30. May 2016 by CodeLionX on Allgemein • Edit→

HW19: Automatic Deployment

Hi together,

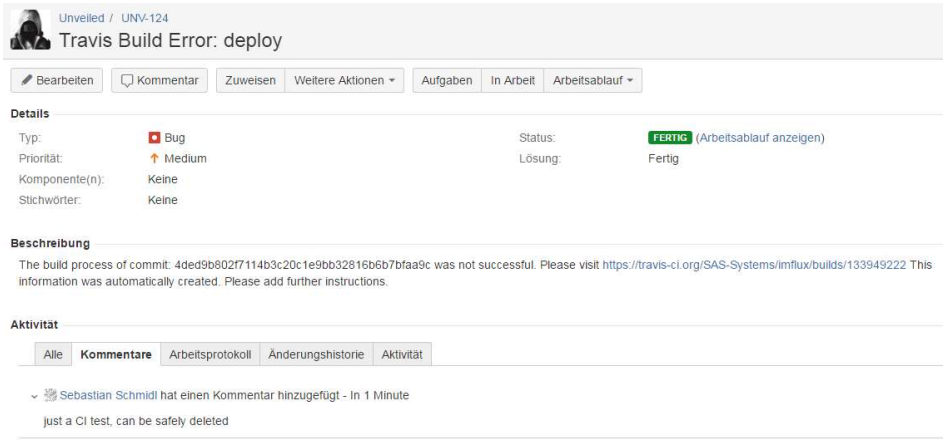
today we want to describe you our deployment process. We used TravisCI for Continuous Integration and you can find our build of our streaming library imflux here and for our Java-Backend-Stack here. We use the following tool chain (for the Unveiled-Server build):

- `mvn install -DskipTests=true -Dmaven.javadoc.skip=true -B -V`
  We use Maven for dependency management and building our application.
- `mvn clean test jacoco:report`
  JaCoCo is used for running JUnit tests and collecting the test results.
- If the build was successful following things are done:
  - `mvn coveralls:report`

The test results are reported to coveralls.io to make them available for analysis (currently there are no test for the Unveiled-Server build, but you can check the imflux test results).

- `mvn sonar:sonar -Dsonar.host.url=$SONAR_HOST_URL`
  Sonarqube analyses the binary files and reports the results to the website.
- `mvn tomcat7:deploy -Ddeploy.url=$DEPLOY_URL -Ddeploy.username=$DEPLOY_USER -Ddeploy.password=$DEPLOY_PASSWORD`
  The builded .war file is deployed to our own server using the tomcat7 maven plugin.

- `./send_jira_ticket.sh`
  If the build was not successful a new Jira ticket is created and sent to the Jira server. Therefore we use a short shell script using curl to send a Json to the Jira API. You can see the result in the following picture:



All this information can be found in our travis.yml of the repositories imflux and Unveiled-Server.

Have a nice week

CodeLionX

Send article as PDF Enter email address

30. May 2016 by CodeLionX on Allgemein • Edit→

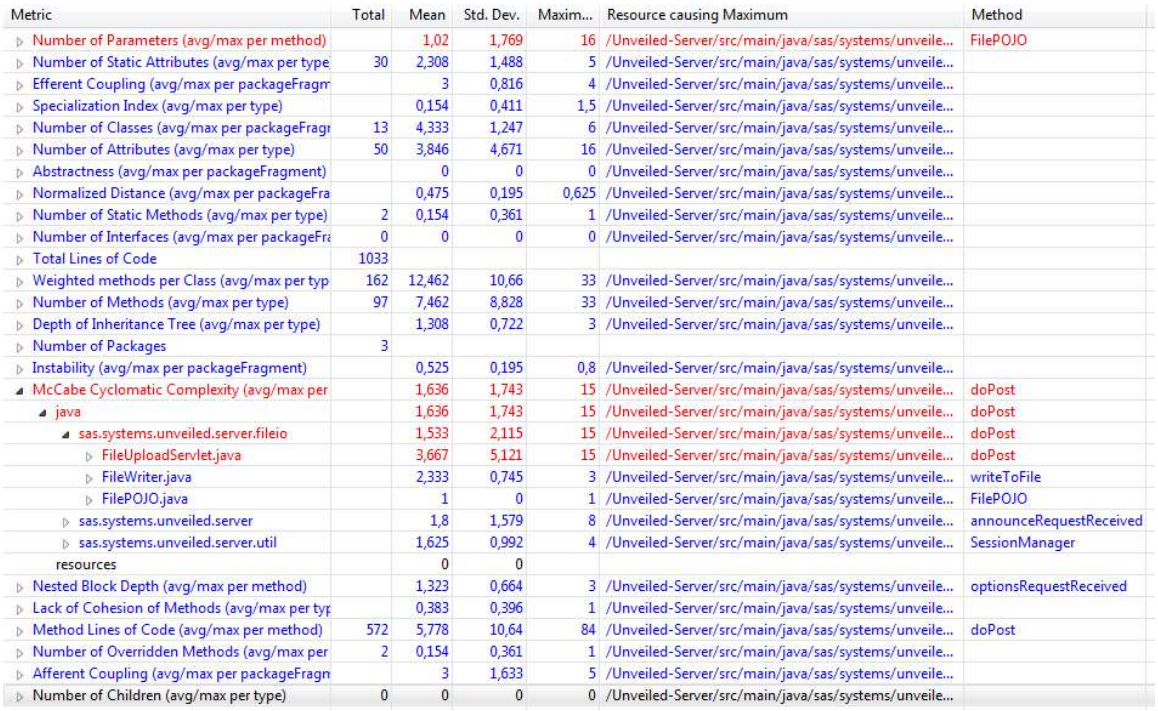
HW18: Metrics

Hey folks,

today we want to share with you, how code metrics helped us to improve our code quality. We have done the metrics analysis within our eclipse IDE and the tool Metrics 1.3.6. Metrics 1.3.6 allows us to run low level analysis. Unfortunately we were not able to find a comparable tool which allows this low level analysis within our build lifecycle. But you can take a look at our Sonarqube projects (imflux, Unveiled-Server). Sonarqube shows some high level metrics.

The following picture shows the results of Metrics 1.3.6

of the Unveiled-Server project before our changes.



Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		1,02	1,769	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
Number of Static Attributes (avg/max per type	30	2,308	1,488	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Efferent Coupling (avg/max per packageFragm		3	0,816	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Specialization Index (avg/max per type)		0,154	0,411	1,5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Classes (avg/max per packageFragr	13	4,333	1,247	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Attributes (avg/max per type)	50	3,846	4,671	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Abstractness (avg/max per packageFragment)		0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Normalized Distance (avg/max per packageFra		0,475	0,195	0,625	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Static Methods (avg/max per type)	2	0,154	0,361	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Total Lines of Code	1033					
Weighted methods per Class (avg/max per typ	162	12,462	10,66	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Methods (avg/max per type)	97	7,462	8,828	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Depth of Inheritance Tree (avg/max per type)		1,308	0,722	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Packages	3					
Instability (avg/max per packageFragment)		0,525	0,195	0,8	/Unveiled-Server/src/main/java/sas/systems/unveile...	
McCabe Cyclomatic Complexity (avg/max per		1,636	1,743	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
java		1,636	1,743	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
sas.systems.unveiled.server.fileio		1,533	2,115	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileUploadServlet.java		3,667	5,121	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileWriter.java		2,333	0,745	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	writeToFile
FilePOJO.java		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
sas.systems.unveiled.server		1,8	1,579	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
sas.systems.unveiled.server.util		1,625	0,992	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	SessionManager
resources		0	0			
Nested Block Depth (avg/max per method)		1,323	0,664	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	optionsRequestReceived
Lack of Cohesion of Methods (avg/max per typ		0,383	0,396	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Method Lines of Code (avg/max per method)	572	5,778	10,64	84	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
Number of Overridden Methods (avg/max per	2	0,154	0,361	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Afferent Coupling (avg/max per packageFragm		3	1,633	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Children (avg/max per type)	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	

As you can see we have two possible points where we are not in the green (in this case blue) area. The first one is the number of parameters, where we have a maximum of 16 parameters per method. This method is a constructor of our File POJO class. We decided to not change this class, because it represents a database record. Instead we decided to change the Cyclomatic Complexity of our `doPost()` method our `FileUploadServlet` class. You can find the code before the refactoring in our Github-repository.

