**Dates**

Created: 4 minutes ago

Updated: 4 minutes ago

**Collaborators**

**Development**


[Create branch](#)

**Agile**

[View on Board](#)

THERE ARE NO ACTIVITIES FOR YOU TO SEE FOR THIS ISSUE.

You have not recorded any activity on this issue.

 SWP\_BASESYSTEM

SWP\_BASESYSTEM

Backlog

Active sprints

Releases

Reports

Issues

Estimates

Components


Timesheets

PROJECT SHORTCUTS

Add a link to useful information for your whole team to see.

+ Add link

Project settings

 SWP\_BASESYSTEM / SWPBASE-1

Als Nutzer möchte ich mich beim Spiel einloggen können, damit ich mit anderen Nutzen chatten und spielen kann.

EditCommentAssignMoreClosedStartAdmin

Details

Type: StoryStatus: TO DO (View Workflow)Priority: MajorResolution: UnresolvedLabels: None

Description

Click to add description

Click to edit

Attachments

Drop files to attach, or browse.

Sub-Tasks

1. Erstelle Login-View

2. Erstelle LoginHandler

3. Erstelle Kommunikationsobjekte

4. Erstelle AuthenticationHandler

ShareExport

People

Assignee: UnassignedAssign to meReporter: Marco GrawunderVotes: 0Watchers: 1 Stop watching this issue

Dates

Created: 5 minutes agoUpdated: 5 minutes ago

Collaborators

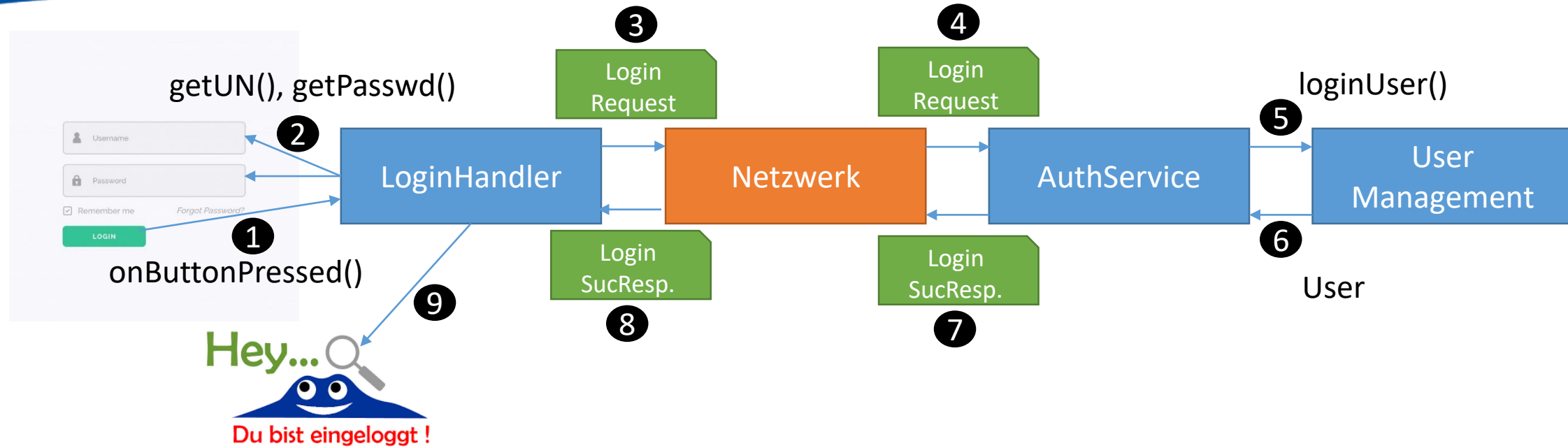
Development

Create branch

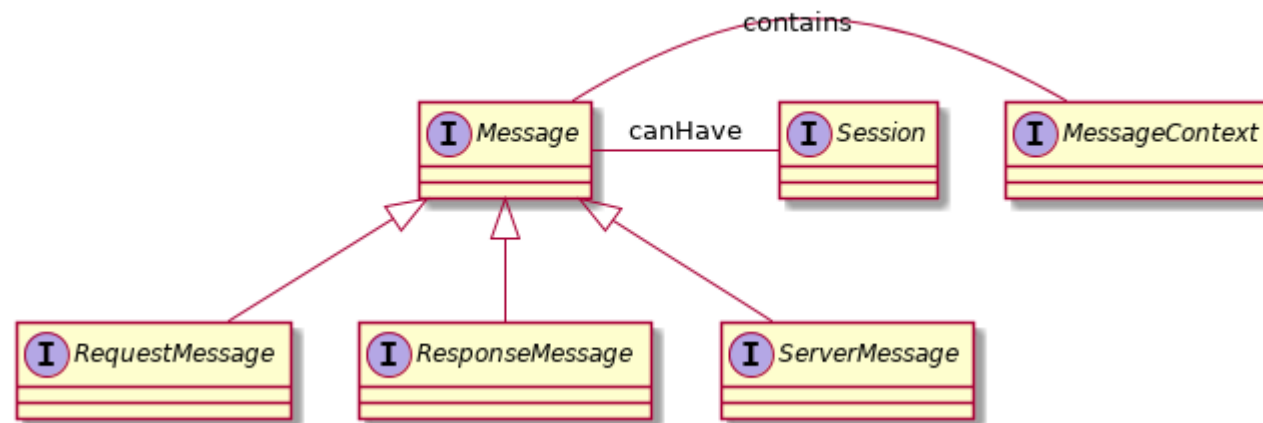
Agile

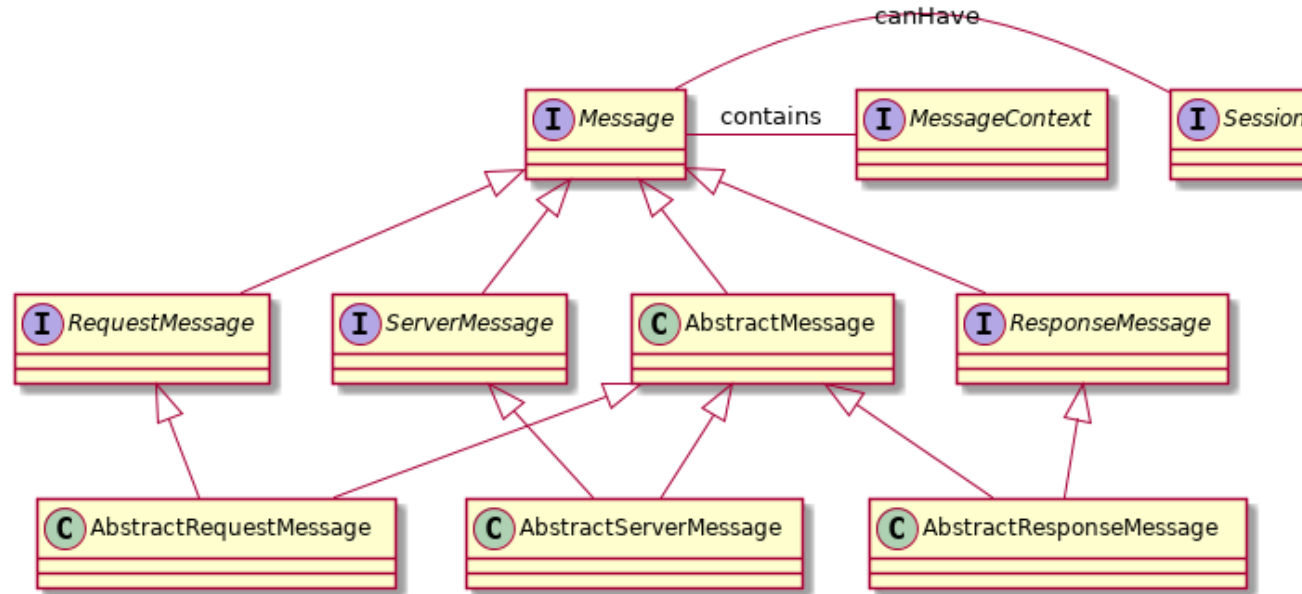
View on Board

- Reicht das so?
- Fehlt noch was?
- Wie passt das alles in den Rest hinein?
  
- Vielleicht doch noch mal etwas zurücklehnen und mehr nachdenken?



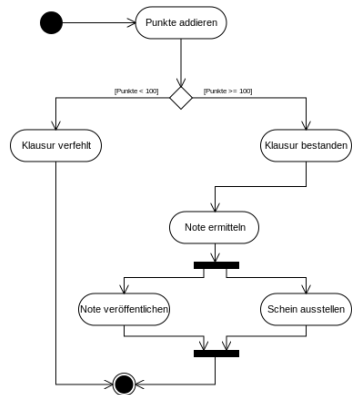
- Die Kommunikation zwischen Client und Server soll über Nachrichten (Message) erfolgen
- Es gibt drei Arten von Nachrichten:
  - **RequestMessage**: Anfragen vom Client an den Server (z.B. LoginRequest)
  - **ResponseMessage**: Antworten vom Server auf bestimmte Anfragen (z.B. LoginResponse)
  - **ServerMessage**: Nachrichten vom Server an Client, ohne (direkte) Anfrage (z.B. UserLoggedInMessage)
- Eine Nachricht (Message)
  - kann eine Session haben (d.h. ein eingeloggter „Nutzer“ hat die Nachricht gesendet)
  - kann einen Kontext (MessageContext) haben (später mehr dazu)





- Ist es für Klassenstrukturen, die mit hoher Wahrscheinlichkeit erweitert werden, sinnvoll, **abstrakte Basisklassen** einzuführen
- Später zu erweitern ist immer mit vielen Problemen verbunden
- Deswegen hier für jeden Typ zusätzlich eine abstrakte Basisklasse
- Anmerkung: Man kann darüber streiten, ob das immer sinnvoll ist ;-)

Es gibt grundsätzlich unterschiedliche Ansätze, in UML Dynamik/Abläufe zu modellieren

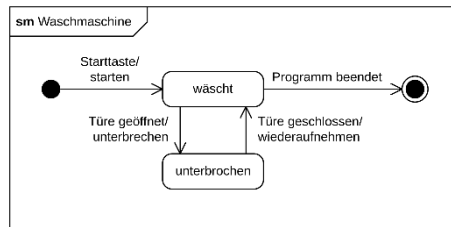


- **Aktivitätsdiagramm:**

- Immer, wenn es relativ abstrakt sein soll und die Details (welche Klassen, welche Methoden, etc.) noch nicht ganz so wichtig sind

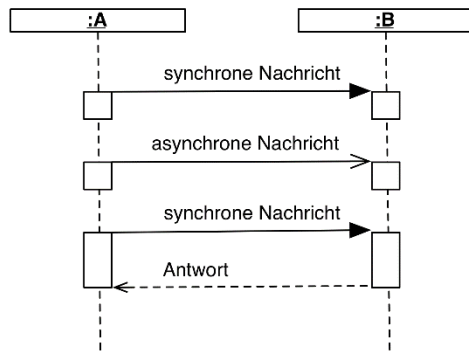
- **Zustandsdiagramm:**

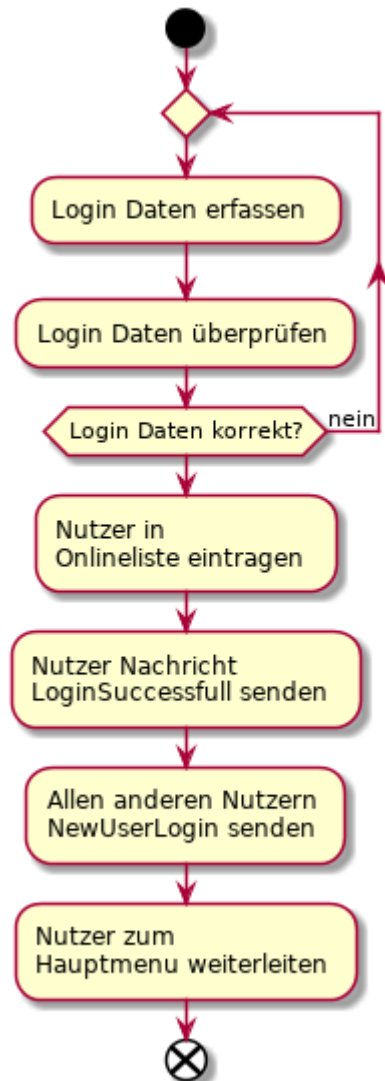
- Immer, wenn man zeigen will, dass ein System/eine Klasse unterschiedliche Zustände durchläuft (Für Spiele z.B. Würfelphase oder Zugphase)



- **Sequenzdiagramm:**

- Immer, wenn man sehr konkret klar machen will, welche Klasse bei welcher anderen Klasse eine Methode aufrufen will
- Grundsätzlich kann man in Sequenzdiagrammen auch komplexe Abläufe auf abstrakter Art modellieren, aber hier im SWP sollte es immer echte Kommunikation (Methodenaufrufe mit Parametern) und echte Klassen geben





PlantUML Macro

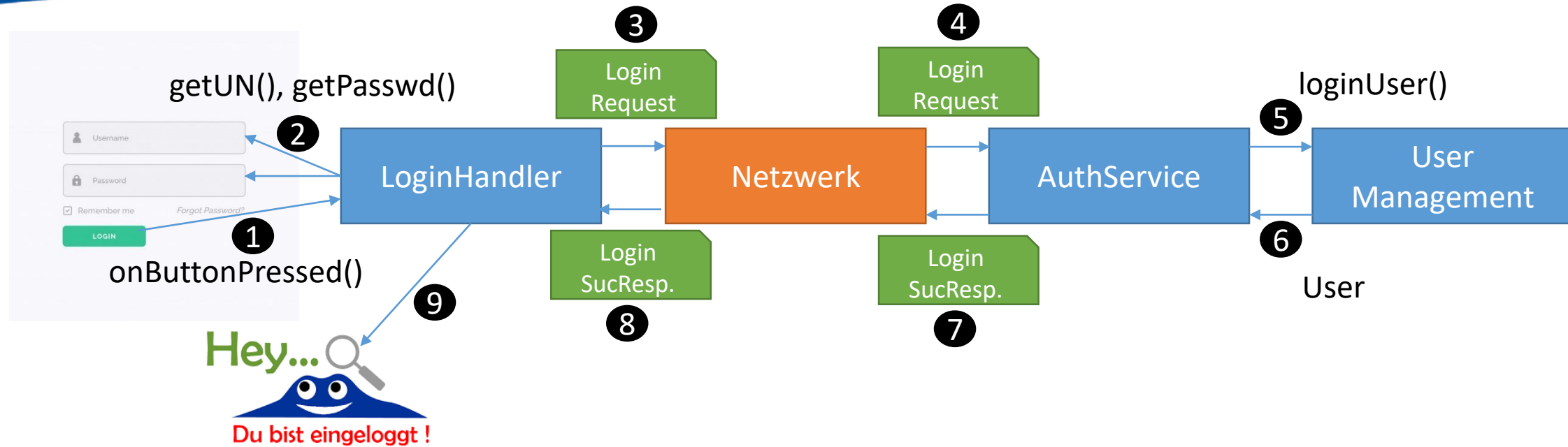
```
@startuml
start
repeat
    :Login Daten erfassen;
    :Login Daten überprüfen;
repeat while (Login Daten korrekt?) is (nein)
:Nutzer in
Onlineliste eintragen;
:Nutzer Nachricht
LoginSuccessfull senden;
:Allen anderen Nutzern
NewUserLogin senden;
:Nutzer zum
Hauptmenu weiterleiten;
end
@enduml
```

Das kann man so jedoch nicht umsetzen. Hier gibt es z.B. gar keinen Client oder Server...