To improve our code quality and the cyclomatic complexity of the `doPost()` method, we extracted some logic into new methods. Therefore it was necessary to create a new class: `FileParameters` and a new exception

class: `BadRequestException:`

```
        /**
         * Exception class for the parameter parsing r
         *
         * @author CodeLionX
         */
        private class BadRequestException extends Exc

                private static final long serialVersi

                public BadRequestException(String str
                        super(string);
                }
        }
```

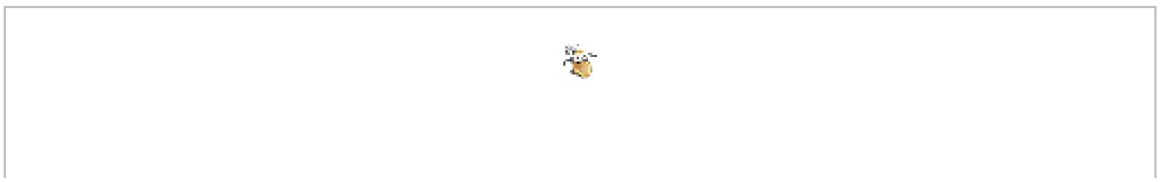
You can find the new source code here. The following screenshot shows you the metrics analysis result after refactoring:

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		1,019	1,716	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
Number of Static Attributes (avg/max per type	31	2,067	1,526	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Efferent Coupling (avg/max per packageFragm		3	0,816	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Specialization Index (avg/max per type)		0,118	0,375	1,5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Classes (avg/max per packageFragı	15	5	0,816	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Attributes (avg/max per type)	51	3,4	4,499	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Abstractness (avg/max per packageFragment)		0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Normalized Distance (avg/max per packageFra		0,475	0,195	0,625	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Static Methods (avg/max per type)	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Interfaces (avg/max per packageFrı	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Total Lines of Code	1053					
Weighted methods per Class (avg/max per typ	167	11,133	10,893	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Methods (avg/max per type)	104	6,933	8,575	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Depth of Inheritance Tree (avg/max per type)		1,4	0,8	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Packages	3					
Instability (avg/max per packageFragment)		0,525	0,195	0,8	/Unveiled-Server/src/main/java/sas/systems/unveile...	
McCabe Cyclomatic Complexity (avg/max per		1,575	1,181	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
java		1,575	1,181	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
sas.systems.unveiled.server		1,8	1,579	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
sas.systems.unveiled.server.fileio		1,423	0,948	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileUploadServlet.java		2,167	1,462	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileUploadServlet		2,273	1,483	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
doPost	6					
readRequest	3					
authenticateUserWithToken	3					
getDoubleSilently	3					
getIntSilently	3					
getBooleanSilently	2					
FileUploadServlet	1					
init	1					
destroy	1					
createDbEntry	1					
writeFile	1					
BadRequestException		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	BadRequestException
FileWriter.java		2,333	0,745	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	writeToFile
FileParameters.java		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FileParameters
FilePOJO.java		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
sas.systems.unveiled.server.util		1,625	0,992	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	SessionManager
resources		0	0			
Nested Block Depth (avg/max per method)		1,34	0,671	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	optionsRequestReceived
Lack of Cohesion of Methods (avg/max per typ		0,337	0,395	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Method Lines of Code (avg/max per method)	596	5,623	7,717	37	/Unveiled-Server/src/main/java/sas/systems/unveile...	insertFile
Number of Overridden Methods (avg/max per	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Afferent Coupling (avg/max per packageFragn		3	1,633	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Children (avg/max per type)	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	

So you can see that we now have reduced the cyclomatic complexity of the `doPost()` method from 15 to 6 by introducing six new methods and two new classes.

Have a nice week

CodeLionX



22. May 2016 by CodeLionX on Allgemein • Edit→

HW17: Test plan and coverage

Hi together,

today we want to provide you the link to our testplan. This document describes all the test we are doing to ensure we have no bugs in our new implemented features. You can find it in under documentation.

Additionally we also reached our goal to have more than 50% test coverage for our streaming library imflux. You can find all test reports on coveralls or on sonarqube (Please be aware that you might not be able to use this link inside the DHBW network. Please use this link instead.).

Have a nice week

CodeLionX

8. May 2016 by CodeLionX on Allgemein • Edit→

HW16: Implement Design Pattern in Project Code

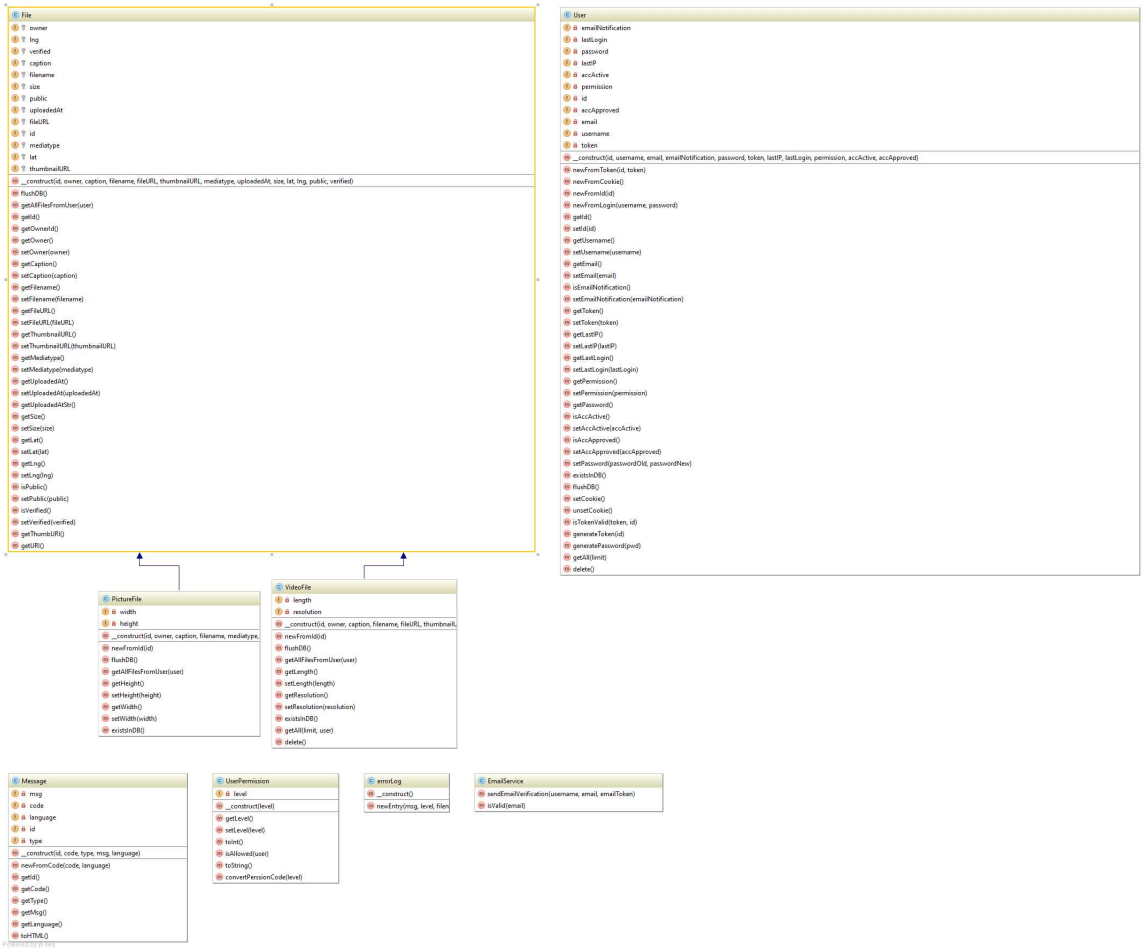
Hi together,

this weeks task was to refactor a part of our project code to implement a Design Pattern. We have chosen to implement the Data Access Object Pattern (DAO Pattern) in our server-side code. We have used it to encapsulate our low-level database communication from our server logic. The DAO Pattern is made up of

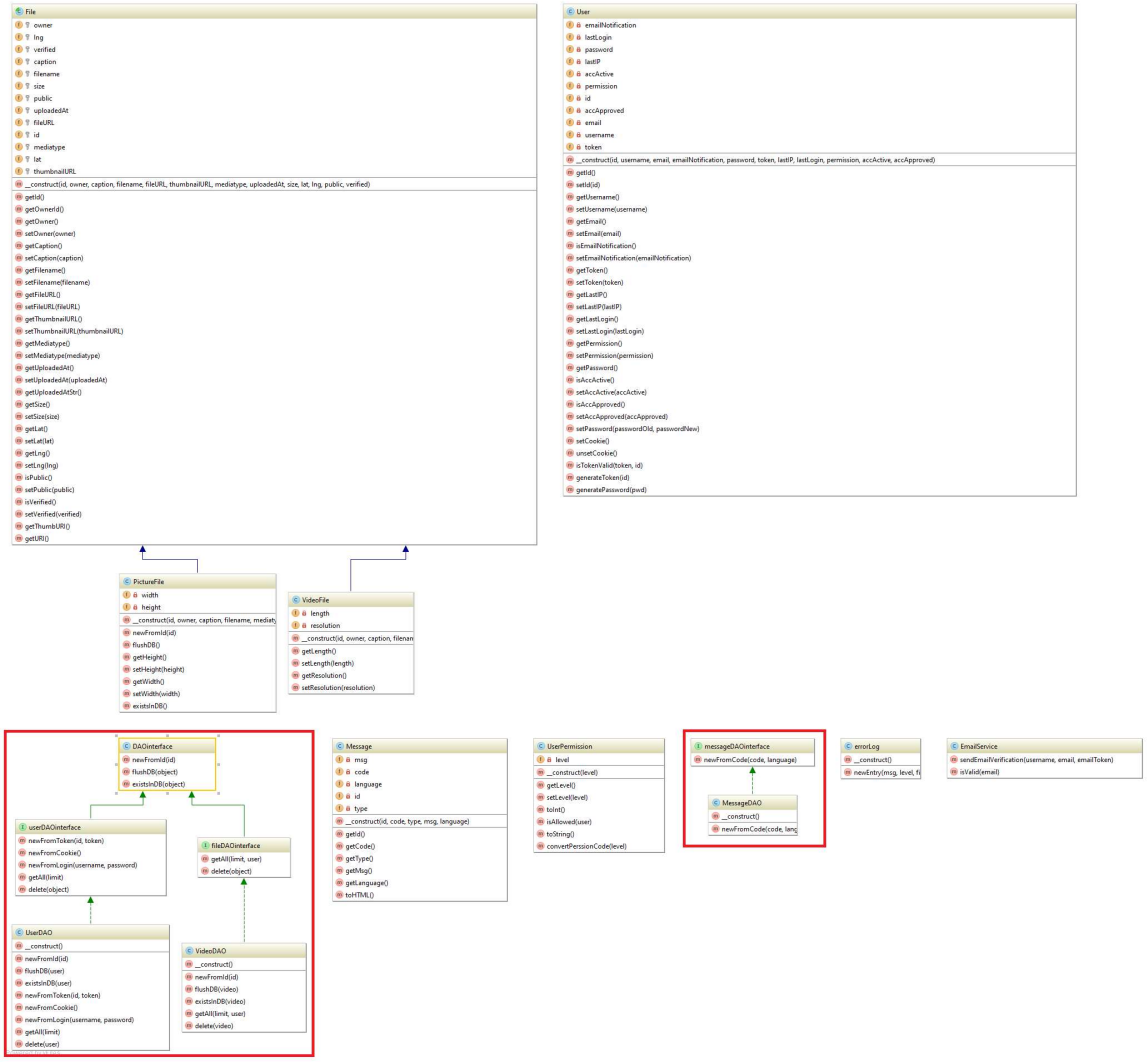
- *Model Objects* which represent one database entry,
- A *Data Access Object Interface (DAO Interface)* which defines standard actions to be performed on the *Model Objects* and
- the *Data Access Object* itself which implements the *DAO Interface* and is responsible for the communication with the data storage.

The following screenshot shows our **old**

implementation of our Backend logic:



After refactoring our code the class diagram looked like this (the **new** implementation):



You can click on both pictures to open them in full resolution.

Have a nice Sunday,

team Unveiled

Send article as PDF Enter email address

28. April 2016 by CodeLionX on Allgemein • Edit→

HW15: Fowler refactoring

Hey there,

this weeks homework was to work through chapter 1 of Fowler's book *Refactoring* and doing the steps he did in the example. We would like to share our Github-Repos in which we documented our refactoring steps. Use the following links:

- CodeLionX: Fowler/refactor
- SeanAda: Fowler/master
- Digister: Fowler-Software-Engineering-Homework/master

We hope you like our work and wish you a nice rest of the week.

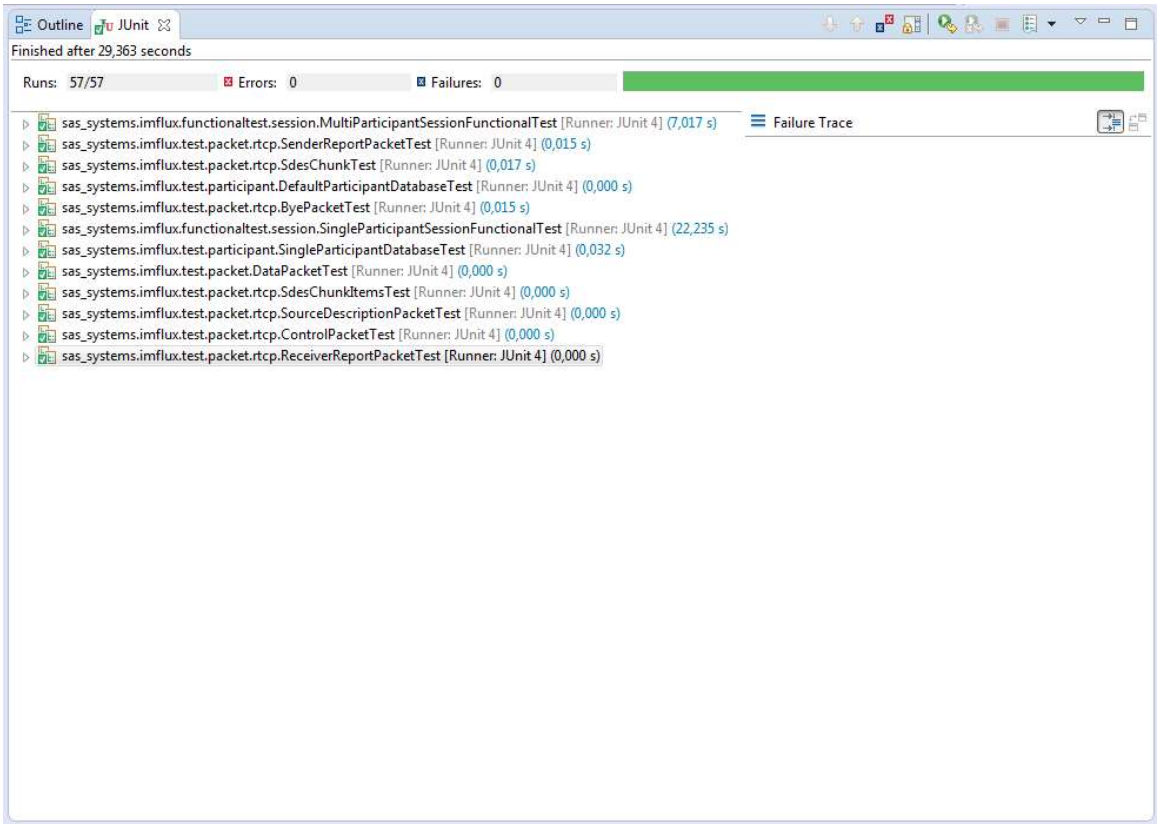
Greetings

– team Unveiled

25. April 2016 by CodeLionX on Allgemein • Edit→

HW14: Unittesting

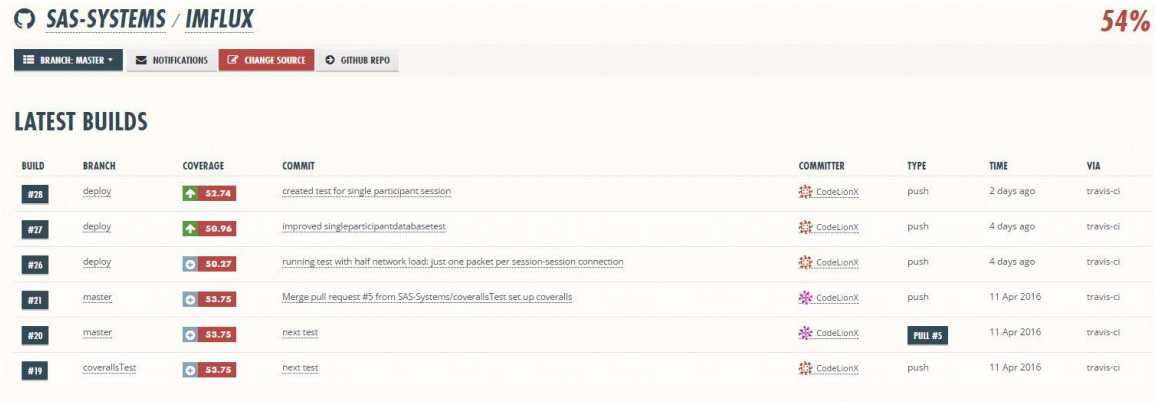
With this blog entry we want to share our testing approach. We are using JUnit for testing our backend. You can find our Testing code on Github and the following screenshot shows our IDE running these tests:



We are also using Travis-CI and Coveralls for running our tests automatically. Our build-tool is Maven and we use it to perform our testing on Travis. You can find our Maven build-file here. See the below screenshots:

```
[INFO] Starting Coveralls job for travis-ci (124033212)
[INFO] Git commit 2ea024e in deploy
[INFO] Writing Coveralls data to /home/travis/build/SAS-Systems/imflux/target/coveralls.json...
[INFO] Processing coverage report from /home/travis/build/SAS-Systems/imflux/target/site/jacoco/jacoco.xml
[INFO] Successfully wrote Coveralls data in 525ms
[INFO] Gathered code coverage metrics for 42 source files with 8269 lines of code:
[INFO] - 2264 relevant lines
[INFO] - 1194 covered lines
[INFO] - 1070 missed lines
[INFO] Submitting Coveralls data to API
[INFO] Successfully submitted Coveralls data in 848ms for Job #28.1
[INFO] https://coveralls.io/jobs/13689600
[INFO] *** It might take hours for Coveralls to update the actual coverage numbers for a job
[INFO]     If you see question marks in the report, please be patient
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 46.180 s
[INFO] Finished at: 2016-04-18T21:48:50+00:00
[INFO] Final Memory: 27M/491M
[INFO] ------------------------------------------------------------------------


Done. Your build exited with 0.
```



Demo Application in TDD

First of all we used Node.js to install the Mocha and Chai javascript Framework for testing. Now that we have access to the frameworks we were able to write our first tests and implement the function afterwards. For Node.js there is no need of an IDE, therefore we just

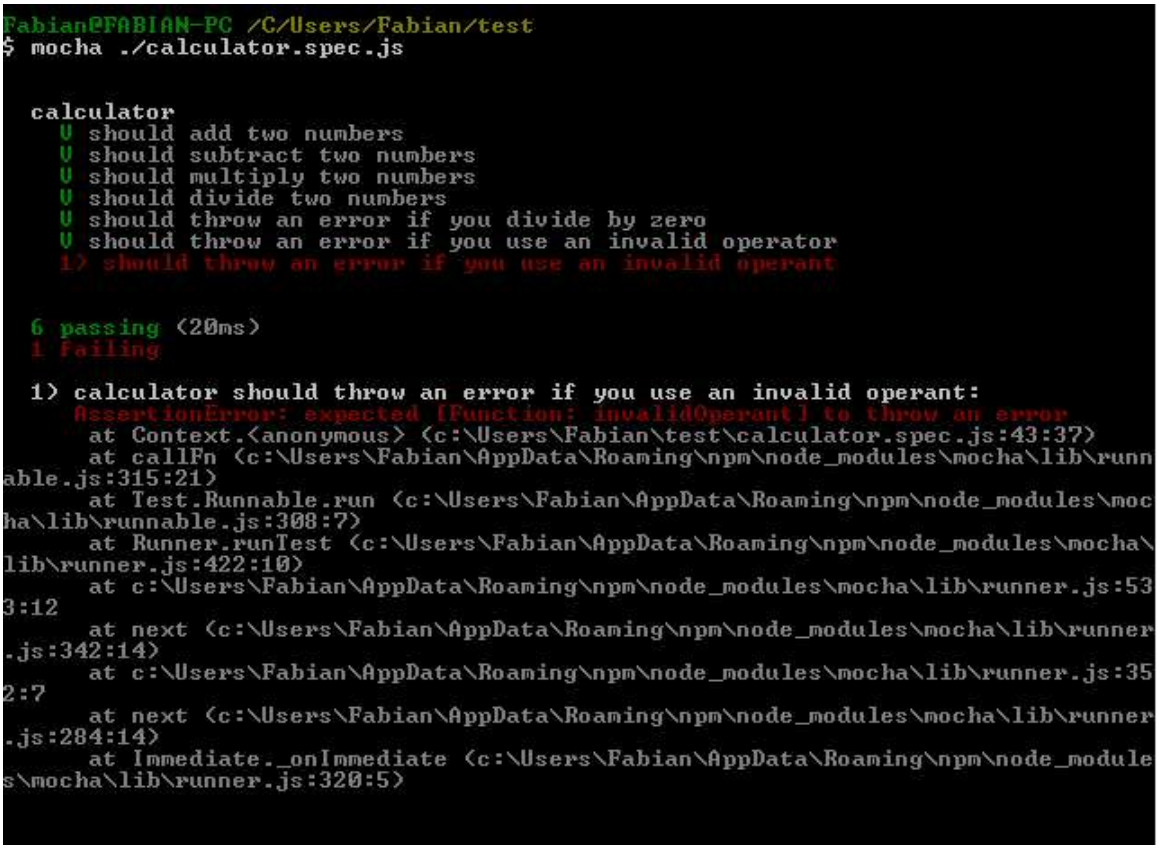
used Sublime Text 2 and the command console.

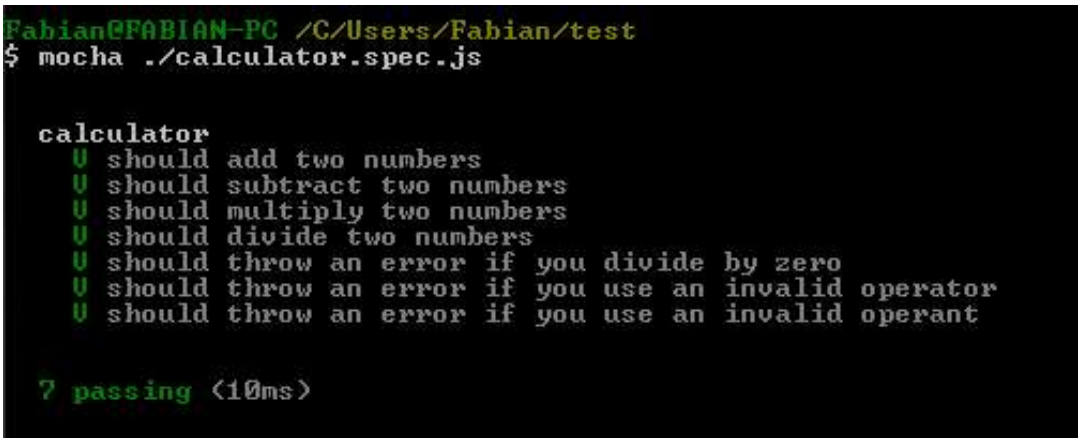
An advantage of Mocha and Chai is that the test is really easy to read. For example:

```
describe('calculator', function(){
    it('should add two numbers', function(){
        var result = calculate('3', '+', '5')
        expect(result).to.equal(8)
    })
    ...
});
```

In this test we describe the calculator. A special case of this test is that it(the calculator) should add two numbers. Now we call the calculate function with the operants 3 and 5 and the operator +. Because 3 + 5 = 8 we also expect that the result equals 8.
If you run mocha now the test will fail, because there is no code which will be executed. Now we implemented the `calculate()` – function to fix this problem. When we are done with fixing this special problem, we are going to repeat this pattern. This means we write a new test and fix it with enhancements of the `calculate()` – function.