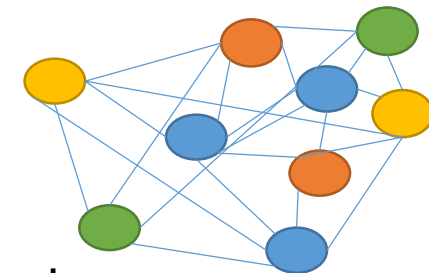
Außerdem noch ein paar Abhängigkeitsprobleme



# Problem: Wie Unabhängigkeit auf Client und Server bewahren?



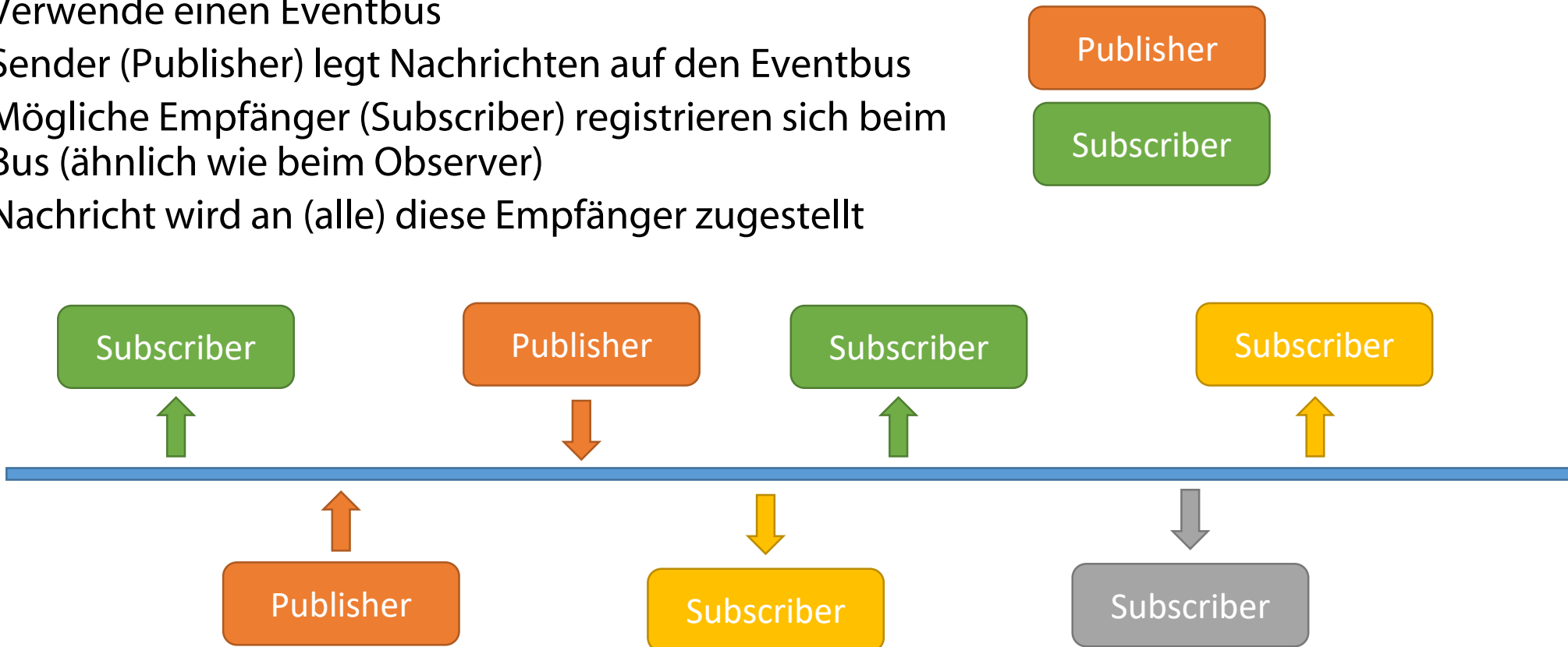
- Im Moment:
  - LoginHandler muss Netzwerk (Client-Seite) kennen
  - Netzwerk (Server-Seite) muss AuthService kennen
  - AuthService muss UserManagement kennen
  - Netzwerk muss LoginHandler kennen
  - ... und Netzwerk muss wissen, wem welche Nachrichten zu schicken sind
- Dann demnächst noch: Logout, Registrierung, Chat, Spiele

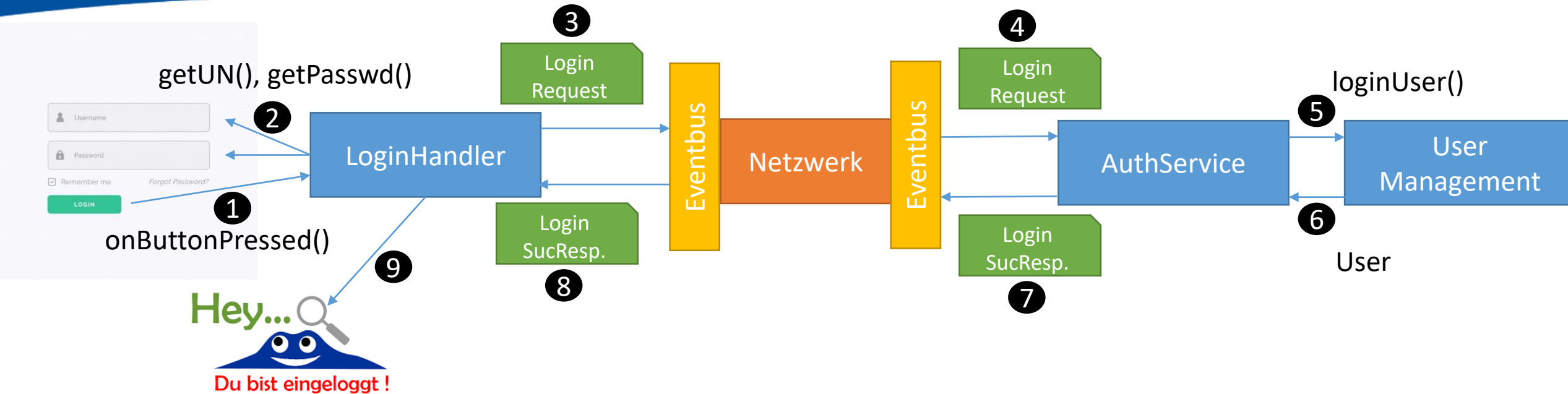




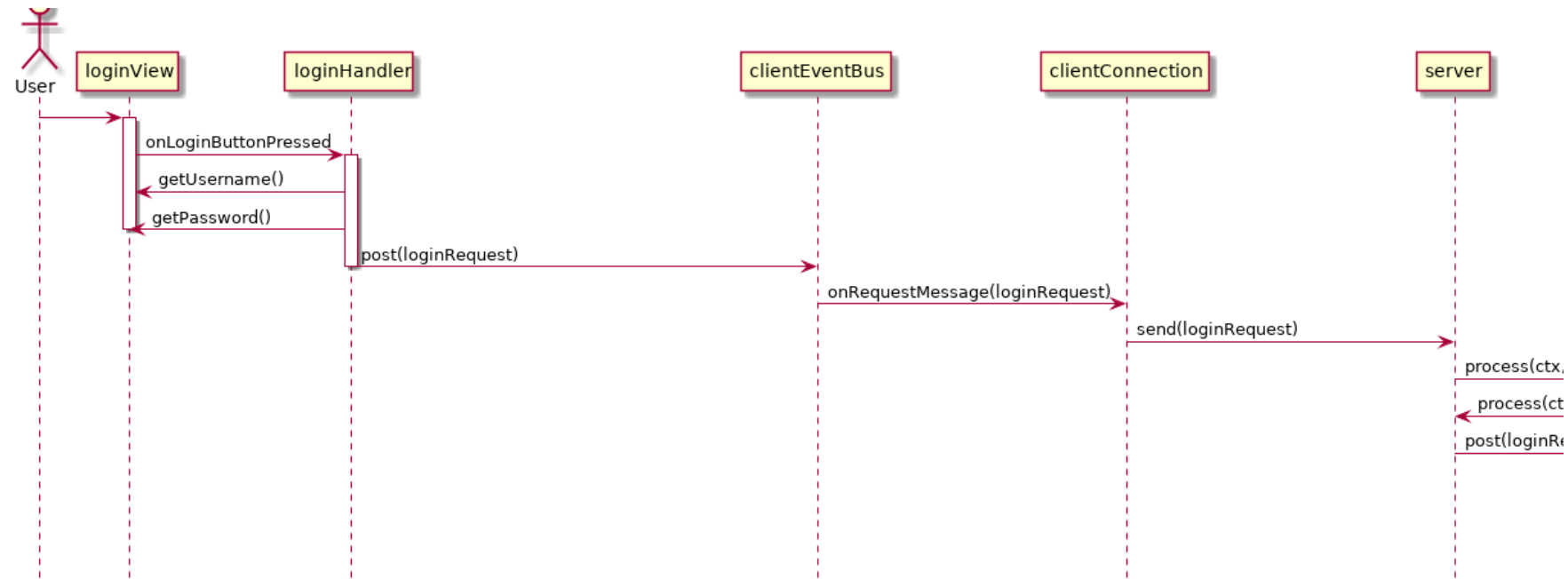
- Ansatz: Observer-Pattern
- Der **LoginHandler** registriert sich beim **Netzwerkclient** für alle Login-Ereignisse
- Der **AuthenticationService** registriert sich beim **Netzwerkserver** für alle Login-Ereignisse
- Guter Weg, führt aber zu relativ vielen zu verwaltenden Listen
- Alternative Eventbus (Publisher-Subscriber-Pattern)

- **Problem:**
  - Welche (Netzwerk-)Nachrichten, sollen eigentlich an wen weitergeleitet werden
- **Ansatz**
  - Verwende einen Eventbus
  - Sender (Publisher) legt Nachrichten auf den Eventbus
  - Mögliche Empfänger (Subscriber) registrieren sich beim Bus (ähnlich wie beim Observer)
  - Nachricht wird an (alle) diese Empfänger zugestellt

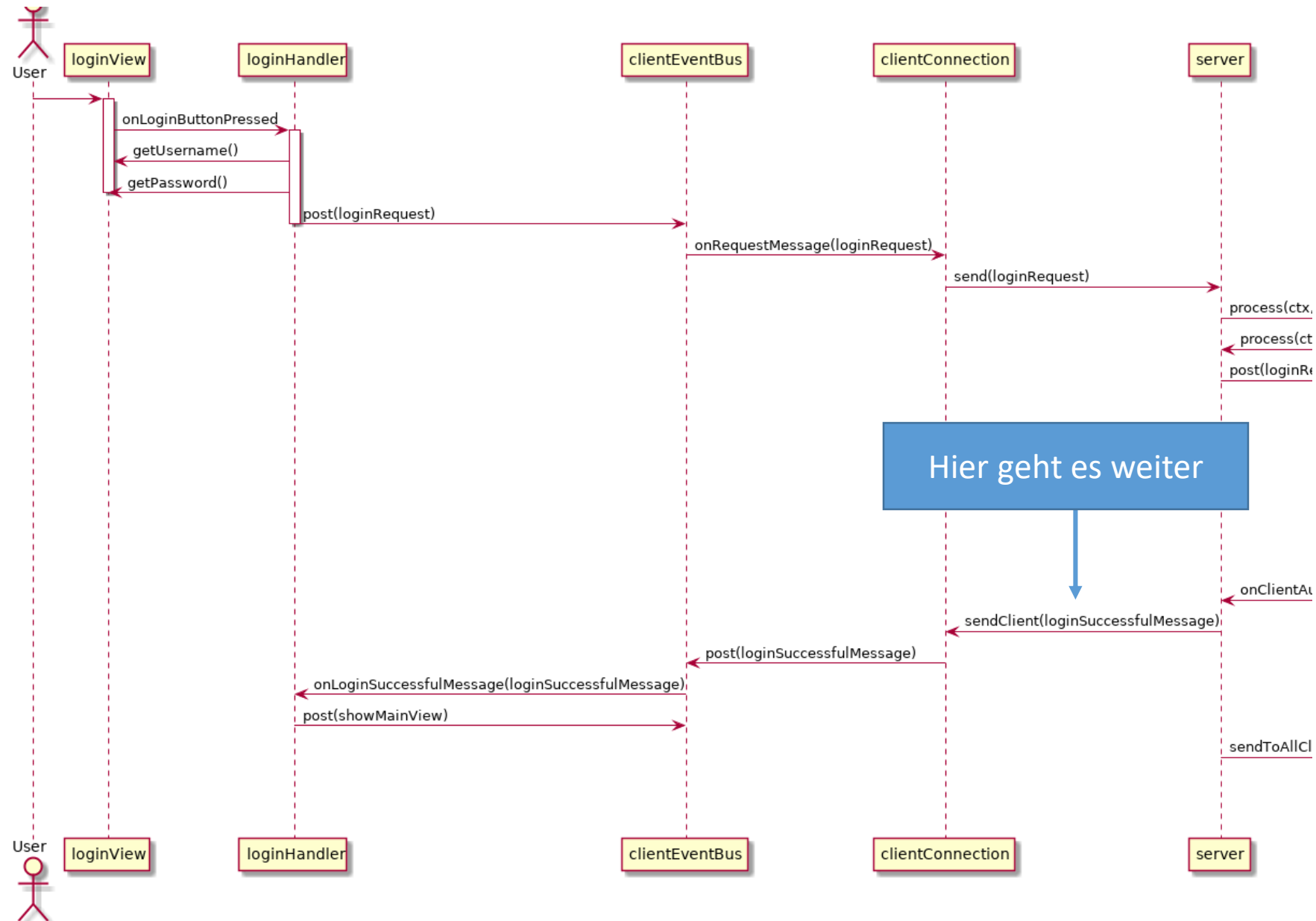


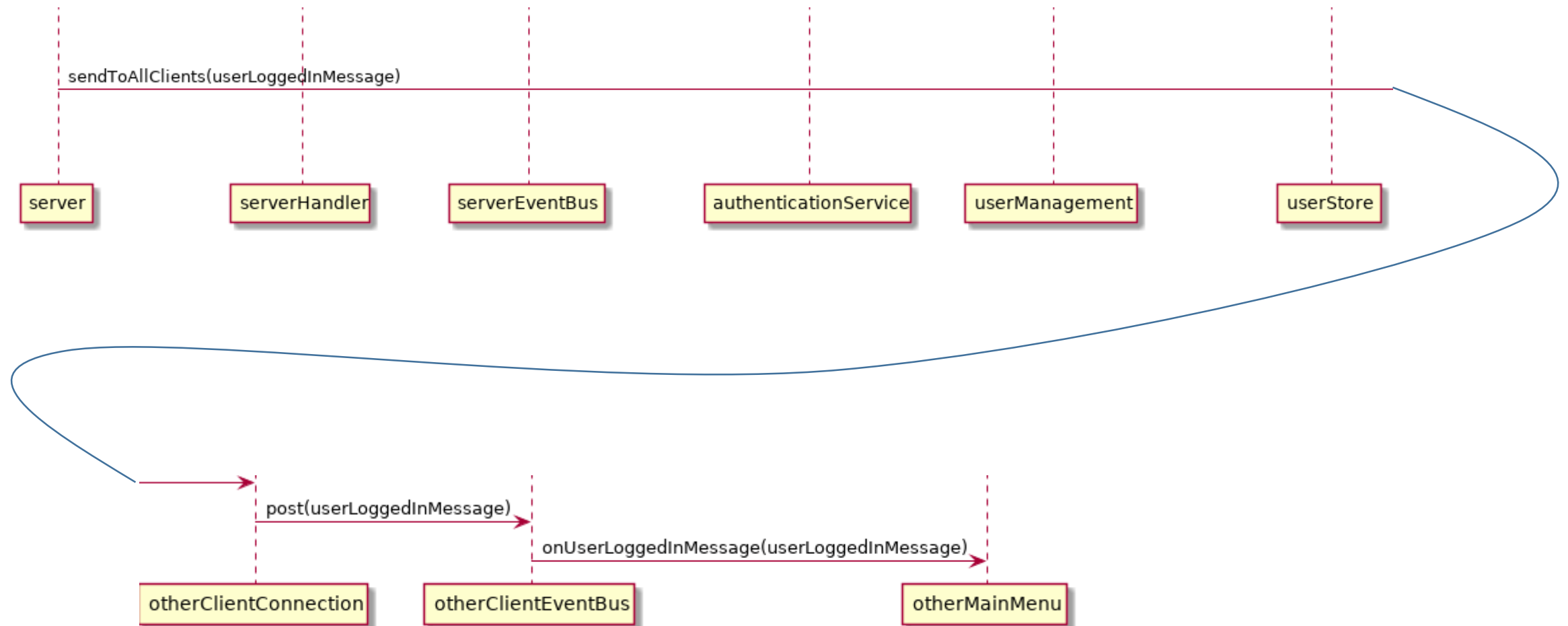


- Und das Netzwerk ist nun vollständig von LoginHandler und AuthService entkoppelt
- Die Darstellung ist aber immer noch zu ungenau für eine Umsetzung
- → Sequenzdiagramm



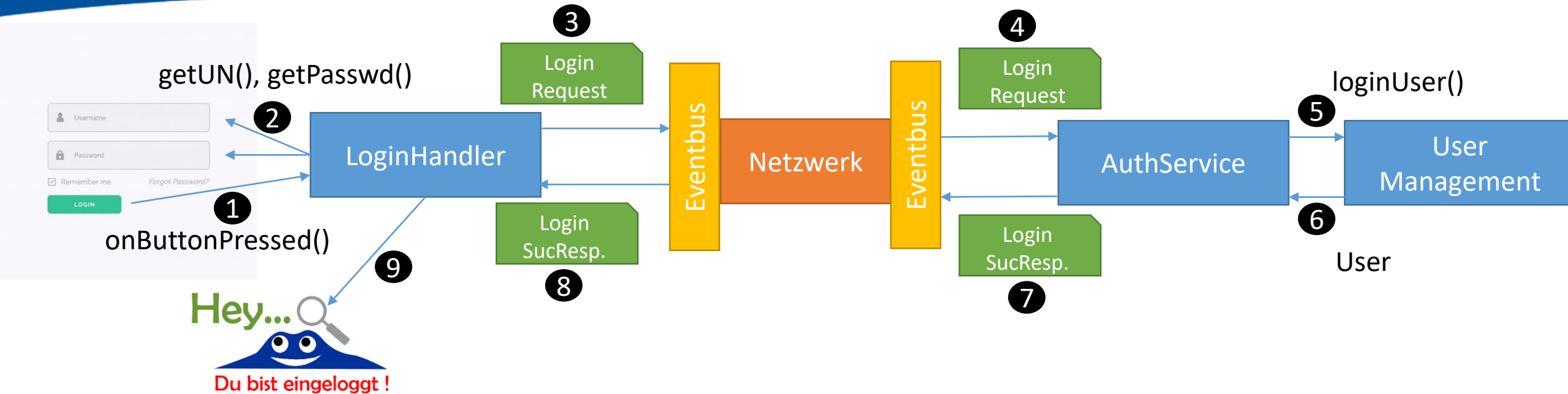






Anmerkung: Die Lebenslinien fehlen ...