You can find the final code in our Documentation-repository and the results of the tests here:





15. April 2016 by CodeLionX on Allgemein • Edit→

Backend API

Documentation
Postman API

We have tested our API with the Google Chrome Plugin Postman. You can add our Collection of Tests to your Postman to see and run the tests as well. Use this link.

Send article as PDF Enter email address

14. April 2016 by CodeLionX on Allgemein • Edit→

HW13: FP calculation and time estimation

Hi folks!

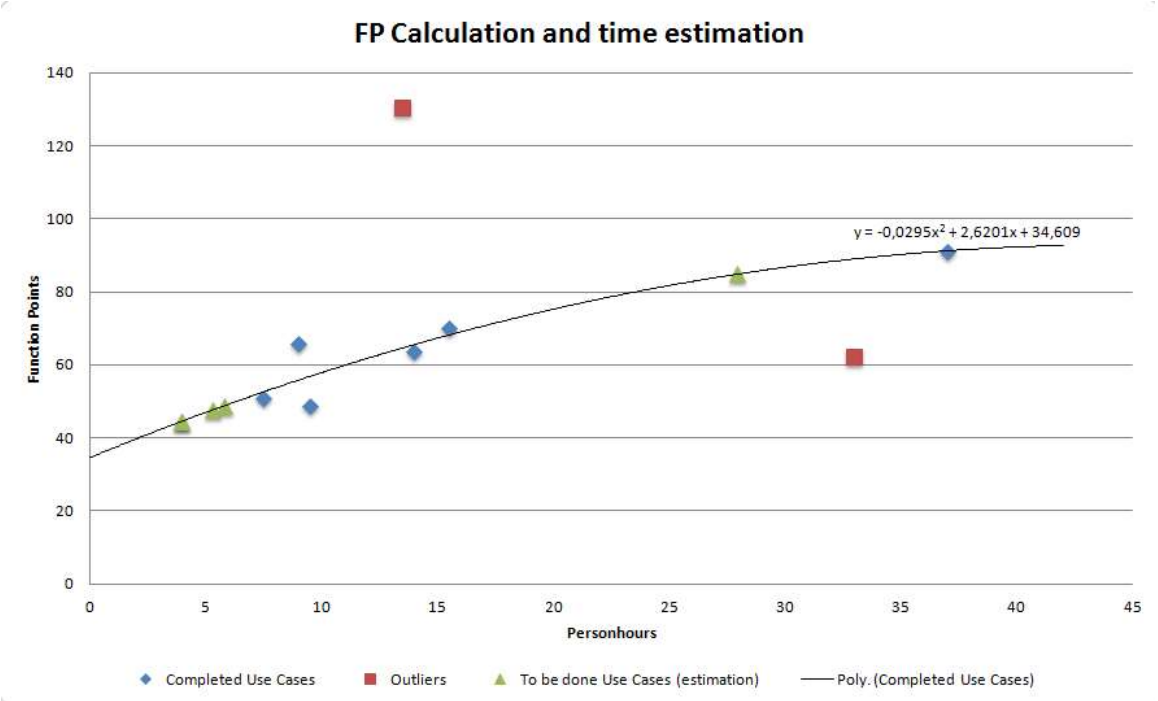
The following table shows our already completed use cases with our time spent implementing it and the corresponding Function Points:

Use Case	Use Case Name	Total Time Spent (Estimation in h)	Function Points
(1	Capture and Stream Video	13,5	130,38)
2	Configure	15,5	69,96

	Settings		
3	Maintain User Profile	9	65,72
4	Switch User	7,5	50,88
5	Register	14	63,60
6	Browse Media	33	62,40
7	Manage Users	9,5	48,76
General	RTSP Library	42	91,16

We plotted this data in a graph with the Function Points on the ordinate and the total time spent on the abscissa. As you can see we have two outliers plotted in red. The first one at x=13,5 is the Use Case Capture and Stream Video. It is very complex and affects various files, therefore it has very much FPs. We have not completed this Use Case yet and have estimated a lower number of hours, because most of the time will go to the streaming library. Maybe this was a mistake and we have to correct our estimation, so that it matchs with our chart.

The second outlier is the Use Case Browse Media, where we have spent too much time in spite of the low number of FPs. The reason for that was a change in our architecture and the technology we used for creating a web-based media browser. This lead thereto that we had to implement most of the functionality again.



We used this graph to estimate our remaining Use Cases. You can see these estimations as green triangles in the graph.

You can find all our Use Case Specifications here: Documentation

Have a nice evening,

your team Unveiled

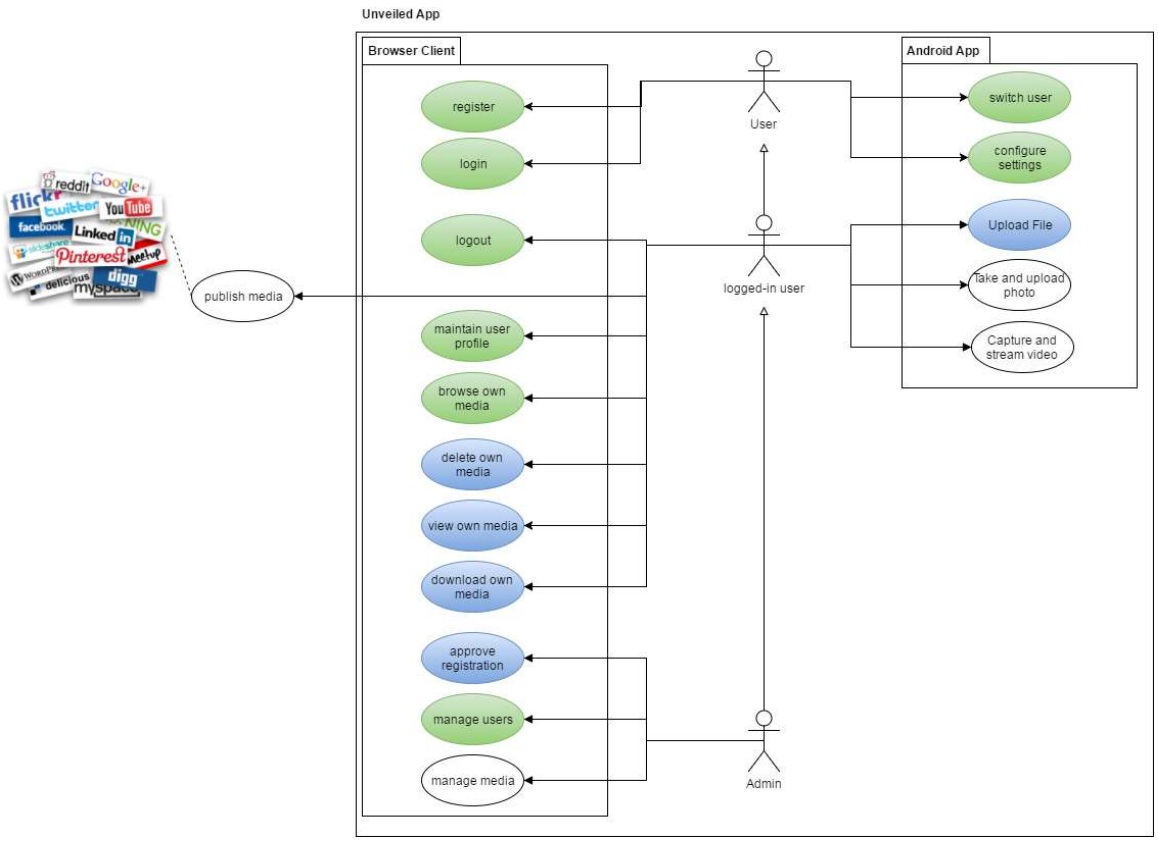
Send article as PDF Enter email address

4. April 2016 by CodeLionX on Allgemein • Edit→

HW12: Scope for 2nd Semester and Risk Management

Hey there,

this is our new overall Use Case Diagram:



The green marked Use Cases were finished in the last semester and the blue ones show the scope for this semesters work on the Unveiled project. You can find all our Use Case specifications on our Documentation site.

These are the new ones (the blue marked Use Cases):

- Use Case: Delete own Media
- Use Case: Download own Media
- Use Case: View own Media
- Use Case: Approve Registrations
- Use Case: Upload File

Risk Management

We used our Google Drive folder to create a new dynamic spreadsheet which contains our list of possible risks. You can see the current status of the list below or in the document:

Risk Rank	Risk Name	Risk Descri-ption	Risk Proba-bility of Occurence	Risk Impact	Risk Factor	Risk Mitigation	Person in Charge of Tracking
1	RTSP library won´t finish	We are writing our own streaming library based on an existing approach.	60%	100%	60%	Don't use streaming to transmit the data -> use standard upload/download	Fabian Schäfer

		This is more complex than expected and may take longer to implement					
2	lib streaming doesn´t work	The Android App will make use of an external library called “libstreaming”; when it does not fit into our architecutre it will not work	50%	90 %	45 %	check dependencies and architecture early; use own library for streaming	Sebastian Adams
3	exam preparation	in final project	100%	40 %	40 %	Make a detailled	Sebastian Schmidl

	disrupts project progress	phase also exam preparation takes place, therefore team members must prioritize and can not spend all their time for the project				planning for this final project phase early enough; cut off requirements	
4	server contract expires	Server contract expires or 1&1 terminates the contract	80%	40 %	32 %	check contract periodically; search and prepare backup solution	Sebastian Adams
5	bad code quality	end product contains a lot of bugs	40%	60 %	24 %	Write tests; make code reviews	Fabian Schäfer

		or the performance of the application is poor					
6	case of illness	A team member sustains a little or serious injury or becomes sick for a longer time period	40%	30%	12%	divide tasks of this person to the other team members; cut off some requirements	Sebastian Schmidl
7	server stops working	Private hosted server stops working, because of any issue	30%	30%	9%	invest additional time on getting server running again; make data backups perio	Sebastian Adams

						dically; provide user guide with server set up steps	
8	team member leaves the university	Team member becomes exmatriculated or leaves the univerity beacause of personal reasons	10%	70 %	7%	cut off requirements and use cases	Sebastian Schmidl

Use Case Estimation

The following table shows our time spent for the different Use Cases without all the work which would belong to multiple Use Cases:

Semester	Use Case	Use Case	Time spent (estimation in h)	Functio

		Name	Documentation	Coding	Testing	Total	n Points
2	1	Capture and Stream Video	1,5	8	4	13,5	
1	2	Configure Settings	1,5	8	6	15,5	
1	3	Maintain User Profile	1	6	1	8	
1	4	Switch User	1	4	2,5	7,5	
1	5	Register	1	8	1	10	
1	6	Browse Media	1	34	2	37	
1	7	Manage Users	1,5	6	2	9,5	

22. December 2015 by CodeLionX on Allgemein • Edit→

HW11: Midterm Presentation and Grading

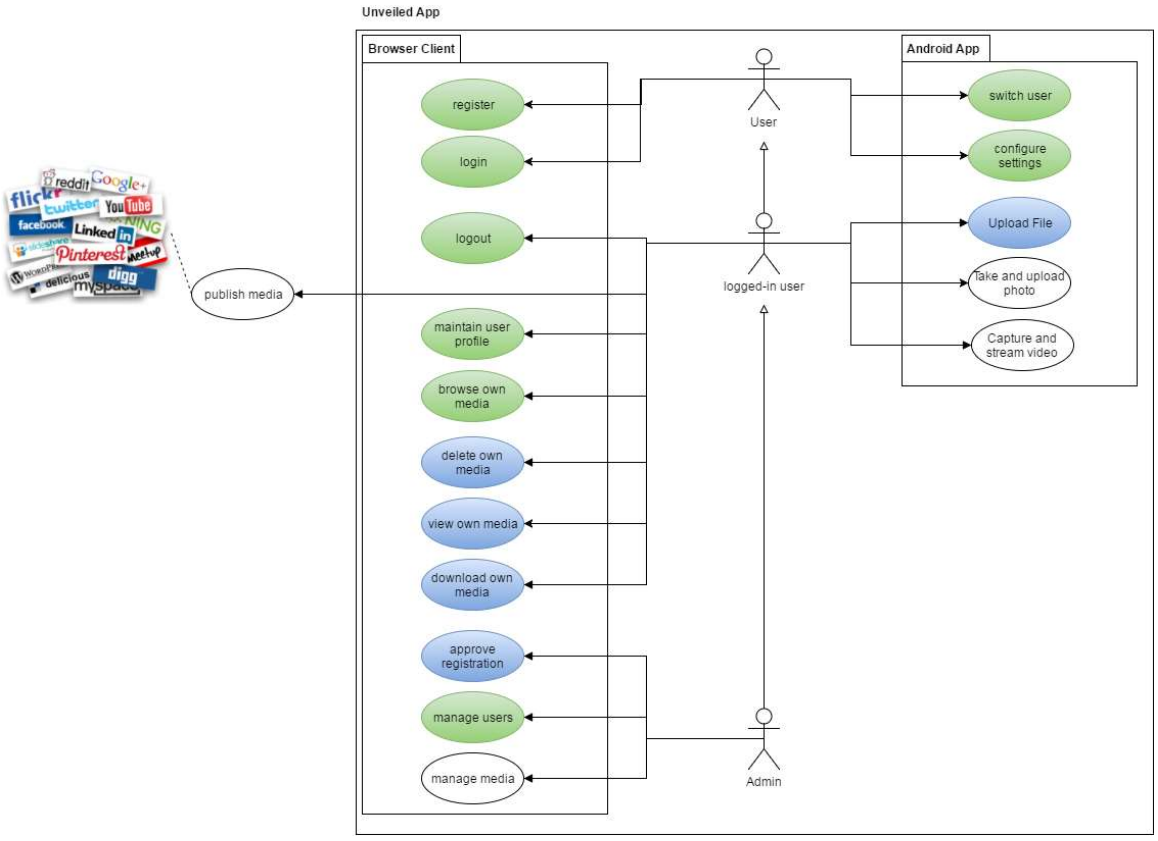
Hello together,

today we want to show you all the work we've done in the past semester. To get a quick overview you can scroll through our presentation uploaded on Github: Unveiled_MidtermPresentation_V1.0.pptx.

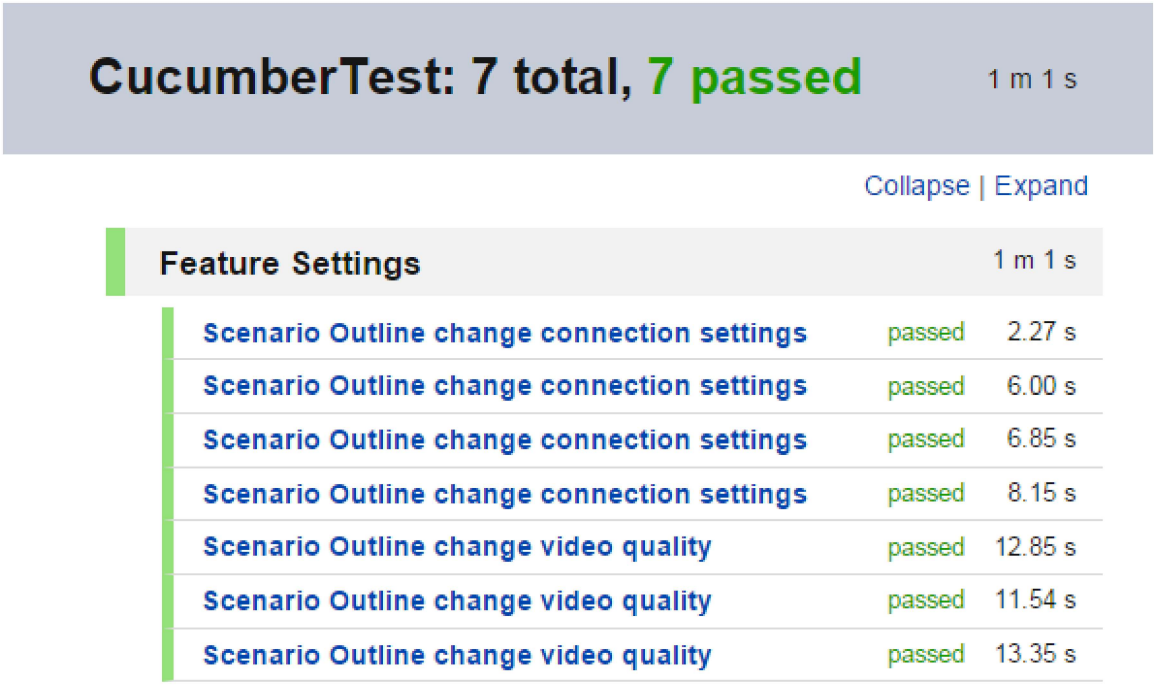
We are team Unveiled and that's how we had split up the tasks among us:

Team	 **Unveiled** **Fight against injustice straightaway**	
Members	**Roles**	**Time spent**
Sebastian Adams	Deployment / Configuration Manager System Analyst Sofware Architect Backend Implementer	86h
Fabian Schäfer	Software Architect Frontend Business Process Analyst Designer Implementer	82h
Sebastian Schmidl	Deployment / Configuration Manager Project Manager Test Manager Implementer	117h

To get a better overview about our tasks, you can see all use cases in the next picture illustrated by a diagram. Green use cases are already done. As you can see, the main focus in this semester was on creating the web interface for managing both the content and the users. Most of the use cases are located in the web interface component because we want the Android app to be as lightweight and lean as possible.