- **Publisher**

- `eventbus.Post(Message)`: Nachricht auf Bus legen

- **Subscriber:**

- `eventbus.register(this)`: Grundsätzlich für Nachrichten auf dem Bus registrieren
- `@Subscribe`-Annotation: Methode wird aufgerufen, wenn auf Eventbus Nachricht mit dem Typ aus der Methode gepostet wird

- Die Server-Bibliothek (Netty, in anderer VL dazu mehr) ruft die folgende Methode auf, wenn eine neue (Request-)Nachricht am Server ankommt:

Eventbus

Netzwerk

Wer ist der Absender

Was ist die Nachricht

```
@Override
public void process(ChannelHandlerContext ctx, RequestMessage msg) {
    LOG.debug("Received new message from client "+msg);
    try {
        msg.setMessageContext(new NettyMessageContext(ctx));
        checkIfMessageNeedsAuthorization(ctx, msg);
        eventBus.post(msg);
    } catch (Exception e) {
        LOG.error("ServerException " + e.getClass().getName() + " " + e.getMessage());
        sendToClient(ctx, new ExceptionMessage(e.getMessage()));
    }
}
```

Nachricht um den Absender erweitern

Teste ob der Absender für diese Nachricht eingeloggt sein muss? Exception falls notwendig aber nicht eingeloggt

Abschließend: Lege (post) die Nachricht auf den Bus

Publisher



@Subscribe als Annotation haben

Den passenden Typ des Events als Parameter haben

```
@Subscribe
public void onLoginRequest(LoginRequest msg) {
    if (LOG.isDebugEnabled()) {
        LOG.debug("Got new auth message with " + msg.getUsername() + " " + msg.getPassword());
    }
    ServerInternalMessage returnMessage;
    try {
        User newUser = userManagement.login(msg.getUsername(), msg.getPassword());
        returnMessage = new ClientAuthorizedMessage(newUser);
        Session newSession = UUIDSession.create();
        userSessions.put(newSession, newUser);
        returnMessage.setSession(newSession);
    } catch (Exception e) {
        LOG.error(e);
        returnMessage = new ServerExceptionMessage(new LoginException("Cannot auth user " + msg.getUsername()));
    }
    returnMessage.setMessageContext(msg.getMessageContext());
    bus.post(returnMessage);
}
```

Antwortnachricht

Das eigentliche Login, wirft Exception, wenn Username und Passwort nicht passen

Antwortnachricht erfolgreiches Anmelden

Nutzer eingeloggt merken

Nachricht um Session erweitern

Fehler bei Anmeldung

Neue Nachricht um ursprünglichen Absender erweitern

Nachricht auf den Bus legen

- Hier nun etwas anders als im Bild oben
- AuthenticationService liefert eine serverinterne Nachricht zurück (ServerInternalMessage)
- Grund:
  - Der Server (die Klasse, die die Verbindungen zum Client hält) möchte wissen, welche Verbindungen auch erfolgreich eingeloggt sind
  - Macht ein Mapping zwischen der Verbindung (ChannelHandlerContext) und den Sessions
  - D.h. der Server empfängt vom Eventbus die Nachricht, dass sich ein Client erfolgreich eingeloggt hat (ClientAuthorizedMessage) und kann dann die Infos eintragen
  - Zusätzlich kann der Server dann eine Nachricht an den Aufrufenden (LoginSuccessfulMessage) und alle anderen Clients (UserLoggedInMessage) senden
- Vorteile:
  - Der AuthService muss nichts über Verbindungen wissen
  - Das Testen wird durch weniger Abhängigkeiten einfacher

- Empfange Nachrichten, die der AuthenticationHandler auf den Bus gelegt hat

@Subscribe als Annotation haben

Den passenden Typ des Events als Parameter haben

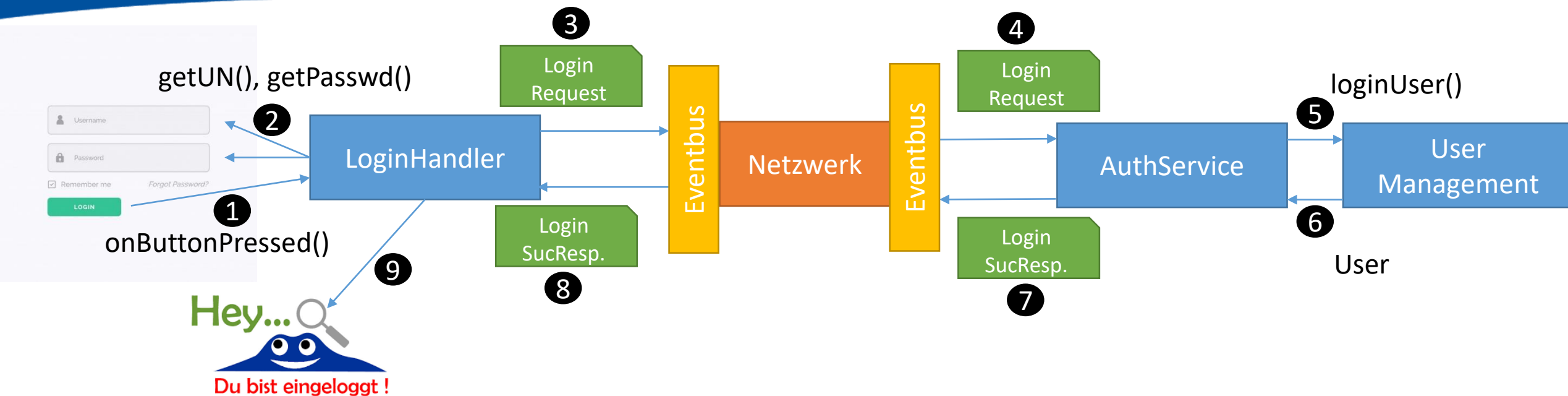
```
@Subscribe
private void onClientAuthorized(ClientAuthorizedMessage msg){
    Optional<ChannelHandlerContext> ctx = getCtx(msg);
    if (ctx.isPresent()) {
        putSession(ctx.get(), msg.getSession());
        sendToClient(ctx.get(), new LoginSuccessfulMessage(msg.getUser()));
        sendToAll(new UserLoggedInMessage(msg.getUser().getUsername()));
    }else{
        ...
    }
}
```

Von wem kam die ursprüngliche Nachricht?  
Der braucht die Antwort.

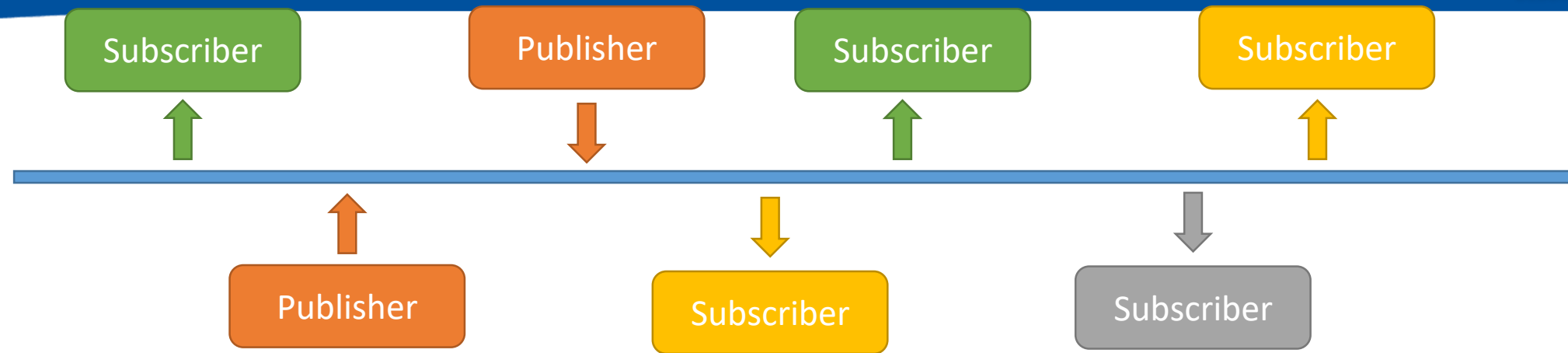
Server merkt sich, dass für  
diese Verbindung,  
ein Login erfolgreich war  
(Session ist vorhanden)

Sendet Nachricht an Absender,  
LoginSuccessfulMessage

Sendet Nachricht an alle anderen:  
UserLoggedInMessage



- Was, wenn man nun Änderungen am Code machen möchte?
- Vor allem so, dass man mit anderen Team-Mitglieder nicht kollidiert?
- Antwort: Versionsverwaltungssystem



- Subscriber registrieren sich am Bus für bestimmte Typen von Nachrichten
- Publisher senden bestimmte Nachrichten auf den Bus
- Der Bus sorgt dafür, dass alle passenden Subscriber die Nachricht erhalten
- Vorteile:
  - Weniger Verwaltungsaufwand als z.B. manuelles Observerlisten
  - weniger enge Kopplung
- Nachteil:
  - Es ist nicht immer sofort offensichtlich, wo Nachrichten hingehen (wer hat sich registriert)
  - Testen wird durch den Bus komplexer (siehe VL zum Thema)

- Was, wenn man nun Änderungen am Code machen möchte?
- Vor allem so, dass man mit anderen Team-Mitglieder nicht kollidiert?
- Antwort: Versionsverwaltungssystem



# Git

Wie bekommt man Änderungen am Code koordiniert?

- Wenn man gemeinsam an Dingen arbeitet, ist es wichtig, das zu koordinieren
- Versionsverwaltungssystem: git
- Grundsätzliche Idee: Es gibt ein zentrales Repository und jeder hat lokale Kopie dieses Repositories

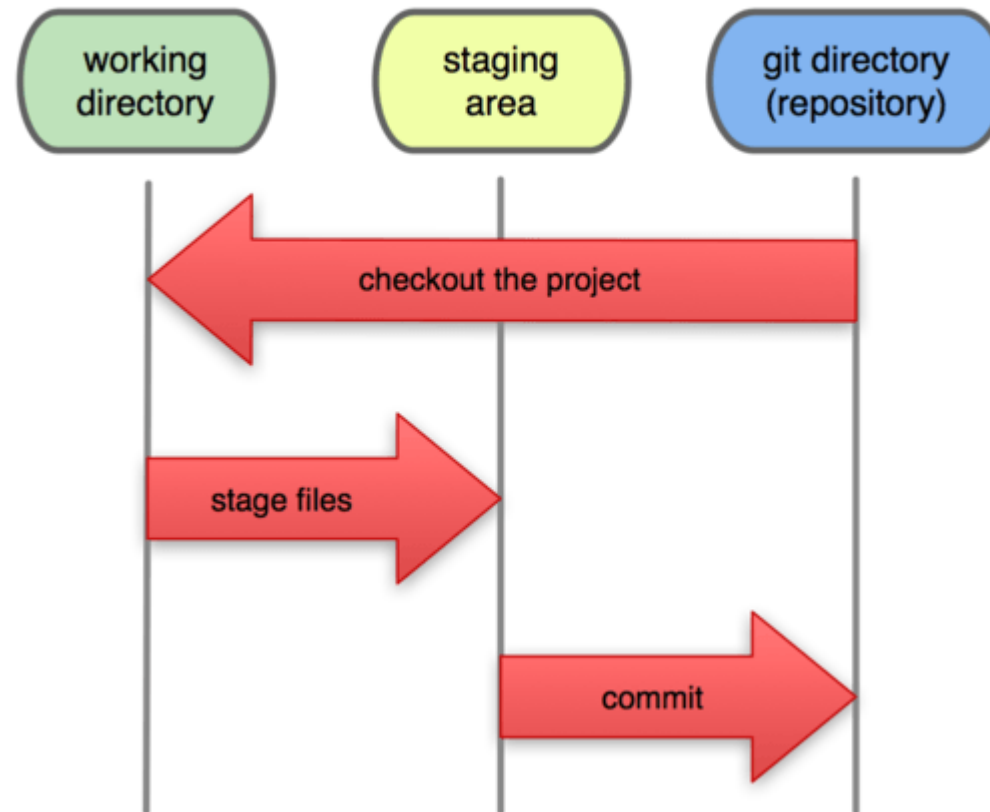
- **Clone:** Eine Version des zentralen Repositories bei sich lokal auschecken

```
> git -clone https://git.swl.informatik.uni-oldenburg.de/scm/.../swpbaseproject.git
```

- **Pull:** Änderungen, die sich auf globaler Ebene ergeben haben bei sich in das lokale Repository einpflegen (z.B. als merge)
- **Commit:**
  - Eine Änderung wird lokal im eigenen Repository vermerkt (z.B. Änderungen an Dateien, Löschen von Dateien oder Hinzufügen von Dateien)
  - Jedes Commit bekommt eine Nachricht: Was ist hier geändert worden. Wichtig! Dabei muss auch immer die **Nummer des Tickets** zu dem dieser Commit gehört, in die Nachricht aufgenommen werden (Bitbucket kann dann Commit mit Ticket in Jira verknüpfen)
- **Push:** Alle lokalen Änderungen an das Hauptrepository übermitteln

Quelle der (meisten) folgenden Informationen/Bilder: <https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud>

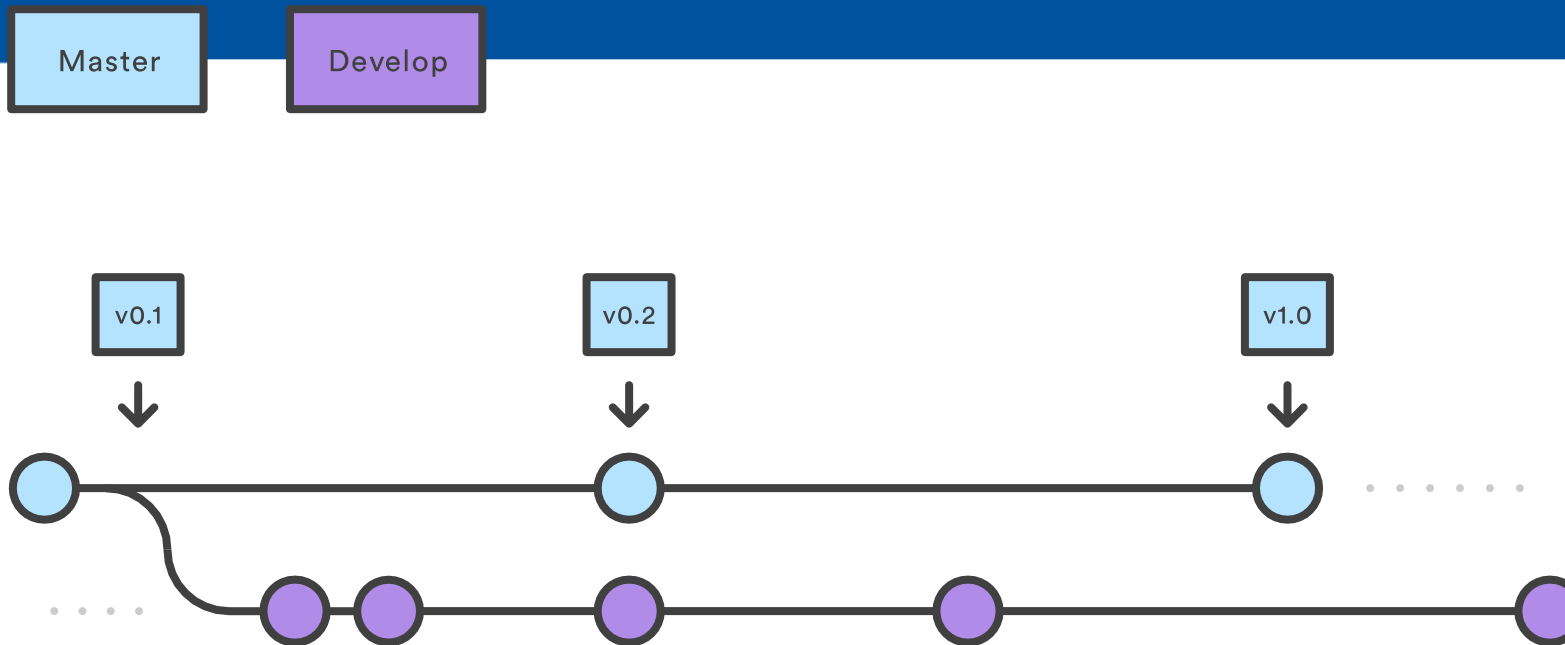
## Local Operations



<https://git-scm.com/book/de/v1/Los-geht%E2%80%99s-Git-Grundlagen>

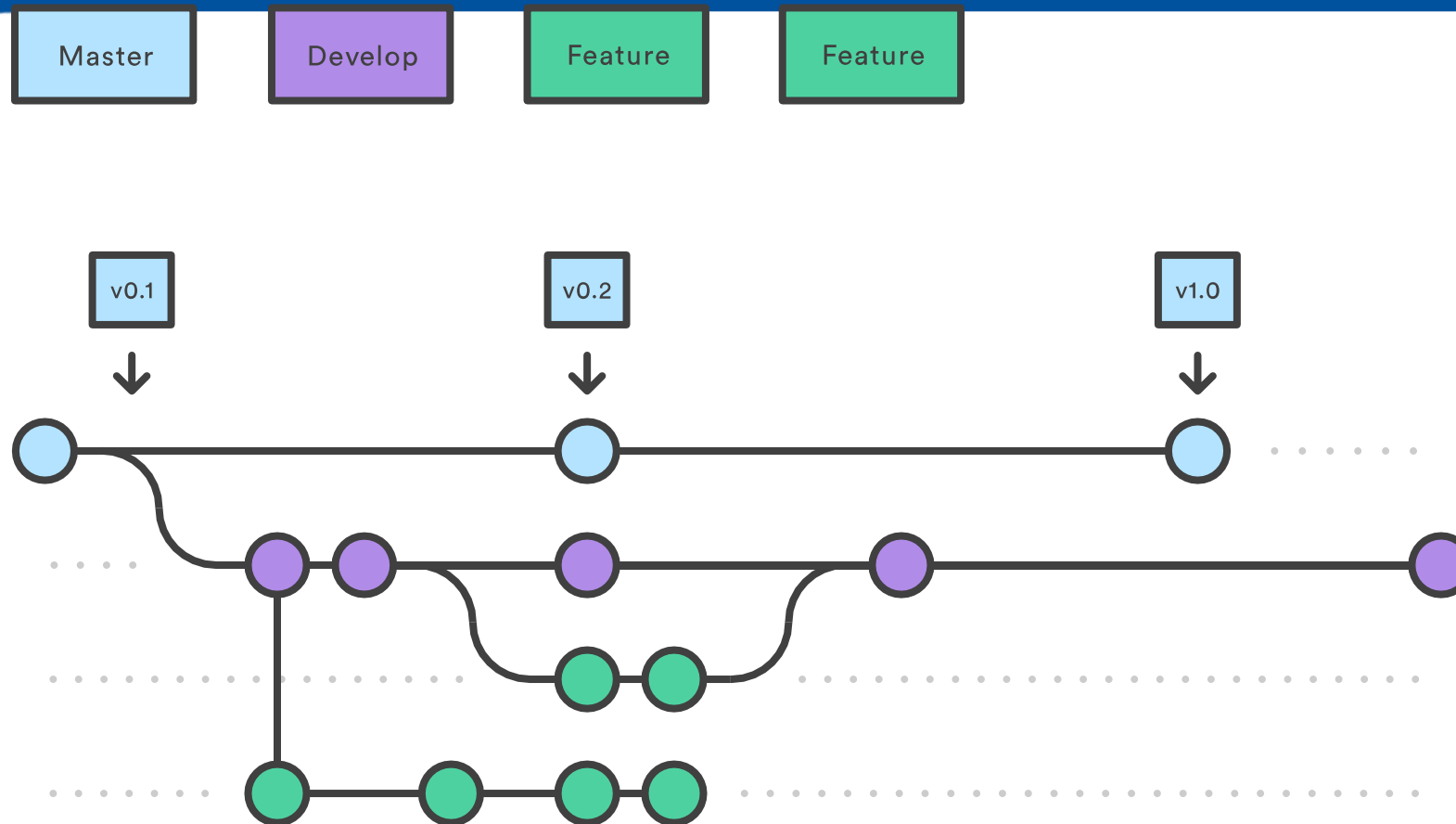
- Ein wichtiges Feature von git sind **Branches** (die im Gegensatz zu SVN auch nicht teuer sind)
- Man arbeitet **immer** auf einem Branch
- In dem SWP Repository gibt es immer:
  - Master:
    - Dieses ist der Hauptbranch, in dem am Ende auch das zu Bewertende liegen muss
    - muss immer korrekt und vollständig sein
    - Muss immer baubar sein
    - Niemand darf in den Master pushen
  - Develop
    - Dieses ist der Entwicklungszweig
    - Sollte die meiste Zeit korrekt sein
    - Niemand darf in den Development Branch pushen
  - Feature Branch(es)
    - Für jede Aufgabe wird ein eigener Branch, von Develop abgezweigt (abgebrancht)
  - Anm.: Master und Develop sind im SWP durch Bitbucket geschützt (man kann auch aus Versehen nichts falsch machen ;-))

- Wie kommen dann Änderungen in Develop bzw. Master?
  - Antwort: **Pull Request (PR)**: Ein PR ist die Anfrage, die Änderungen des eines (Ausgangs-)Branches in einen (Ziel-)Branch zu übernehmen
  - Ein Pull Request besitzt immer eine Review-Phase, in der **andere** die Änderungen überprüfen und den PR kommentieren, **akzeptieren** oder ggf. **zurückweisen**
  - Ein positiv bescheinigter PR kann dann in den Zielbranch gemergt werden
  - Ein PR darf dabei nie zu Konflikten führen
  - Konflikt:
    - Es haben unterschiedliche Entwickler so am selben Code gearbeitet, dass git nicht mehr in der Lage ist zu entscheiden, welche der Änderungen übernommen werden sollen
    - Auflösen von Konflikten ist oft mühsame Handarbeit
    - Deswegen: Regelmäßig pushen! Immer vor dem PR develop in den eigenen Feature Branch mergen
- Unterschiedliche Arten, wie mit Branches umgegangen wird → Im SWP Gitflow



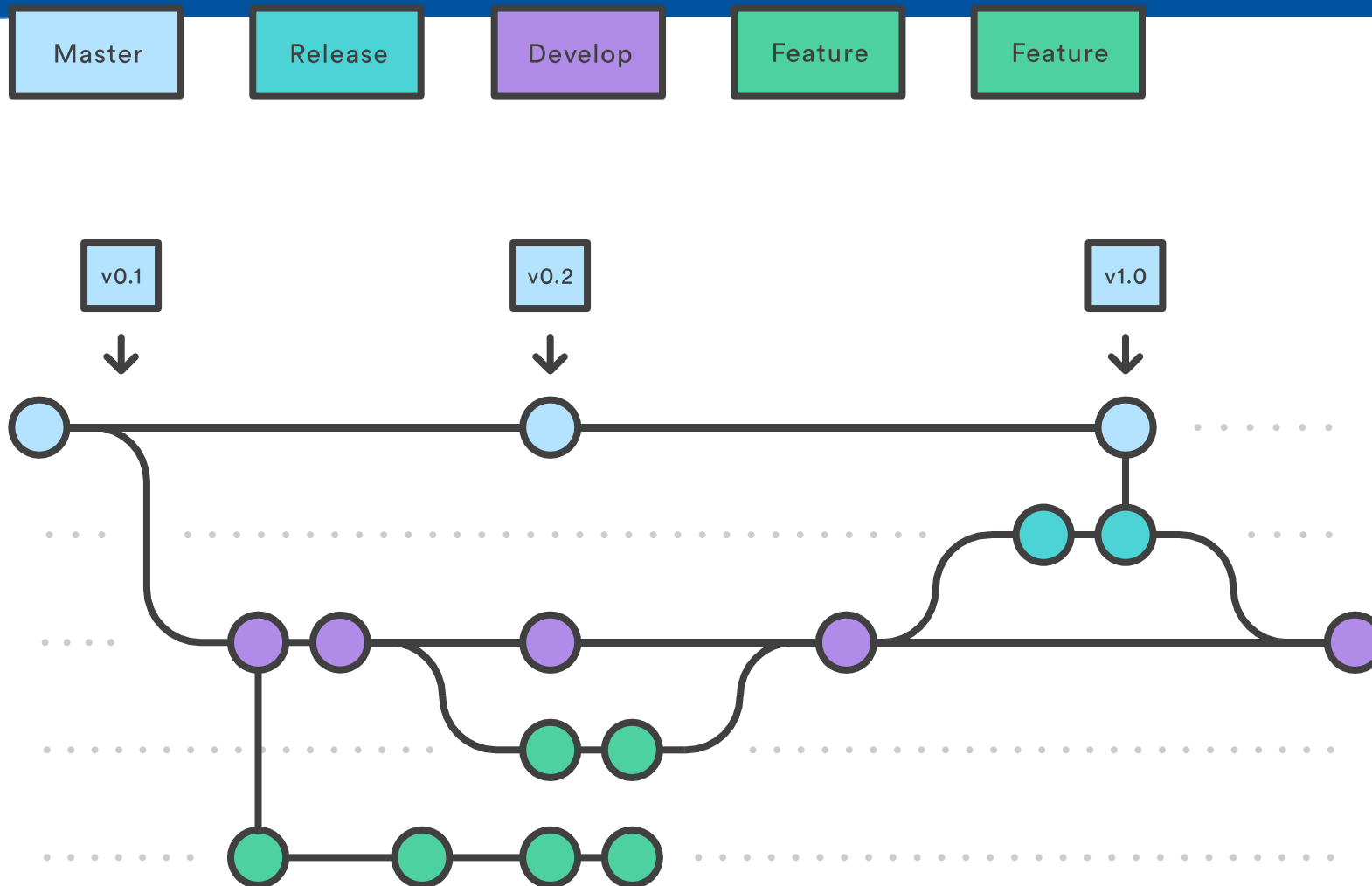
- Master: Historie des Projektes (offizielle Release History)
- Develop: Integrationsbranch für Feature
- Feature Branches:
  - Jedes neue Feature in eigenem Branch
  - Ausgehend von Develop
  - Wenn fertig: Merge in Develop (Feature werden nie in master gemergt!)
  - Develop wird regelmäßig in Master gemergt

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>



- Feature Branches

- Von Develop: Branch
- Nach develop: PR
- Nicht zu groß werden lassen! Es muss für jeden PR einen Review geben!

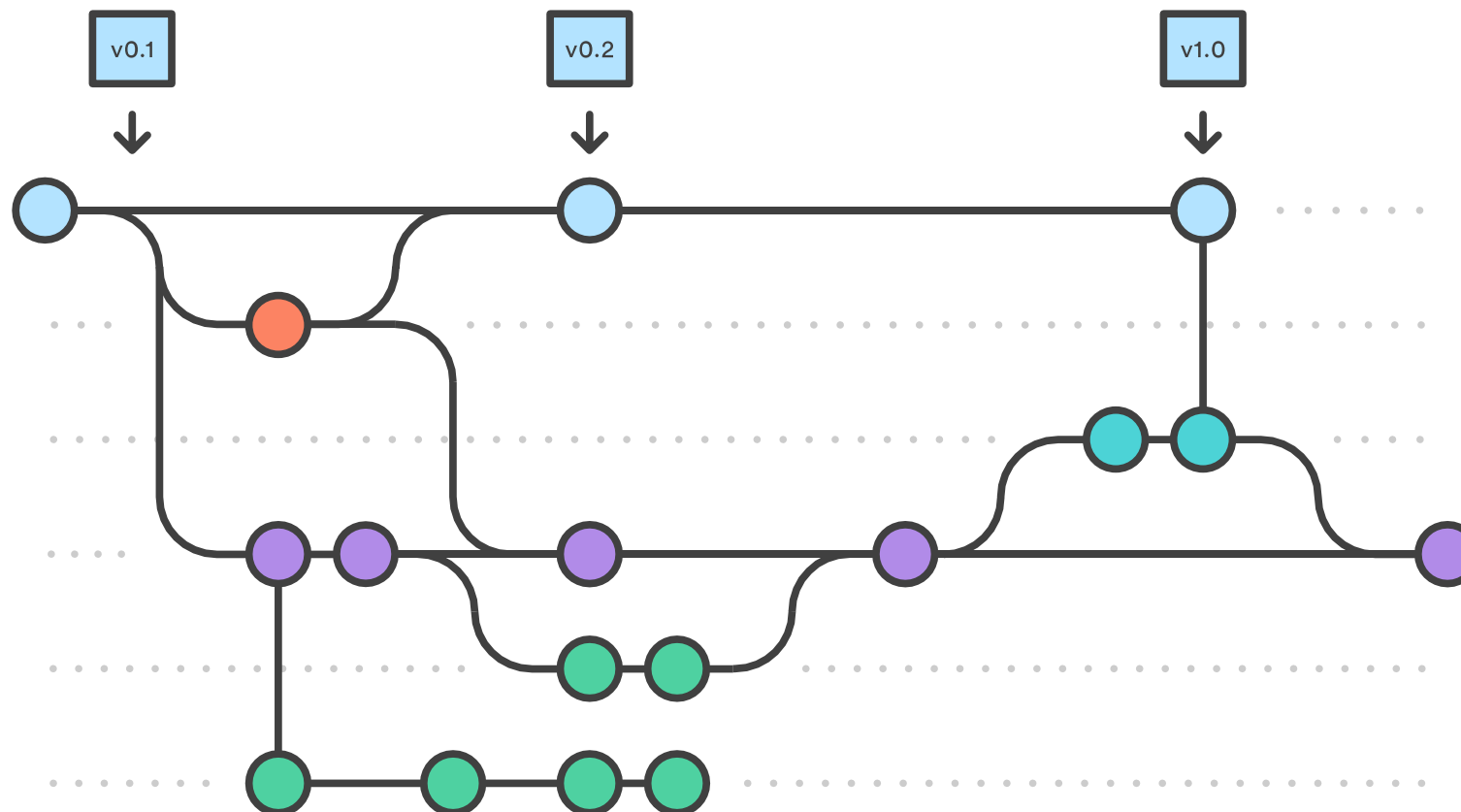


Release Branch  
für das SWP nicht  
notwendig

- Release Branches

- Statt direkt von develop nach master zu mergen
- Man kann parallel auf develop an weiteren Feature arbeiten, und kann den Release testen





Es wird ein Fehler im Master gefunden:  
Dann Hotfix-Branch erstellen

- Zum Beheben im Master
- Zum Nachpflegen im Develop

- Zusammenarbeit mit git
- Jede Änderung in einem Feature branch
- Verzweige develop von master ab
- Feature werden von develop „abgebrancht“
- Wenn Feature fertig, wird es in develop gemergt
- Wenn develop stabil → in master mergen
- Wenn ein Problem auf dem master erkannt wird:
  - Hotfix-Branch vom master
  - Nach Abschluss: Mergen in master **und** develop
- Weitere Infos:
  - <https://rogerdudler.github.io/git-guide/index.de.html>
  - <https://www.embedded-software-engineering.de/git-tutorial-git-und-die-wichtigsten-befehle-kennenlernen-a-725074/>
  - [http://pi.informatik.uni-siegen.de/lehre/Vertiefungspraktika/2017s/materials/slides/2017s\\_PEP\\_slides\\_git.pdf](http://pi.informatik.uni-siegen.de/lehre/Vertiefungspraktika/2017s/materials/slides/2017s_PEP_slides_git.pdf)

- Zurück zum Ticket:

**SWP\_BASESYSTEM**

SWP\_BASESYSTEM / SWPBASE-1 Als Nutzer möchte ich mich beim Spiel einloggen können, damit ich mit anderen Nutzen chatten und spielen kann. / SWPBASE-4

Erstelle Kommunikationsobjekte

Edit Comment Assign More Closed Start Admin

**Details**

Type: Sub-task Status: **TO DO** (View Workflow)  
Priority: Major Resolution: Unresolved  
Labels: None

**Description**

Click to add description

**Attachments**

Drop files to attach, or

**People**

Assignee: Unassigned  
Assign to me  
Reporter: Marco Grawunder  
Votes: 0  
Watchers: 1 Stop watching this issue

**Dates**

Created: 2 hours ago  
2 hours ago

**Tempo**

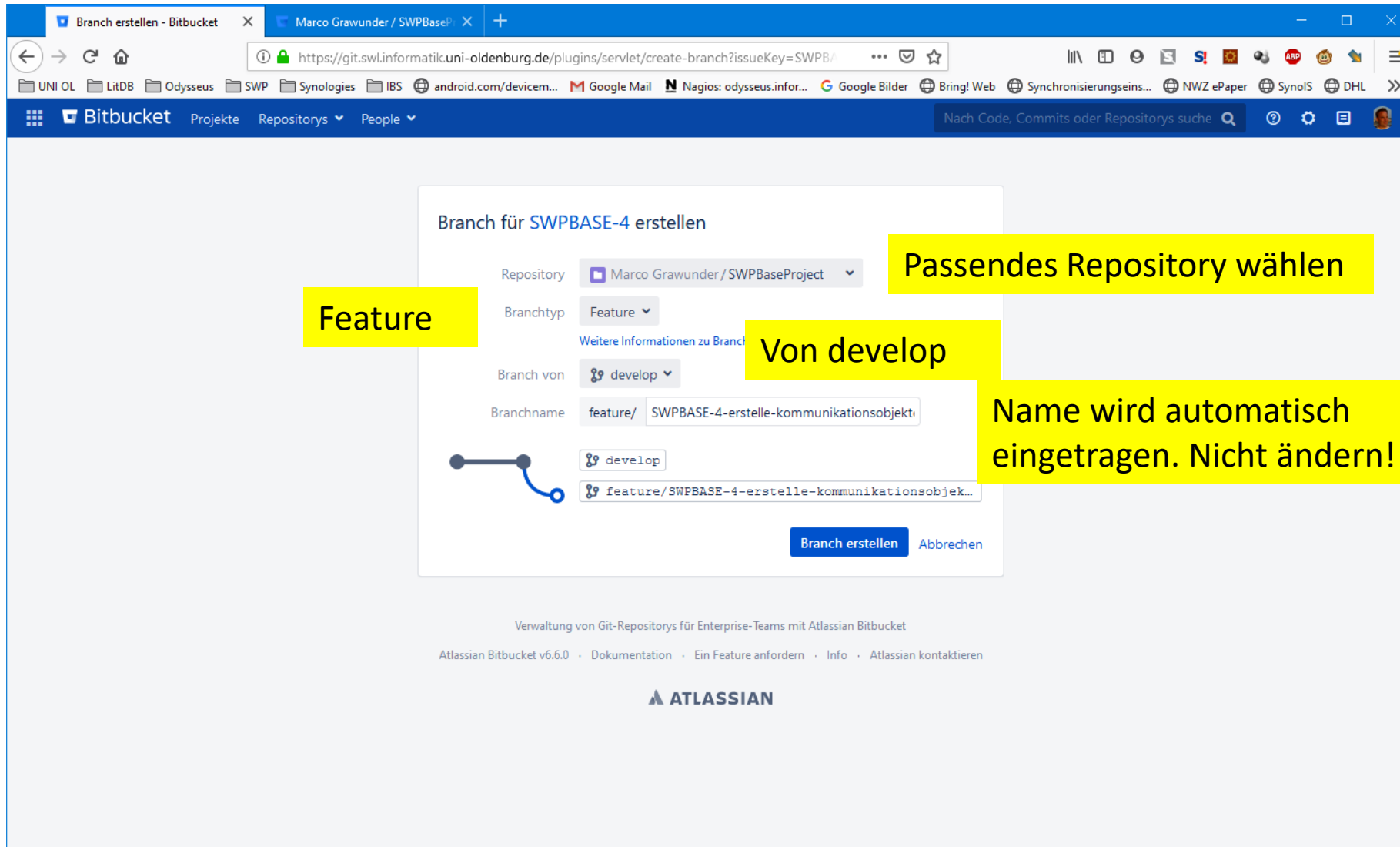
THERE ARE NO ACTIVITIES FOR YOU TO SEE FOR THIS ISSUE.  
You have not recorded any activity on this issue.