You can find all documents including the SRS, use case descriptions and the SAD on our Documentation-Page. Our test cases written in Gherkin are shown in the corresponding use case descriptions (UC1: Capture and stream video, UC2: Configure settings and UC4: Switch user). The following picture shows the test log of the use case: Configure settings:



CucumberTest: 7 total, 7 passed 1 m 1 s

Collapse | Expand

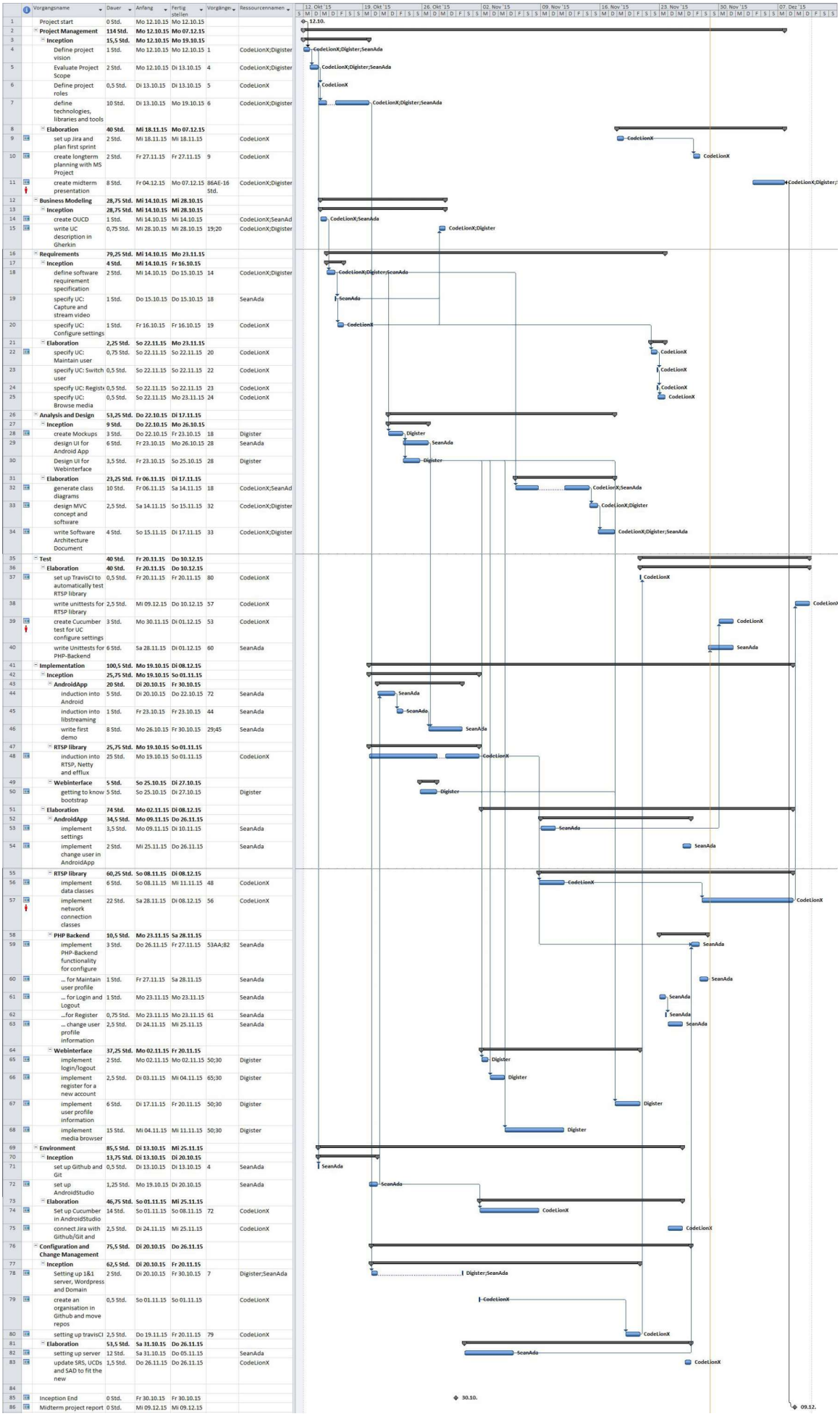
Feature Settings		1 m 1 s
Scenario Outline change connection settings	passed	2.27 s
Scenario Outline change connection settings	passed	6.00 s
Scenario Outline change connection settings	passed	6.85 s
Scenario Outline change connection settings	passed	8.15 s
Scenario Outline change video quality	passed	12.85 s
Scenario Outline change video quality	passed	11.54 s
Scenario Outline change video quality	passed	13.35 s

We have also done some unit testing for the server-side code and our own streaming library. You can find the test code in the related Github-Repository (SAS-Systems/imflux/test). These tests are also run by travisCI as you can see in the travis-logs.

Regarding project management, you can find our Jira-Board and our Burndown-Diagrams here. The long-term planning was made with MS Project and you can get the file from our Github-Repository (Unveiled.mpp). The following picture shows our complete Gantt-Chart.

@Ms. Berkling: We've discussed in class whether it's better to group the tasks by discipline or by phase first. We came to the decision that grouping the tasks by

discipline first makes also sense.