Log Time Plan Time Add Expense

**Development**  
Create branch

**Agile**  
View on Board

Project settings



Branch erstellen - Bitbucket

Marco Grawunder / SWPBaseProject

https://git.swl.informatik.uni-oldenburg.de/plugins/servlet/create-branch?issueKey=SWPB/

UNI OL LitDB Odysseus SWP Synologies IBS android.com/devicem... Google Mail Nagios: odysseus.infor... Google Bilder Bring! Web Synchronisierungseins... NWZ ePaper SynoIS DHL

Bitbucket Projekte Repositories People

Nach Code, Commits oder Repositories suche

### Branch für SWPBASE-4 erstellen

Repository: **Marco Grawunder / SWPBaseProject**

Branchtyp: **Feature**

Weitere Informationen zu Branch

Branch von: **develop**

Branchname: **feature/ SWPBASE-4-erstelle-kommunikationsobjekt**

Branch erstellen Abbrechen

Verwaltung von Git-Repositorys für Enterprise-Teams mit Atlassian Bitbucket

Atlassian Bitbucket v6.6.0 · Dokumentation · Ein Feature anfordern · Info · Atlassian kontaktieren

ATLASSIAN

**Feature**

**Passendes Repository wählen**

**Von develop**

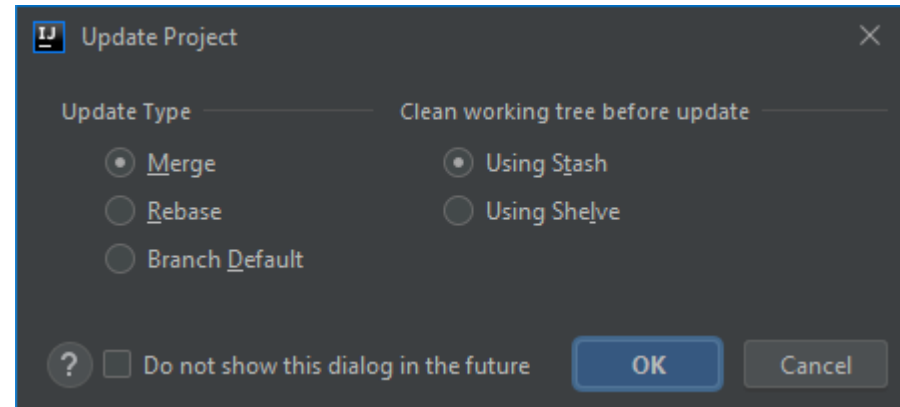
**Name wird automatisch eingetragen. Nicht ändern!**

The screenshot shows a Bitbucket web interface for a repository named 'SWPBaseProject' by Marco Grawunder. The browser address bar shows the URL: `https://git.swl.informatik.uni-oldenburg.de/projects/MG/repos/swpbaseproject/browse?at=`. The repository statistics show 72 commits, 3 branches, 0 pull requests, 2 contributors, 1 fork, and 0 tags. A red rectangle highlights the 'Dateien' (Files) section, which contains a dropdown menu showing 'feature/SWPBASE-4-erstelle-k...' and a search bar. Below this, a table lists the repository's files and folders:

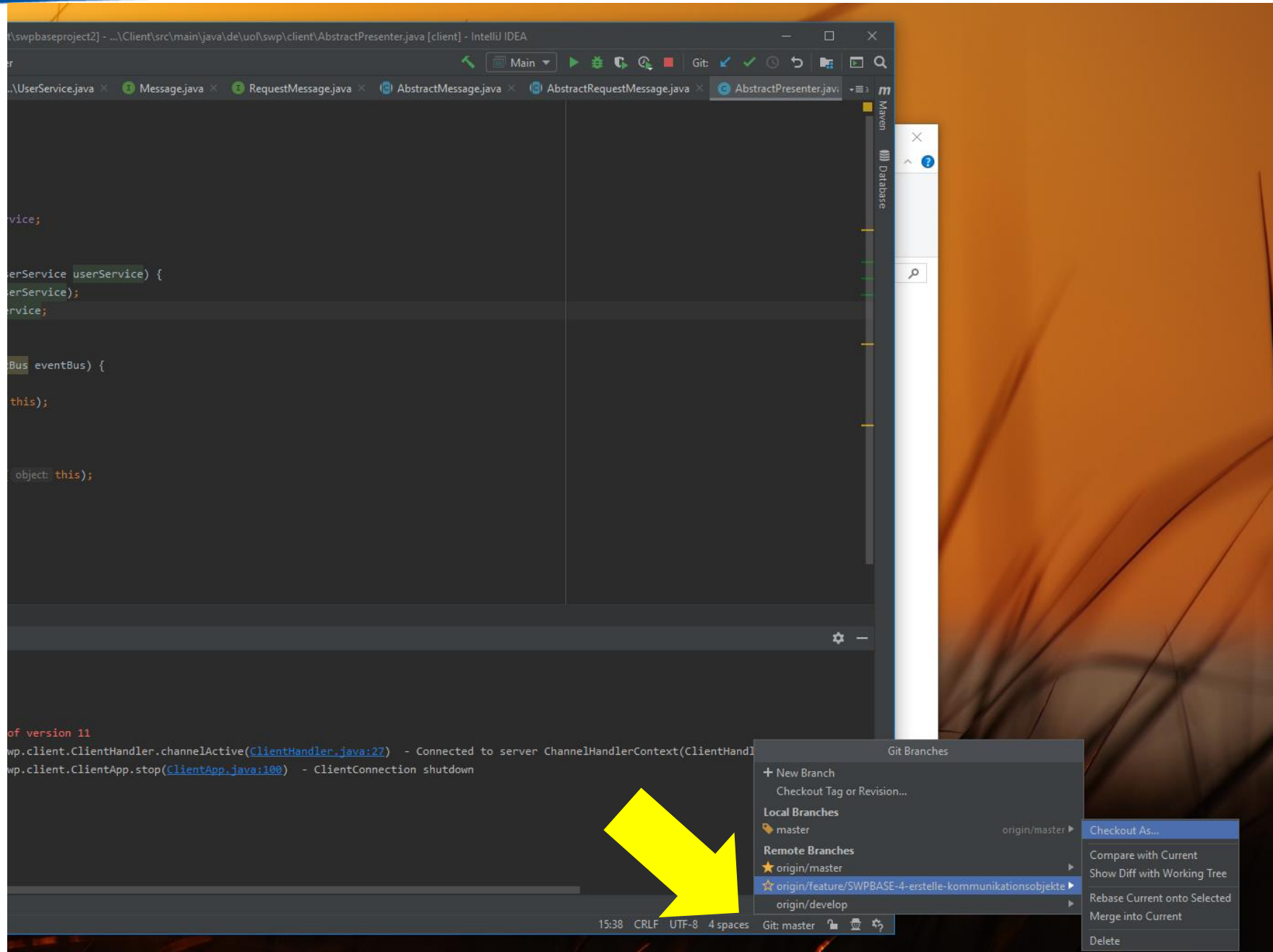
Quelle	Beschreibung	Letzte Änderung
Client		
Common		
Server		
.gitignore	Ignore .idea	16 Aug. 2019
pom.xml	Package with maven and start with java -jar	Vor 3 Tagen
Readme.MD	Readme for maven	Vor 3 Tagen

Below the table, there is a section for 'Readme.MD' with the title 'SWP Basissystem'.

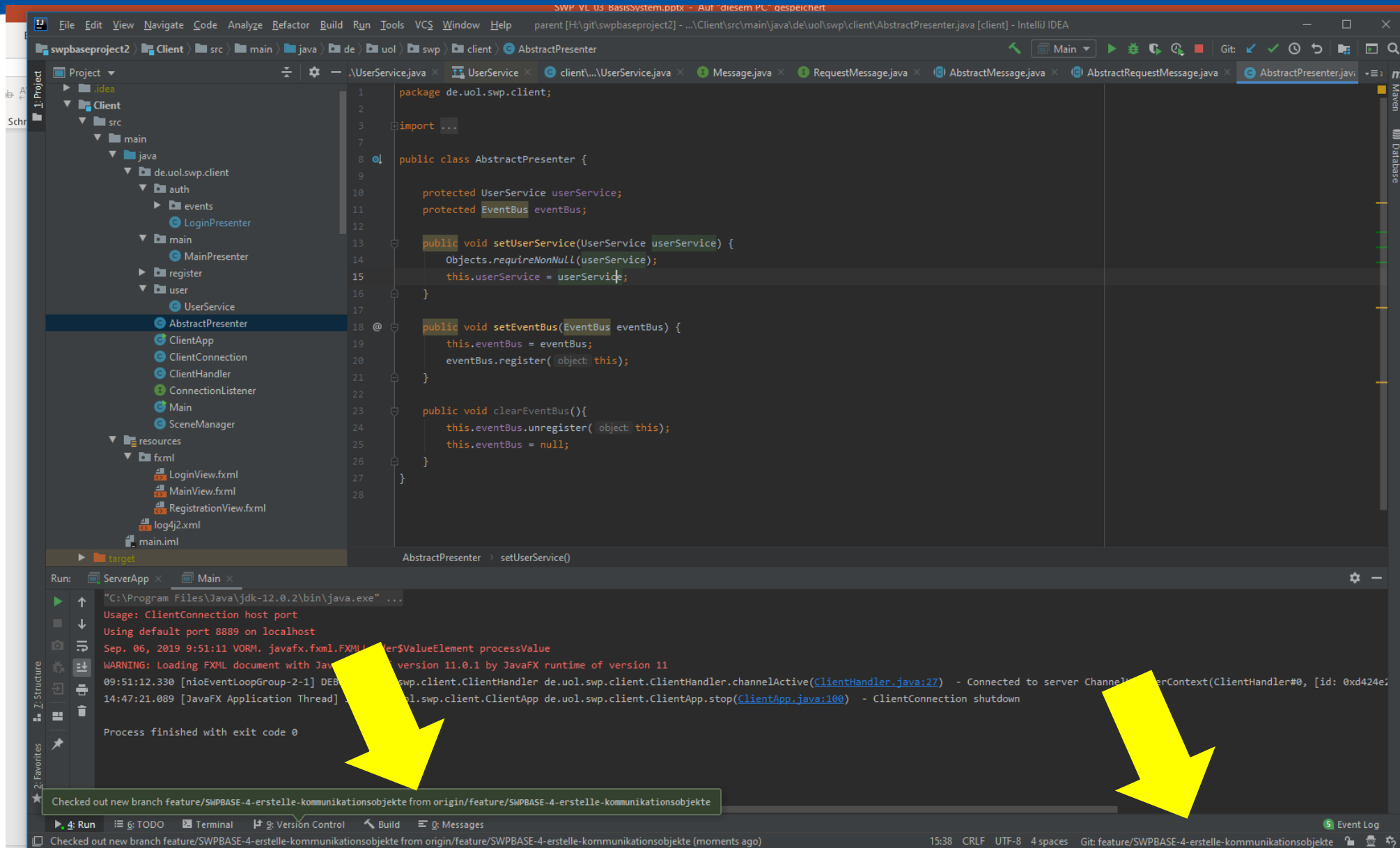
- Update Project (STRG-T)



# Neuen Branch auschecken



# Dann auf neuem Branch arbeiten



The screenshot shows the IntelliJ IDEA IDE with a project named 'swpbaseproject2'. The project structure on the left includes a 'Client' directory with 'src/main/java/de/uol/swp/client' containing 'AbstractPresenter.java'. The main editor displays the code for 'AbstractPresenter.java', which includes package declarations, imports, and methods for setting user service and event bus. The bottom panel shows the 'Run' tab with a log of the application execution, including warnings and messages. A green bar at the bottom indicates that a new branch 'feature/SWPBASE-4-erstelle-kommunikationsobjekte' has been checked out from the origin. Two large yellow arrows point from the 'Run' tab and the branch notification bar towards the bottom right corner.

```
package de.uol.swp.client;

import ...

public class AbstractPresenter {

    protected UserService userService;
    protected EventBus eventBus;

    public void setUserService(UserService userService) {
        Objects.requireNonNull(userService);
        this.userService = userService;
    }

    public void setEventBus(EventBus eventBus) {
        this.eventBus = eventBus;
        eventBus.register( object: this);
    }

    public void clearEventBus(){
        this.eventBus.unregister( object: this);
        this.eventBus = null;
    }
}
```

Run: ServerApp x Main x

"C:\Program Files\Java\jdk-12.0.2\bin\java.exe" ...

Usage: ClientConnection host port

Using default port 8889 on localhost

Sep. 06, 2019 9:51:11 VORM. javafx.fxml.FXMLLoader\$ValueElement processValue

WARNING: Loading FXML document with JavaFX version 11.0.1 by JavaFX runtime of version 11

09:51:12.330 [nioEventLoopGroup-2-1] DEBUG de.uol.swp.client.ClientHandler de.uol.swp.client.ClientHandler.channelActive(ClientHandler.java:27) - Connected to server ChannelHandlerContext(ClientHandler#0, [id: 0xd424e2...])

14:47:21.089 [JavaFX Application Thread] DEBUG de.uol.swp.client.ClientApp de.uol.swp.client.ClientApp.stop(ClientApp.java:100) - ClientConnection shutdown

Process finished with exit code 0

Checked out new branch feature/SWPBASE-4-erstelle-kommunikationsobjekte from origin/feature/SWPBASE-4-erstelle-kommunikationsobjekte

Run | TODO | Terminal | Version Control | Build | Messages | Event Log

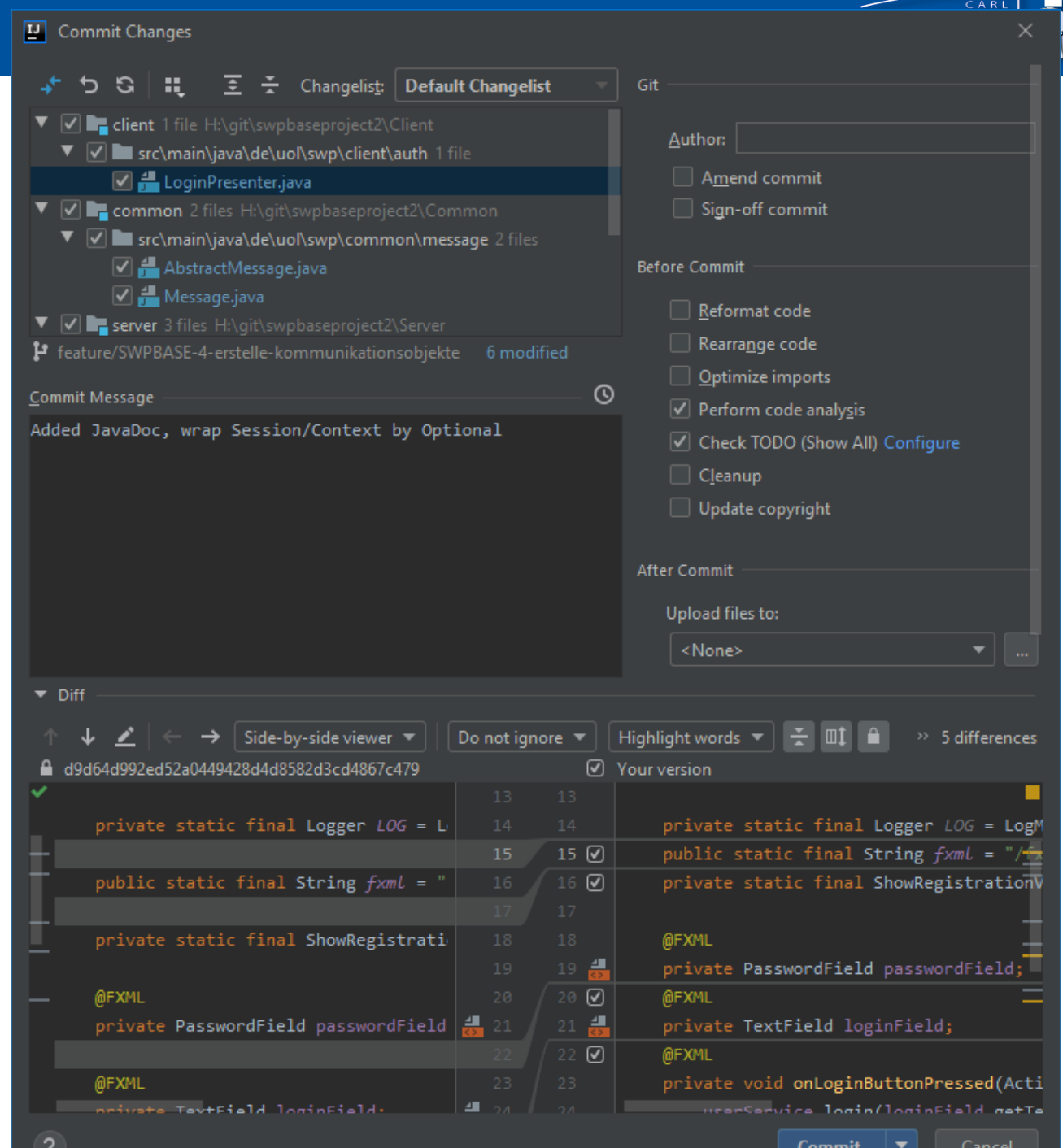
Checked out new branch feature/SWPBASE-4-erstelle-kommunikationsobjekte from origin/feature/SWPBASE-4-erstelle-kommunikationsobjekte (moments ago)

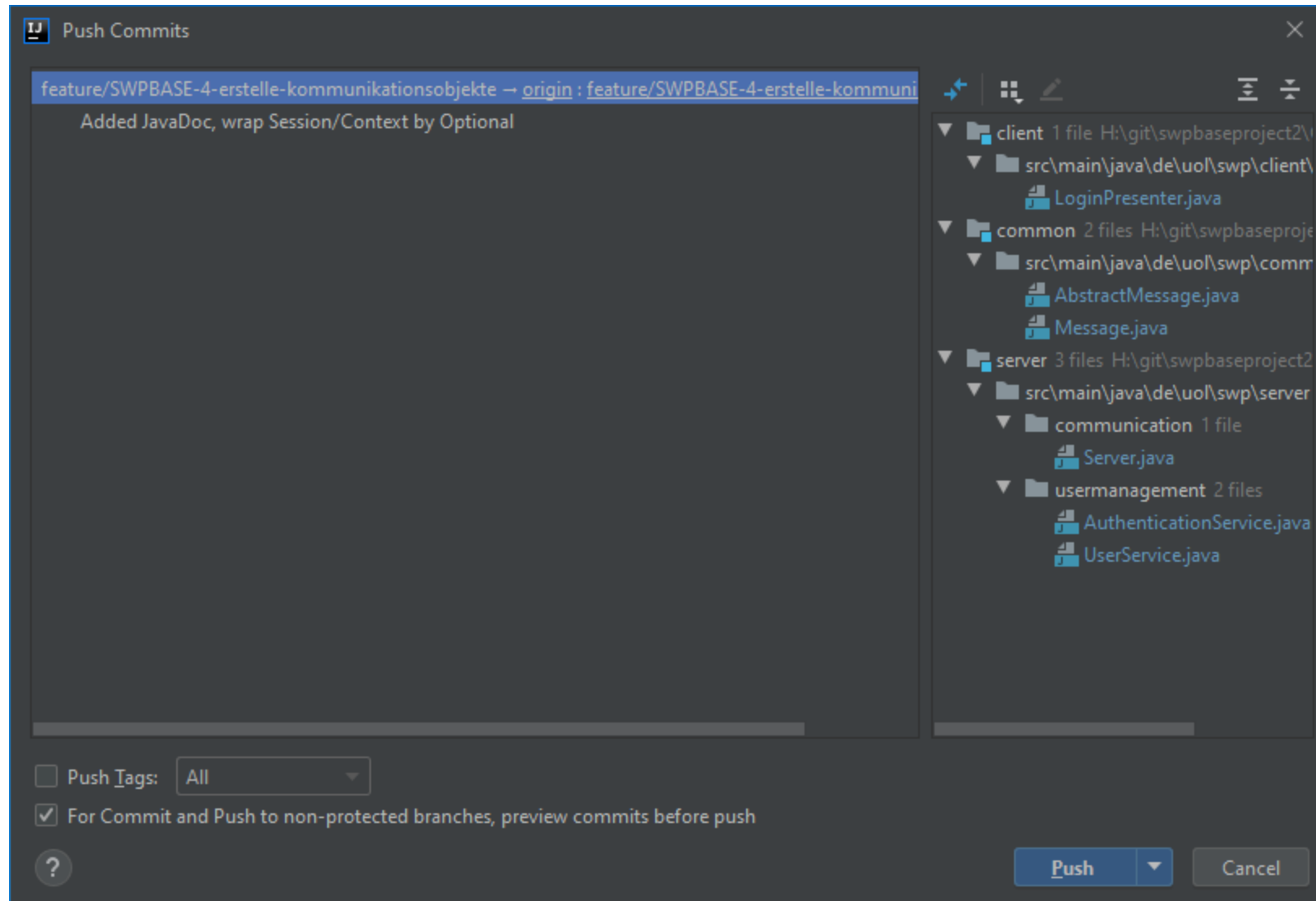
15:38 CRLF UTF-8 4 spaces Git: feature/SWPBASE-4-erstelle-kommunikationsobjekte

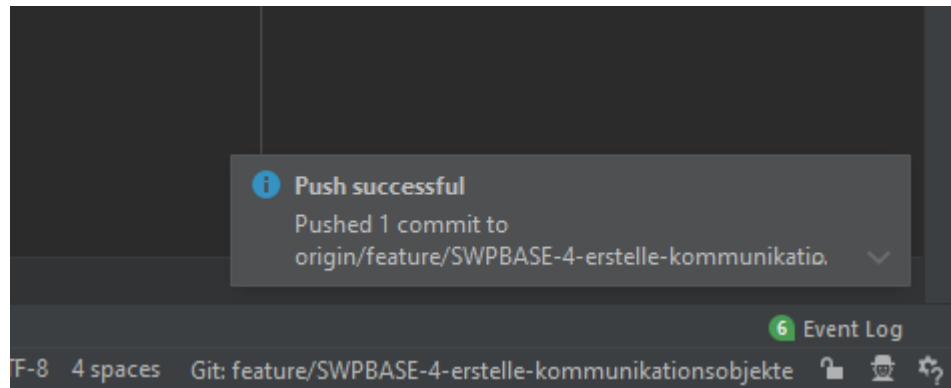


# Änderungen am Code machen

- Vor dem committen noch einmal aktualisieren und ggf. Konflikte beheben
- Änderungen committen
- und pushen
- Achtung: Default ist nur commit!
- Before commit:
  - Hier Code überprüft werden
  - Warnungen können vor dem Commit behandelt werden







Im anderen Fall gibt es u.U. Merge-Konflikte, die erst behoben werden müssen

Bitbucket Projekte Repositories People

Nach Code, Commits oder Repositories suchen

Marco Grawunder / SWPBaseProject

## Commits

feature/SWPBASE-4-erstelle-k... Show all

Autor	Commit ID	Kommentar	Commit-Datum
Marco Grawunder	f605ec850d7	Added JavaDoc, wrap Session/Context by Optional	Vor 2 Minuten
Marco Grawunder	d9d64d992ed	Renamed subscribe methods with "on" pattern	Vor 3 Tagen

feature/SWPBASE-4-erstelle-kommunikationsobjekte master develop

Anschließend Pull Request (PR) erstellen

Bitbucket Projekte Repositories People

Nach Code, Co

Marco Grawunder / SWPBaseProject


## Commits






feature/SWPBASE-4-erstelle-k... Show all
















Pull Request erstellen

Autor	Commit ID	Kommentar
Marco Grawunder	f605ec850d7	Added JavaDoc, wrap Session/Context by Optional
Marco Grawunder	d9d64d992ed	Renamed subscribe methods with "on" pattern
Marco Grawunder	bbc68fa6e2a	Removed empty session
Marco Grawunder	0183cd8ba07	Refactorings, renamed Session to UUIDSession
Marco Grawunder	8c138a9fde8	Readme for maven
Marco Grawunder	3cc35180294	Readme for maven

feature/SWPBASE-4-er


 **Bitbucket** Projekte Repositories People

Nach Code, Commits oder Repositories suchen     


              


## Pull Request erstellen

Arbeiten Sie beim Code zusammen, indem Sie Teammitglieder auswählen, die Ihre Änderungen an einem Branch überprüfen.




Quelle


Marco Grawu... / SWPBaseProj... 

feature/SWPBASE-4-erstelle-kommunikations... 

Marco Grawunder hat Commit [f605ec850d7](#) erstellt Vor 5 Minuten

Ziel

Marco Grawu... / SWPBaseProj... 


develop 

Marco Grawunder hat Commit [d9d64d992ed](#) erstellt Vor 3 Tagen

Fortsetzen


Diff

Commits

Autor	Commit ID	Kommentar	Commit-Datum
 Marco Grawunder	<a href="#">f605ec850d7</a>	Added JavaDoc, wrap Session/Context by Optional	Vor 5 Minuten

Verwaltung von Git-Repositories für Enterprise-Teams mit Atlassian Bitbucket

Atlassian Bitbucket v6.6.0 · [Dokumentation](#) · [Ein Feature anfordern](#) · [Info](#) · [Atlassian kontaktieren](#)

 **ATLASSIAN**

UML - M How to: Basic C [ODY-] [ODY-] karteil Git Fe: 02 (2): studi: Carl v: Depart: studi: Pul X +

git.swl.informatik.uni-oldenburg.de/projects/MG/repos/swpbaseproject/pull-requests?create&targetBranch=refs%2Fhe...

Apps UNI OL LitDB Odysseus SWP OFFIS-Telefonliste Nagios: odysseus.in... android.com/device... Google Mail Google Bilder Bring! Web

Bitbucket Projekte Repositories People Nach Code, Commits oder Repositories suche

### Pull Request erstellen

SWP... feature/SWPBASE-4-erstell... → SWPBasePr... develop Ändern

Überschrift\* Added JavaDoc, wrap Session/Context by Optional

Beschreibung

Prüfer admin

Prüfer können einen Pull Request genehmigen, um anderen mitzuteilen, wann ein Merge ausgeführt werden kann

Erstellen Abbrechen

Diff Commits

Autor	Commit ID	Kommentar	Commit-Datum
Marco Grawunder	f605ec850d7	Added JavaDoc, wrap Session/Context by Optional	Vor 5 Minuten

Verwaltung von Git-Repositories für Enterprise-Teams mit Atlassian Bitbucket

Atlassian Bitbucket v6.6.0 · Dokumentation · Ein Feature anfordern · Info · Atlassian kontaktieren

UML - M How to Basic C [ODY-] [ODY-] karteik Git Fe 02 (2) studi Carl v Depart studi Pul x +

git.swl.informatik.uni-oldenburg.de/projects/MG/repos/swpbaseproject/pull-requests/1/overview

Apps UNI OL LitDB Odysseus SWP OFFIS-Telefonliste Nagios: odysseus.in... android.com/device... Google Mail Google Bilder Bring! Web

Bitbucket Projekte Repositories People

Nach Code, Commits oder Repositories suchen ? ? ? ?

Marco Grawunder feature/SWPBASE-4-erstelle-kommunikationsobjekte → develop OFFEN Mergen ...

### Added JavaDoc, wrap Session/Context by Optional

Überblick Diff Commits

#### Details

Marco Grawunder hat den Pull Request erstellt Gerade eben  
Keine Beschreibung für diesen Pull Request. [Eine hinzufügen](#)

SWPBASE-4  
[Weitere Informationen](#)

#### Aktivität

Was möchten Sie sagen?

Marco Grawunder hat den Pull Request **GEÖFFNET** Gerade eben

Verwaltung von Git-Repositories für Enterprise-Teams mit Atlassian Bitbucket

Atlassian Bitbucket v6.6.0 · [Dokumentation](#) · [Ein Feature anfordern](#) · [Info](#) · [Atlassian kontaktieren](#)

ATLASSIAN

# Jemand anders kann sich nun den PR anschauen


The screenshot shows a web browser window displaying a pull request diff for the file `Message.java`. The browser's address bar shows the URL `git.swl.informatik.uni-oldenburg.de/projects/MG/repos/swpbaseproject/pull-requests/1/diff#Common/src/main/java/d...`. The browser's toolbar includes various icons for navigation and search. The left sidebar of the web interface shows a file tree with the following structure:












- Client/src/main/java/de/uol/swp/client/auth
  - LoginPresenter.java
- Common/src/main/java/de/uol/swp/common/message
  - AbstractMessage.java
  - Message.java
- Server/src/main/java/de/uol/swp/server
  - communication
    - Server.java
  - usermanagement
    - AuthenticationService.java
    - UserService.java

The main content area displays the diff for `Message.java`. The diff shows changes to the `Message` interface, which extends `Serializable`. The changes include:

- Line 16: `/**`
- Line 17:  `* Allows to set a MessageContext, e.g. for network purposes`
- Line 18:  `* @param messageContext`
- Line 19:  `*/`
- Line 20: `void setMessageContext(MessageContext messageContext);`
- Line 21: `MessageContext getMessageContext();`
- Line 22: `/**`
- Line 23:  `* Retrieve the current message context`
- Line 24:  `* @return`
- Line 25:  `*/`
- Line 26: `Optional<MessageContext> getMessageContext();`
- Line 27: `/**`
- Line 28:  `* Set the current session`
- Line 29:  `* @param session`
- Line 30:  `*/`
- Line 31: `void setSession(Session session);`
- Line 32: `Session getSession();`
- Line 33: `/**`
- Line 34:  `* Retrieve current session`
- Line 35:  `* @return`
- Line 36:  `*/`
- Line 37: `Optional<Session> getSession();`
- Line 38: `/**`
- Line 39:  `* Allow to create a new message, based on`
- Line 40:  `* the given one (copy)`
- Line 41:  `* @param otherMessage`
- Line 42:  `*/`
- Line 43: `void initWithMessage(Message otherMessage);`
- Line 44: `}`



 Bitbucket Projekte Repositories




feature/IBS-198-spiel-verlassen-wenn-tab-geschlossen → master **GEMERGT**


Feature/IBS-198 spiel verlassen wenn tab geschlossen


Überblick Diff Commits


Details

 hat den Pull Request erstellt 08 Mrz. 2017  
Beim Verlassen eines Spiels durch schließen des Tabs wird der Spieler nun vom Server aus dem Spiel entfernt.  
Zudem wird das Spiel clientseitig vom EventBus unregistriert.


Aktivität


 Was möchten Sie sagen?


 hat einen manuellen Merge des Pull Requests von feature/IBS-198-spiel-verlassen-wenn-tab-geschlossen nach master ausgeführt Vor 3 Tagen




Ich habe abgelehnt, da es zu kurzfristig vor der Präsentation ist. Nicht dass da noch irgendwo ein versteckter Bug sitzt und darauf wartet uns die Live Demo zu zerstören :D  
Ich werde den Donnerstag akzeptieren.

Antworten · Löschen · Task erstellen · Gefällt mir ·  Sören Gade gefällt das · Vor 6 Tagen

 hat den Pull Request als **BENÖTIGT NACHBESSERUNG** markiert Vor 6 Tagen

 hat den Pull Request als **GENEHMIGT** markiert Vor 6 Tagen

 hat eine Datei kommentiert Vor 6 Tagen in Commit-Umfang b603bd662de..b12e4e79fd9

client / src / main / resources / **ResourceBundle.properties**

```
68 68 Lobby = Lobby
69 69 Chat = Chat
70 70 #</editor-fold>
71 71
72 72 #<editor-fold desc="GamePresenterImpl">
73 73 SessionLeaveFailed = Session konnte nicht Verlassen werden
```

- Wenn dann nach einiger Zeit alle den PR genehmigt haben, kann er gemergt werden



















The screenshot shows the Bitbucket web interface for the repository `git.swl.informatik.uni-oldenburg.de/projects/MG/repos/swpbaseproject`. The page displays the commit history for the `develop` branch. The interface includes a sidebar with navigation icons, a top navigation bar with the Bitbucket logo and search bar, and a main content area with the commit list.

Commits

develop Show all

Autor	Commit ID	Kommentar	Commit-Datum	Vorgänge
Marco Grawunder	f948f3733a5 M	Merge pull request #1 in MG/swpbaseproject from feature/SV	Gerade eben	SWPBASE-4
Marco Grawunder	f605ec850d7	Added JavaDoc, wrap Session/Context by Optional	Vor 11 Minuten	



Marco Grawunder

feature/SWPBASE-6-Dependency-Injection → develop


OFFEN

Mergen

...