	Vorgangsname	Dauer	Anfang	Fertig stellen	Vorgänger	Ressourcennamen
1	Project start	0 Std.	Mo 12.10.15	Mo 12.10.15		
2	**Project Management**	**114 Std.**	**Mo 12.10.15**	**Mo 07.12.15**		
3	**Inception**	**15,5 Std.**	**Mo 12.10.15**	**Mo 19.10.15**		
4	Define project vision	1 Std.	Mo 12.10.15	Mo 12.10.15	1	CodeLionX;Digister
5	Evaluate Project Scope	2 Std.	Mo 12.10.15	Di 13.10.15	4	CodeLionX;Digister
6	Define project roles	0,5 Std.	Di 13.10.15	Di 13.10.15	5	CodeLionX
7	define technologies, libraries and tools	10 Std.	Di 13.10.15	Mo 19.10.15	6	CodeLionX;Digister
8	**Elaboration**	**40 Std.**	**Mi 18.11.15**	**Mo 07.12.15**		
9	set up Jira and plan first sprint	2 Std.	Mi 18.11.15	Mi 18.11.15		CodeLionX
10	create longterm planning with MS Project	2 Std.	Fr 27.11.15	Fr 27.11.15	9	CodeLionX
11	create midterm presentation	8 Std.	Fr 04.12.15	Mo 07.12.15	86AE-16 Std.	CodeLionX;Digister
12	**Business Modeling**	**28,75 Std.**	**Mi 14.10.15**	**Mi 28.10.15**		
13	**Inception**	**28,75 Std.**	**Mi 14.10.15**	**Mi 28.10.15**		
14	create OUCD	1 Std.	Mi 14.10.15	Mi 14.10.15		CodeLionX;SeanAd
15	write UC description in Gherkin	0,75 Std.	Mi 28.10.15	Mi 28.10.15	19;20	CodeLionX;Digister
16	**Requirements**	**79,25 Std.**	**Mi 14.10.15**	**Mo 23.11.15**		
17	**Inception**	**4 Std.**	**Mi 14.10.15**	**Fr 16.10.15**		
18	define software requirement specification	2 Std.	Mi 14.10.15	Do 15.10.15	14	CodeLionX;Digister
19	specify UC: Capture and stream video	1 Std.	Do 15.10.15	Do 15.10.15	18	SeanAda
20	specify UC: Configure settings	1 Std.	Fr 16.10.15	Fr 16.10.15	19	CodeLionX
21	**Elaboration**	**2,25 Std.**	**So 22.11.15**	**Mo 23.11.15**		
22	specify UC: Maintain user	0,75 Std.	So 22.11.15	So 22.11.15	20	CodeLionX
23	specify UC: Switch user	0,5 Std.	So 22.11.15	So 22.11.15	22	CodeLionX
24	specify UC: Registe	0,5 Std.	So 22.11.15	So 22.11.15	23	CodeLionX
25	specify UC: Browse media	0,5 Std.	So 22.11.15	Mo 23.11.15	24	CodeLionX
26	**Analysis and Design**	**53,25 Std.**	**Do 22.10.15**	**Di 17.11.15**		
27	**Inception**	**9 Std.**	**Do 22.10.15**	**Mo 26.10.15**		
28	create Mockups	3 Std.	Do 22.10.15	Fr 23.10.15	18	Digister
29	design UI for Android App	6 Std.	Fr 23.10.15	Mo 26.10.15	28	SeanAda
30	Design UI for Webinterface	3,5 Std.	Fr 23.10.15	So 25.10.15	28	Digister
31	**Elaboration**	**23,25 Std.**	**Fr 06.11.15**	**Di 17.11.15**		
32	generate class diagrams	10 Std.	Fr 06.11.15	Sa 14.11.15	18	CodeLionX;SeanAd
33	design MVC concept and software	2,5 Std.	Sa 14.11.15	So 15.11.15	32	CodeLionX;Digister
34	write Software Architecture Document	4 Std.	So 15.11.15	Di 17.11.15	33	CodeLionX;Digister
35	**Test**	**40 Std.**	**Fr 20.11.15**	**Do 10.12.15**		
36	**Elaboration**	**40 Std.**	**Fr 20.11.15**	**Do 10.12.15**		
37	set up TravisCI to automatically test RTSP library	0,5 Std.	Fr 20.11.15	Fr 20.11.15	80	CodeLionX
38	write unittests for RTSP library	2,5 Std.	Mi 09.12.15	Do 10.12.15	57	CodeLionX
39	create Cucumber test for UC configure settings	3 Std.	Mo 30.11.15	Di 01.12.15	53	CodeLionX
40	write Unittests for PHP-Backend	6 Std.	Sa 28.11.15	Di 01.12.15	60	SeanAda
41	**Implementation**	**100,5 Std.**	**Mo 19.10.15**	**Di 08.12.15**		
42	**Inception**	**25,75 Std.**	**Mo 19.10.15**	**So 01.11.15**		
43	**AndroidApp**	**20 Std.**	**Di 20.10.15**	**Fr 30.10.15**		
44	induction into Android	5 Std.	Di 20.10.15	Do 22.10.15	72	SeanAda
45	induction into libstreaming	1 Std.	Fr 23.10.15	Fr 23.10.15	44	SeanAda
46	write first demo	8 Std.	Mo 26.10.15	Fr 30.10.15	29;45	SeanAda
47	**RTSP library**	**25,75 Std.**	**Mo 19.10.15**	**So 01.11.15**		
48	induction into RTSP, Netty and efflux	25 Std.	Mo 19.10.15	So 01.11.15		CodeLionX
49	**Webinterface**	**5 Std.**	**So 25.10.15**	**Di 27.10.15**		
50	getting to know bootstrap	5 Std.	So 25.10.15	Di 27.10.15		Digister
51	**Elaboration**	**74 Std.**	**Mo 02.11.15**	**Di 08.12.15**		
52	**AndroidApp**	**34,5 Std.**	**Mo 09.11.15**	**Do 26.11.15**		
53	implement settings	3,5 Std.	Mo 09.11.15	Di 10.11.15		SeanAda
54	implement change user in AndroidApp	2 Std.	Mi 25.11.15	Do 26.11.15		SeanAda
55	**RTSP library**	**60,25 Std.**	**So 08.11.15**	**Di 08.12.15**		
56	implement data classes	6 Std.	So 08.11.15	Mi 11.11.15	48	CodeLionX
57	implement network connection classes	22 Std.	Sa 28.11.15	Di 08.12.15	56	CodeLionX
58	**PHP Backend**	**10,5 Std.**	**Mo 23.11.15**	**Sa 28.11.15**		
59	implement PHP-Backend functionality for configure	3 Std.	Do 26.11.15	Fr 27.11.15	53AA;82	SeanAda
60	... for Maintain user profile	1 Std.	Fr 27.11.15	Sa 28.11.15		SeanAda
61	... for Login and Logout	1 Std.	Mo 23.11.15	Mo 23.11.15		SeanAda
62	...for Register	0,75 Std.	Mo 23.11.15	Mo 23.11.15	61	SeanAda
63	... change user profile information	2,5 Std.	Di 24.11.15	Mi 25.11.15		SeanAda
64	**Webinterface**	**37,25 Std.**	**Mo 02.11.15**	**Fr 20.11.15**		
65	implement login/logout	2 Std.	Mo 02.11.15	Mo 02.11.15	50;30	Digister
66	implement register for a new account	2,5 Std.	Di 03.11.15	Mi 04.11.15	65;30	Digister
67	implement user profile information	6 Std.	Di 17.11.15	Fr 20.11.15	50;30	Digister
68	implement media browser	15 Std.	Mi 04.11.15	Mi 11.11.15	50;30	Digister
69	**Environment**	**85,5 Std.**	**Di 13.10.15**	**Mi 25.11.15**		
70	**Inception**	**13,75 Std.**	**Di 13.10.15**	**Di 20.10.15**		
71	set up Github and Git	0,5 Std.	Di 13.10.15	Di 13.10.15	4	SeanAda
72	set up AndroidStudio	1,25 Std.	Mo 19.10.15	Di 20.10.15		SeanAda
73	**Elaboration**	**46,75 Std.**	**So 01.11.15**	**Mi 25.11.15**		
74	Set up Cucumber in AndroidStudio	14 Std.	So 01.11.15	So 08.11.15	72	CodeLionX
75	connect Jira with Github/Git and	2,5 Std.	Di 24.11.15	Mi 25.11.15		CodeLionX
76	**Configuration and Change Management**	**75,5 Std.**	**Di 20.10.15**	**Do 26.11.15**		
77	**Inception**	**62,5 Std.**	**Di 20.10.15**	**Fr 20.11.15**		
78	Setting up 1&1 server, Wordpress and Domain	2 Std.	Di 20.10.15	Fr 30.10.15	7	Digister;SeanAda
79	create an organisation in Github and move repos	0,5 Std.	So 01.11.15	So 01.11.15		CodeLionX
80	setting up travisCI	2,5 Std.	Do 19.11.15	Fr 20.11.15	79	CodeLionX
81	**Elaboration**	**53,5 Std.**	**Sa 31.10.15**	**Do 26.11.15**		
82	setting up server	12 Std.	Sa 31.10.15	Do 05.11.15		SeanAda
83	update SRS, UCDs and SAD to fit the new	1,5 Std.	Do 26.11.15	Do 26.11.15		CodeLionX
84						
85	Inception End	0 Std.	Fr 30.10.15	Fr 30.10.15		
86	Midterm project report	0 Std.	Mi 09.12.15	Mi 09.12.15		

Now we can show you the most important item:
A **demo** of our application.
You can access our web interface here. If you have an Android device, you can get our packed application from Github (Unveiled_app.apk) and install it on your device or virtual device.

All our code is hosted by Github:

- AndroidApp: https://github.com/SAS-Systems/Unveiled/tree/Android
- Webinterface: https://github.com/SAS-Systems/Unveiled/tree/Backend-PHP-Stack/Backend-PHP-Stack/webinterface
- Backend PHP-Stack: https://github.com/SAS-Systems/Unveiled/tree/Backend-PHP-Stack/Backend-PHP-Stack
- Backend Java-Stack: not yet implemented (needs imflux to be finished first)
- imflux (streaming library): https://github.com/SAS-Systems/imflux

We wish you merry Christmas and a happy new year!!

Your team Unveiled
– CodeLionX
– Digister

– Seanada

Send article as PDF Enter email address

Page 1 of 2 Older Posts →

Documentation links:

- SRS
- SAS
- Use-Case: Capture and stream video
- Use-Case: Configure settings
- Use-Case: Maintain user