## Feature/SWPBASE-6 Dependency Injection


Überblick Diff Commits

 **Dieser Pull Request kann nicht gemergt werden.**  
Sie müssen Konflikte beheben, um zu mergen. [Weitere Informationen.](#)

Details


4 Jira-Vorgänge


 [Weitere Informationen](#)

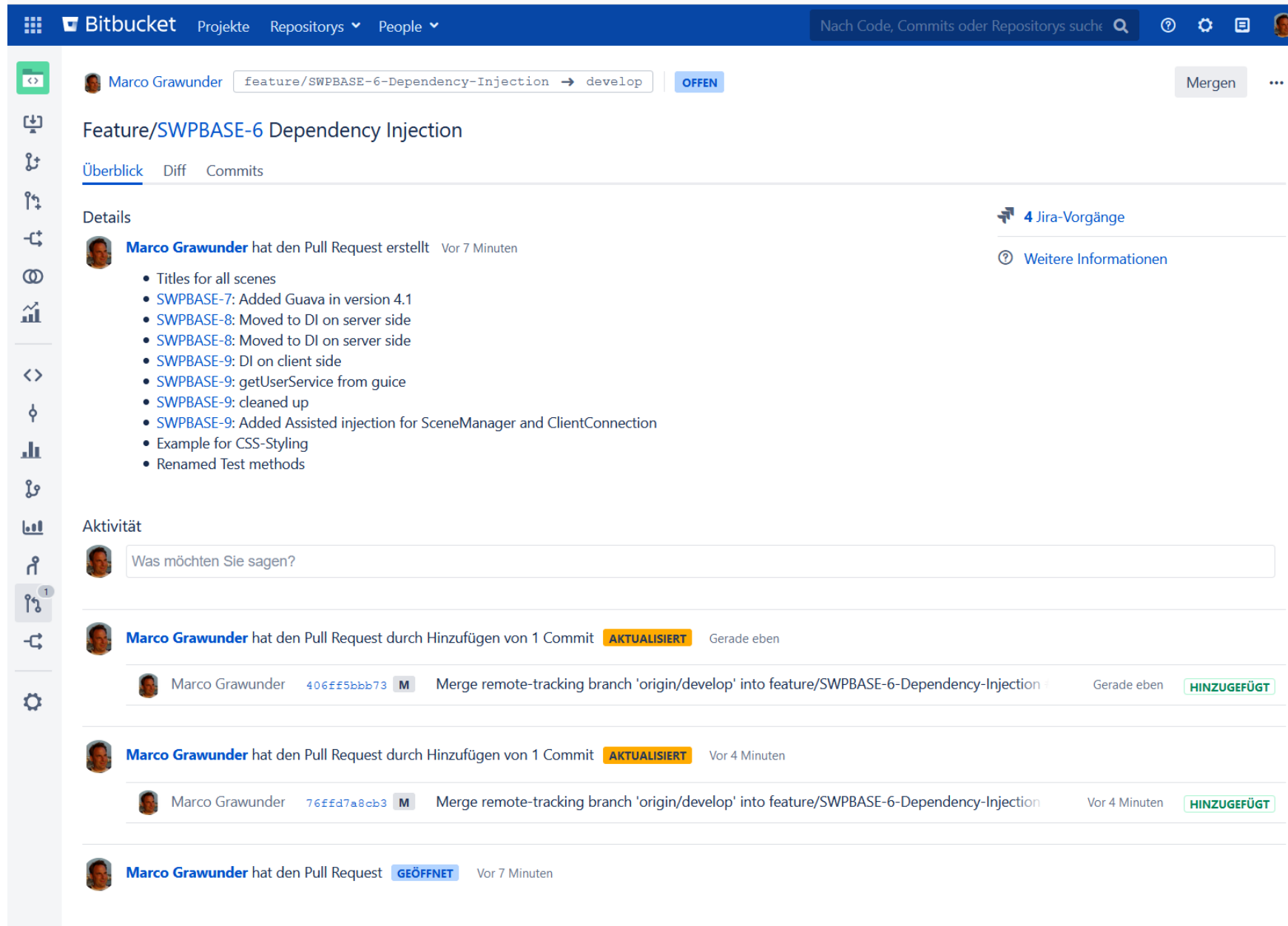
 **Marco Grawunder** hat den Pull Request erstellt Gerade eben

- Titles for all scenes
- [SWPBASE-7](#): Added Guava in version 4.1
- [SWPBASE-8](#): Moved to DI on server side
- [SWPBASE-8](#): Moved to DI on server side
- [SWPBASE-9](#): DI on client side
- [SWPBASE-9](#): getUserService from guice
- [SWPBASE-9](#): cleaned up
- [SWPBASE-9](#): Added Assisted injection for SceneManager and ClientConnection
- Example for CSS-Styling
- Renamed Test methods

Aktivität

 Was möchten Sie sagen?

 **Marco Grawunder** hat den Pull Request **GEÖFFNET** Gerade eben



The screenshot shows a Bitbucket pull request interface. At the top, the Bitbucket logo and navigation links (Projekte, Repositories, People) are visible. A search bar contains the text 'Nach Code, Commits oder Repositories suchen'. The main header shows the pull request title 'Feature/SWPBASE-6 Dependency Injection' with a status 'OFFEN' and a 'Mergen' button. Below the header, the 'Details' section shows the pull request was created by 'Marco Grawunder' 7 minutes ago. A list of changes is provided, including adding Guava, moving dependencies, and cleaning up. The 'Aktivität' section shows a comment 'Was möchten Sie sagen?' and two update events where the pull request was updated by adding commits, each marked 'HINZUGEFÜGT'. The bottom of the page shows the pull request was opened 7 minutes ago.

Bitbucket Projekte Repositories People

Nach Code, Commits oder Repositories suchen

Marco Grawunder feature/SWPBASE-6-Dependency-Injection → develop OFFEN Mergen

## Feature/SWPBASE-6 Dependency Injection

Überblick Diff Commits

### Details

Marco Grawunder hat den Pull Request erstellt Vor 7 Minuten

- Titles for all scenes
- SWPBASE-7: Added Guava in version 4.1
- SWPBASE-8: Moved to DI on server side
- SWPBASE-8: Moved to DI on server side
- SWPBASE-9: DI on client side
- SWPBASE-9: getUserService from guice
- SWPBASE-9: cleaned up
- SWPBASE-9: Added Assisted injection for SceneManager and ClientConnection
- Example for CSS-Styling
- Renamed Test methods

4 Jira-Vorgänge Weitere Informationen

### Aktivität

Was möchten Sie sagen?

Marco Grawunder hat den Pull Request durch Hinzufügen von 1 Commit AKTUALISIERT Gerade eben

Marco Grawunder 406f55bb73 M Merge remote-tracking branch 'origin/develop' into feature/SWPBASE-6-Dependency-Injection Gerade eben HINZUGEFÜGT

Marco Grawunder hat den Pull Request durch Hinzufügen von 1 Commit AKTUALISIERT Vor 4 Minuten

Marco Grawunder 76fd7a8cb3 M Merge remote-tracking branch 'origin/develop' into feature/SWPBASE-6-Dependency-Injection Vor 4 Minuten HINZUGEFÜGT

Marco Grawunder hat den Pull Request GEÖFFNET Vor 7 Minuten

- Bisher:
  - Ereignisgetriebene Kommunikation
  - Modellierung des Logins
  - Stories und Tasks in Jira
  - Eventbus
  - Git: Umgang mit Branches
- Als nächstes: Umsetzung des Clients

# Model-View-Presenter

Wie kann der Client möglichst gut erweiterbar und testbar gebaut werden?



## View

- Stellt das Modell dar
- Sendet Nutzeraktionen zum Presenter



## Presenter

- Definiert das Anwendungsverhalten
- Bildet Nutzeraktionen auf Modellaktualisierungen ab
- Steuert View



## Model

- Kapselt den Zustand der Anwendung
- Antwortet auf Zustandsanfragen
- Bietet Anwendungsfunktionalität
- Informiert über Änderungen

- Ein Entwurfsmuster
- Aus Model-View-Controller hervorgegangen
- Noch stärkere Trennung von View und Modell
- Unterschiedliche Varianten bei Aufteilung der Arbeit zwischen View und Presenter
- Siehe auch: <https://martinfowler.com/eaDev/PassiveScreen.html>

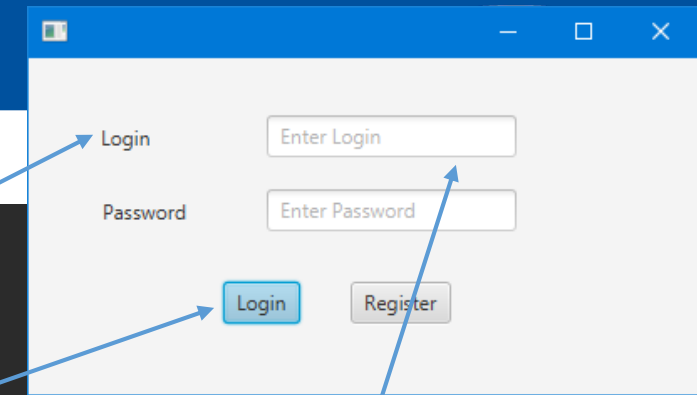


- Model-View-Presenter Muster
- Das Modell ist die eigentliche Anwendung (inkl. der „Geschäftslogik“)
- Der View ist „dumm“: Im SWP: Java-Fxml-Datei (z.B. LoginView.fxml)
- Es gibt einen Presenter (heißt in JavaFX Controller) der die Inhalte des Views in Aufrufe an das Modell übersetzt
- Presenter kennt den View
- Am Beispiel: Login:
  - Es gibt eine `LoginView.fxml`
  - Es gibt einen `LoginPresenter` (heißt hier in der VL noch `LoginHandler`)
  - Und es gibt auf der Server-Seite den `AuthenticationService`, das `UserManagement` und den `UserStore` (gehört alles zum Modell)
- Für MVP: Immer das obige Schema verwenden
- Ziel: View bleibt austauschbar, weiß nichts vom Modell

# Quellcode LoginView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.AnchorPane?>
<AnchorPane prefHeight="196.0" prefWidth="407.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="de.uol.swp.client.auth.LoginPresenter">
    <Label layoutX="43.0" layoutY="39.0" text="Login"/>
    <TextField id="loginField" fx:id="loginField" layoutX="142.0" layoutY="34.0" promptText="Enter Login">
        <tooltip>
            <Tooltip text="Enter your login here"/>
        </tooltip>
    </TextField>
    <PasswordField id="passwordField" fx:id="passwordField" layoutX="142.0" layoutY="78.0" promptText="Enter Password"/>
    <Button id="loginButton" defaultButton="true" layoutX="116.0" layoutY="133.0"
        mnemonicParsing="false" onAction="#onLoginButtonPressed" text="Login"/>
    <Button id="registerButton" layoutX="192.0" layoutY="133.0" mnemonicParsing="false"
        onAction="#onRegisterButtonPressed" text="Register"/>
    <Label layoutX="44.0" layoutY="83.0" text="Password"/>
</AnchorPane>
```



Anmerkung: Scenebuilder verwenden (<https://gluonhq.com/products/scene-builder/>)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.*?>
```

```
<?import javafx.scene.layout.*?>
```

Heißt hier Controller

```
<AnchorPane xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
```

```
fx:controller="de.uol.swp.client.auth.LoginPresenter">
```

```
<Label layoutX="42.0" layoutY="34.0" text="Login"/>
```

```
<TextField id="loginField" fx:id="loginField" layoutX="142.0" layoutY="34.0" promptText="Enter Login">
```

```
<tooltip>
```

```
<Tooltip text="Enter your login here"/>
```

```
</tooltip>
```

```
</TextField>
```

```
<PasswordField id="passwordField" fx:id="passwordField" layoutX="142.0" layoutY="78.0" promptText="Enter Password"/>
```

```
<Button id="loginButton" defaultButton="true" layoutX="110.0" layoutY="133.0"
```

```
mnemonicParsing="false" fx:id="loginButton" onAction="#onLoginButtonPressed" text="Login"/>
```

```
<Button id="registerButton" layoutX="152.0" layoutY="133.0" mnemonicParsing="false"
```

```
onAction="#onRegisterButtonPressed" text="Register"/>
```

```
<Label layoutX="44.0" layoutY="83.0" text="Password"/>
```

```
</AnchorPane>
```

Alle Elemente mit fx:id im Presenter zugreifbar

Methode im Presenter, die beim Drücken des Buttons aufgerufen wird

```
package de.uol.swp.client.auth;
```

```
import ...
```

fx:Controller

Eigene Basisklasse, nicht JavaFX

```
public class LoginPresenter extends AbstractPresenter {
```

```
    private static final Logger LOG = LogManager.getLogger(LoginPresenter.class);
```

```
    public static final String fxml = "/fxml/LoginView.fxml";
```

```
    private static final ShowRegistrationViewEvent showRegViewMessage = new ShowRegistrationViewEvent();
```

```
@FXML
```

```
private PasswordField passwordField;
```

```
@FXML
```

```
private TextField loginField;
```

```
@FXML
```

```
private void onLoginButtonPressed(ActionEvent event) {
```

```
    userService.login(loginField.getText(), passwordField.g
```

Verknüpfung über Annotationen, Namen sind identisch wie im FXML-File

Hier jetzt dafür sorgen, dass der LoginRequest erzeugt und an den Server gesendet wird

```
}
```

```
@FXML
```

```
private void onRegisterButtonPressed(ActionEvent event) {
```

```
    EventBus.post(showRegViewMessage);
```

```
}
```

```
}
```

- Jede View braucht passenden Presenter, Neue, andere View → neuer Presenter
- Presenter sind schwierig zu testen, da GUI enthalten (notwendig z.B. TestFX)
- deswegen
  - Presenter möglichst dumm halten
  - Logik in andere Klassen auslagern
  - Presenter ruft Logik in diesen Klassen auf
- Für den Login z.B. UserService,
  - Übersetzung von Nutzernamen und Passwort in das LoginRequest-Objekt
  - Legt LoginRequest auf den EventBus
- Nicht jeder Presenter muss exklusiven Service nutzen, z.B. RegistrationPresenter nutzt auch den UserService
- Später für Chat: ChatView → ChatPresenter → ChatService

```
package de.uol.swp.client.auth;

import ...

public class LoginPresenter extends AbstractPresenter {

    private static final Logger LOG = LogManager.getLogger(LoginPresenter.class);
    public static final String fxml = "/fxml/LoginView.fxml";
    private static final ShowRegistrationViewEvent showRegViewMessage = new ShowRegistrationViewEvent();

    @FXML
    private PasswordField passwordField;
    @FXML
    private TextField loginField;
    @FXML
    private void onLoginButtonPressed(ActionEvent event) {
        userService.login(loginField.getText(), passwordField.getText());
    }
    @FXML
    private void onRegisterButtonPressed(ActionEvent event) {
        EventBus.post(showRegViewMessage);
    }
}
```

Nur Weiterleitung an den User-Service

```
UserService  
login(String, String)    User  
isLoggedIn(User)         boolean  
logout(User)             void  
createUser(User)         User  
updateUser(User)         User  
retrieveAllUsers()       List<User>
```

Anmerkung: Nicht ganz aktuell ...

```
public class UserService implements de.uol.swp.common.user.UserService {  
  
    private static final Logger LOG = LogManager.getLogger(UserService.class);  
    private final EventBus bus;  
  
    public UserService(EventBus bus) {  
        this.bus = bus;  
    }  
  
    @Override  
    public User login(String username, String password){  
        LoginRequest msg = new LoginRequest(username, password);  
        bus.post(msg);  
        return null; // asynch call  
    }  
}
```



- Der Presenter wird von JavaFX automatisch erstellt (FXMLLoader)
- Wie bekommt man nun den User-Service und den EventBus in den Presenter hinein?
- Abstrakte Basisklasse: AbstractPresenter mit settern (→ später anders)

```
public class AbstractPresenter {  
  
    protected UserService userService;  
    protected EventBus eventBus;  
  
    public void setUserService(UserService userService) {  
        Objects.requireNonNull(userService);  
        this.userService = userService;  
    }  
  
    public void setEventBus(EventBus eventBus) {  
        this.eventBus = eventBus;  
        eventBus.register(this);  
    }  
  
    public void clearEventBus(){  
        this.eventBus.unregister(this);  
        this.eventBus = null;  
    }  
}
```

```
public class SceneManager {  
  
    static final Logger LOG = LogManager.getLogger(SceneManager.class);  
    static final String styleSheet = "css/swp.css";  
  
    final private Stage primaryStage;  
    final private EventBus eventBus;  
    final private UserService userService;  
    private Scene loginScene;  
    private String lastTitle;  
    private Scene registrationScene;  
    private Scene mainScene;  
    private Scene lastScene = null;  
    private Scene currentScene = null;  
  
    private User currentUser;  
  
    private Injector injector;  
  
    @Inject  
    public SceneManager(EventBus eventBus, UserService userService,  
        Injector injected, @Assisted Stage primaryStage) {  
        this.eventBus = eventBus;  
        this.eventBus.register(this);  
        this.userService = userService;  
        this.primaryStage = primaryStage;  
        this.injector = injected;  
        initView();  
    }  
}
```

- SceneManager kennt alle Szenen (könnte man generischer machen)
- Initialisiert alle Views und Presenter
- Presenter bekommen EventBus und UserService zugewiesen
- Kennt immer die letzte Szene
- Bietet Möglichkeiten zum Wechseln einer Szene (oder einmalig zur letzten Szene)
- @Inject/@Assisted → Guava, VL Dependency Injection

```
private void initMainView() {
    if (mainScene == null) {
        Parent rootPane = initPresenter(MainMenuPresenter.fxml);
        mainScene = new Scene(rootPane, 800, 600);
        mainScene.getStylesheets().add(styleSheet);
    }
}

private void initLoginView() {
    if (loginScene == null) {
        Parent rootPane = initPresenter(LoginPresenter.fxml);
        loginScene = new Scene(rootPane, 400, 200);
        loginScene.getStylesheets().add(styleSheet);
    }
}

private void initRegistrationView(){
    if (registrationScene == null){
        Parent rootPane = initPresenter(RegistrationPresenter.fxml);
        registrationScene = new Scene(rootPane, 400,200);
        registrationScene.getStylesheets().add(styleSheet);
    }
}
```

```
private Parent initPresenter(String fxmlFile) {  
    Parent rootPane;  
    FXMLLoader loader = new FXMLLoader(getClass().getResource(fxmlFile));  
    try {  
        rootPane = loader.load();  
    } catch (IOException e) {  
        throw new RuntimeException("Could not load View!");  
    }  
    AbstractPresenter presenter = loader.getController();  
    presenter.setEventBus(eventBus);  
    presenter.setUserService(userService);  
    return rootPane;  
}
```

- ...auf dem EventBus umschalten zwischen den Scenes

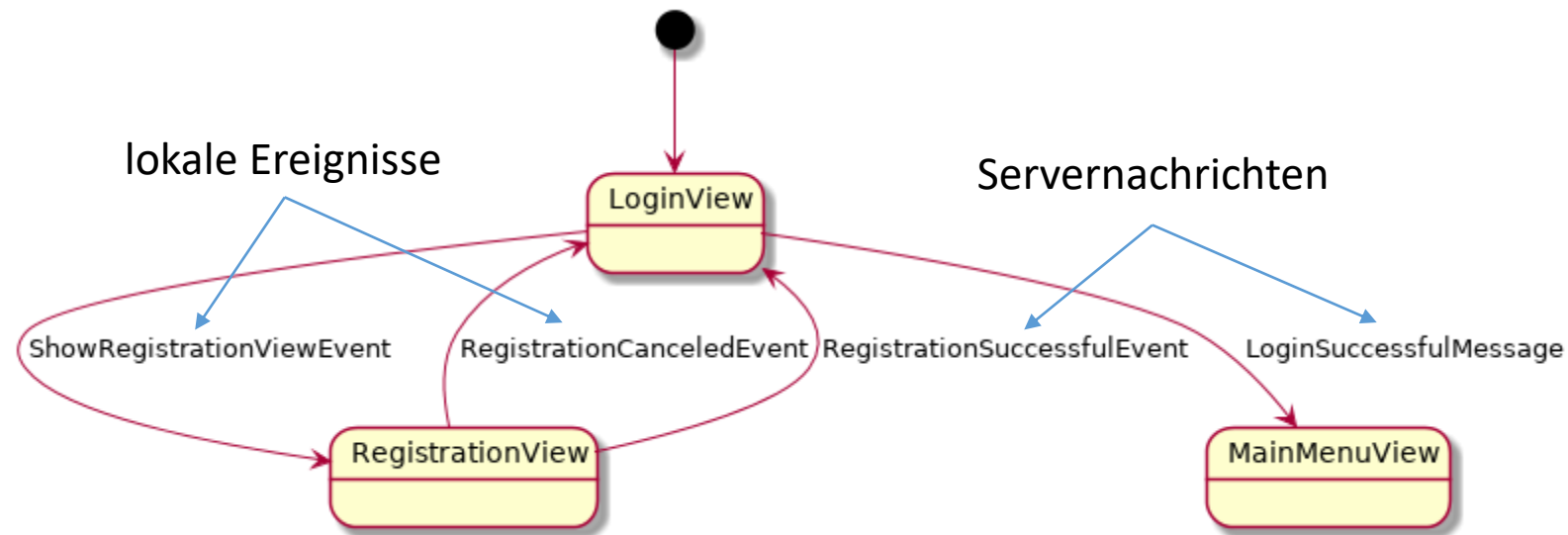
```
@Subscribe
public void onShowRegistrationViewEvent(ShowRegistrationViewEvent event){
    showRegistrationScreen();
}

@Subscribe
public void onShowLoginViewEvent(ShowLoginViewEvent event){
    showLoginScreen();
}

@Subscribe
public void onRegistrationCanceledEvent(RegistrationCanceledEvent event){
    showScene(lastScene, lastTitle);
}

@Subscribe
public void onRegistrationErrorEvent(RegistrationErrorEvent event) {
    showError(event.getMessage());
}
```

- Übergänge zwischen den Views, abhängig von Ereignissen/Nachrichten



Sollten umbenannt werden:

- RegistrationSuccessfulResponse
- LoginSuccessfulResponse

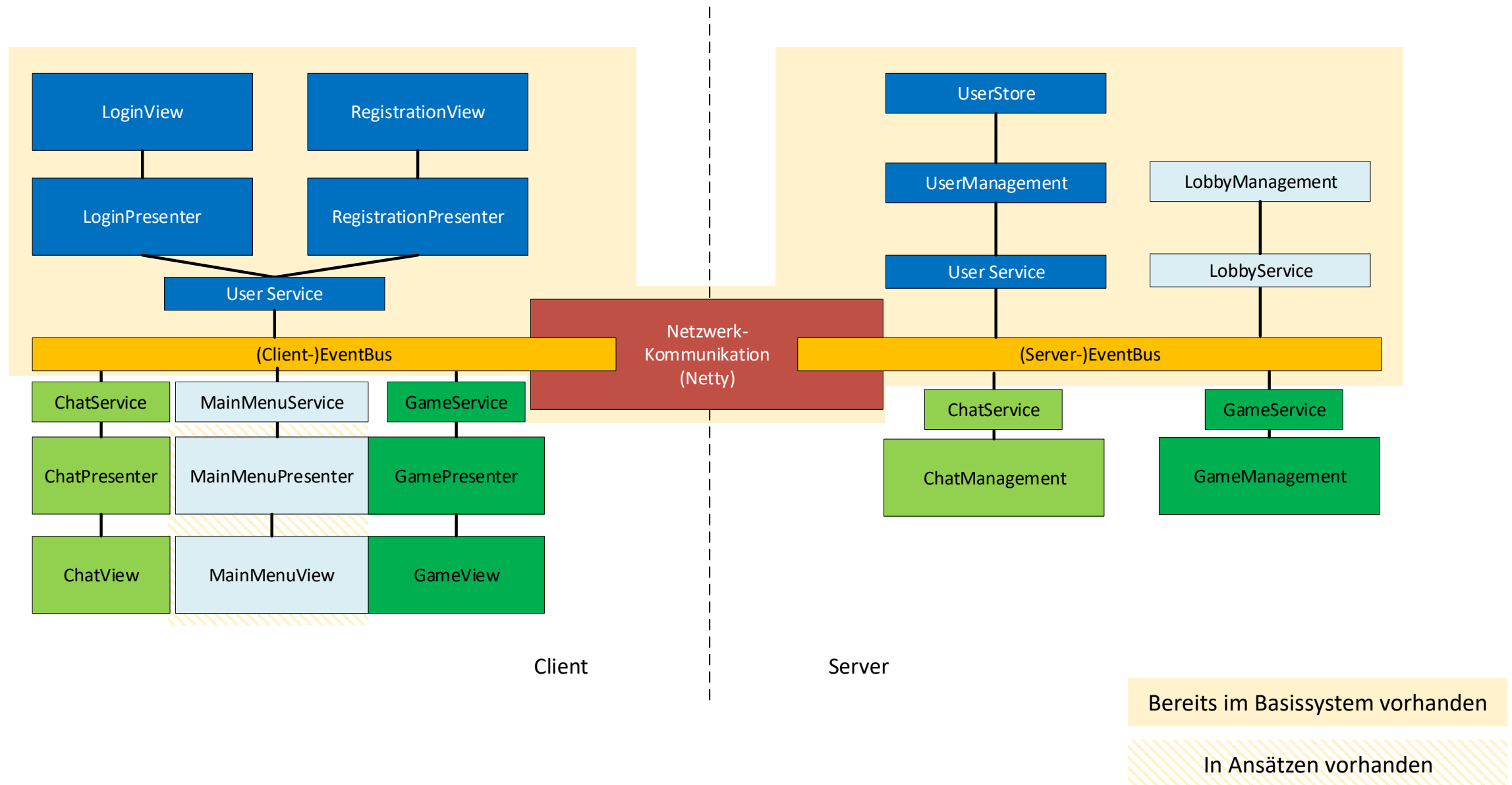
Im zentralen Basissystem aktualisiert

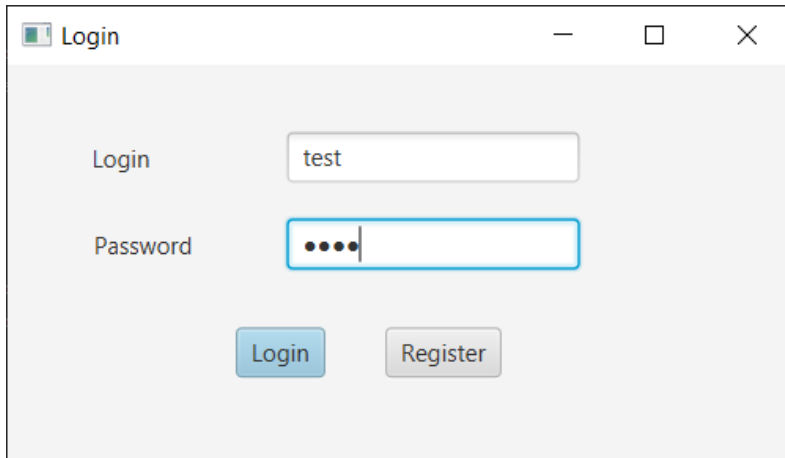
```
private void showScene(final Scene scene, final String title) {  
    this.lastScene = currentScene;  
    this.lastTitle = primaryStage.getTitle();  
    this.currentScene = scene;  
    Platform.runLater(() -> {  
        primaryStage.setTitle(title);  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    });  
}
```

- **Achtung!**
  - Netzwerk (Netty) und JavaFX haben unterschiedliche Threads
  - Nur der JavaFX-Thread darf etwas in der GUI ändern → `Platform.runLater()`;
  - Mit Lambda sieht das nicht mehr so schlimm wie früher aus ;-)

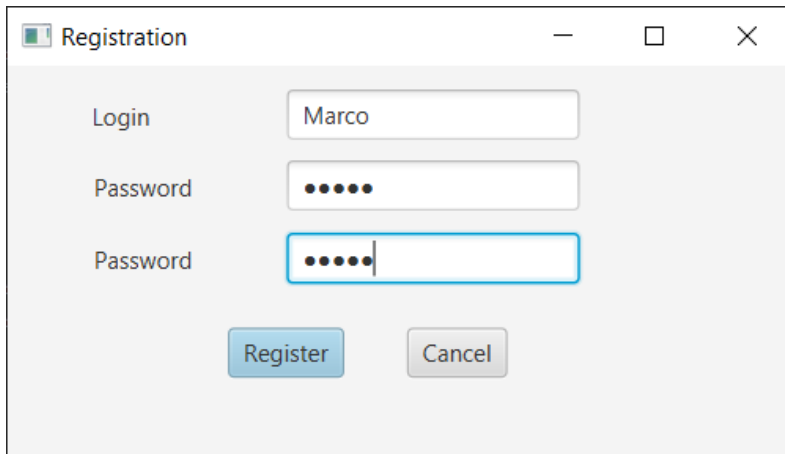
# Putting it all together



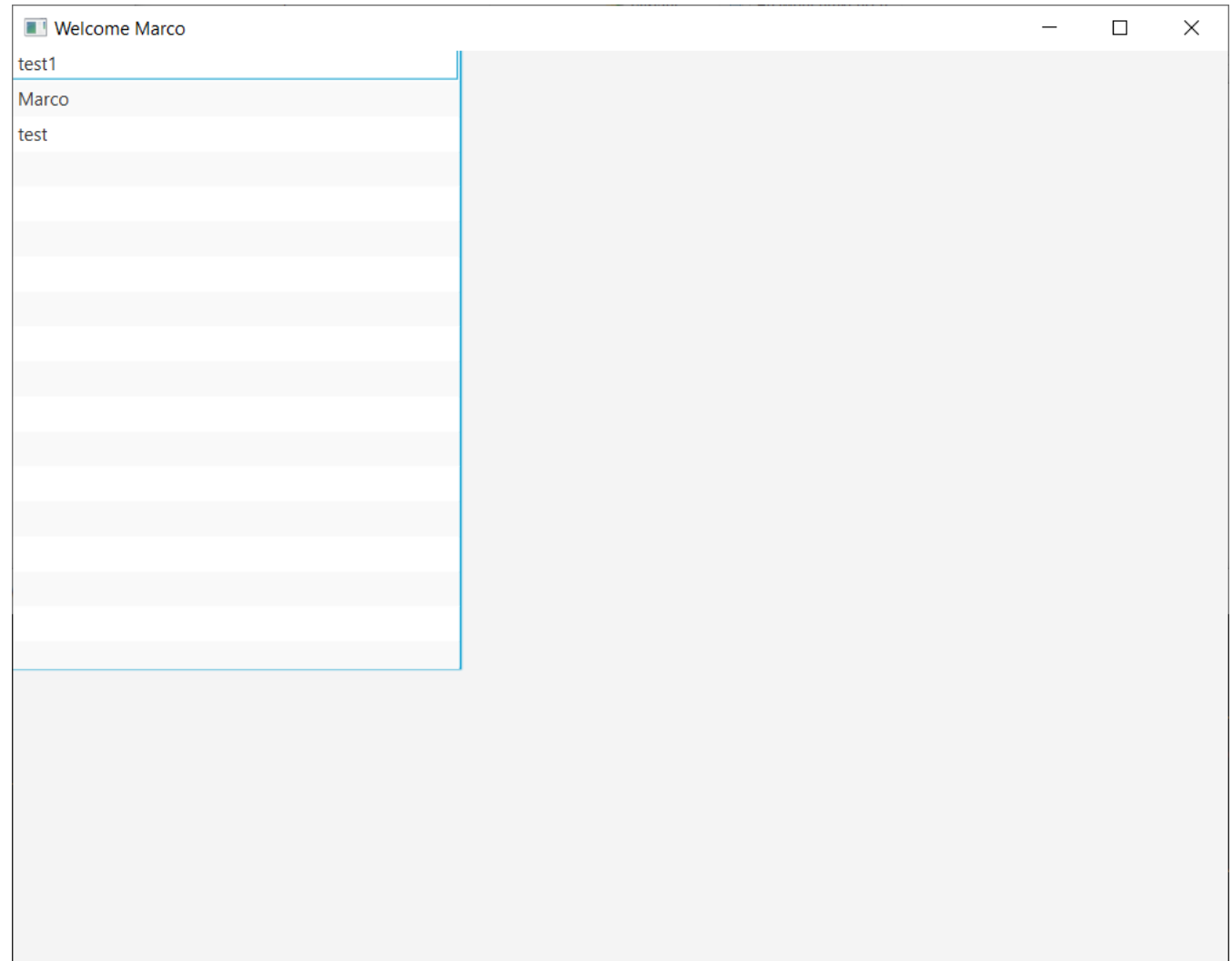




A screenshot of a web application window titled "Login". It contains two input fields: "Login" with the text "test" and "Password" with masked characters "••••". Below the fields are two buttons: "Login" (highlighted in blue) and "Register" (disabled, grey).



A screenshot of a web application window titled "Registration". It contains three input fields: "Login" with the text "Marco", "Password" with masked characters "•••••", and another "Password" field with masked characters "•••••". Below the fields are two buttons: "Register" (highlighted in blue) and "Cancel" (disabled, grey).



A screenshot of a web application window titled "Welcome Marco". It features a sidebar on the left with a list of items: "test1", "Marco", and "test". The "Marco" item is highlighted with a blue border. The main content area on the right is empty and grey.

Anmerkung: Im Basis-Projekt CSS schon drin, sollte aber dringend überarbeitet werden ;-)

- Aufbauend auf Aufgabe 1:
- Erzeugen Sie in **Jira** ein neues Ticket in Ihrem Projekt, mit einem Namen der Ihren StudIP-Login enthält: z.B. aufgabe2\_abcd1234
- Erzeugen Sie mit Hilfe des Links in dem Jira-Ticket in **Bitbucket** einen Branch. Der Name des Branches muss dem des Tickets entsprechen
- Wechseln Sie in **IntelliJ** auf diesen Branch
- Erweitern Sie das Hauptmenü um einen Logout-Button und ergänzen Sie die notwendigen Verarbeitungsschritte (analog zum Login) um ein **Logout** durchzuführen.
- Erstellen Sie einen Pull-Request in Bitbucket und weisen Sie den PR ihrem Tutor zu.
- Abgabe: bis Sonntag 23:59 Uhr.

- Erzeugen Sie in **Jira** ein neues Ticket in Ihrem Projekt, mit einem Namen der Ihren StudIP-Login enthält: z.B. aufgabe3\_abcd1234
- Erzeugen Sie mit Hilfe des Links in dem Jira-Ticket in **Bitbucket** einen Branch. Der Name des Branches muss dem des Tickets entsprechen
- Wechseln Sie in **IntelliJ** auf diesen Branch
- Erweitern Sie das Hauptmenü um einen Button zum Löschen des aktuellen Nutzers und ergänzen Sie die notwendigen Verarbeitungsschritte auf Client und Server-Seite.
- Erstellen Sie einen Pull-Request in Bitbucket und weisen Sie den PR ihrem Tutor zu.
- Abgabe: bis Sonntag 23:59 Uhr.

- Basissystem
- Git Workflow
- Bitbucket
- Jira
- MVP
- Architektur