

Contents

1	2018	5
1.1	October	5
	Going For Another Adventurous Journey (2018-10-06 22:54)	5
	Project Vision (2018-10-10 08:24)	6
	Distribution Of Roles, Technology Stack And More... (2018-10-14 12:30)	8
	Software Requirements Specification (2018-10-21 14:36)	20
	Use Case Specifiactions (2018-10-28 23:38)	22
1.2	November	26
	Gherkin Feature Files And Selenium Testing (2018-11-04 17:41)	26
	DB Scheme And Class Diagram (2018-11-11 23:00)	29
	Scrumming With YouTrack And Project Management (2018-11-18 09:52)	33
	Scrum Retrospective (2018-11-23 09:43)	41
1.3	December	43
	MVC Architecture (2018-12-02 18:26)	43
	Midterm Summary (2018-12-22 14:50)	46
2	2019	49
2.1	April	49
	Confirmation Of Scope (2019-04-02 14:34)	49
	Risk Management (2019-04-07 13:39)	51
	Time Estimation With Function Points (2019-04-22 18:43)	54
	Testing (2019-04-26 12:01)	58
2.2	May	62
	Refactoring (2019-05-05 15:08)	62
	Design Patterns (2019-05-13 19:05)	66
	Scrum Retrospective II (2019-05-17 14:43)	71
	Test Coverage (2019-05-20 14:01)	73
	Metrics (2019-05-26 19:25)	75

2.3	June	77
	Installation (2019-06-07 15:17)	77
	Deployment, Continuous Integration & Tech Stack (2019-06-13 18:44)	78
	Indeed, it was a very adventurous journey (2019-06-17 01:21)	80

1. 2018

1.1 October

Going For Another Adventurous Journey (2018-10-06 22:54)



In the first session of our weekly software engineering course, we were told not to overestimate our capabilities according to the Dunning-Kruger effect.

We as "team dashup" drive another mentality:

If there is no enemy within, the enemy outside can do us no harm. Who should stop us?

Join us on our trip through the dashup software project!

Project Vision (2018-10-10 08:24)



What is dashup?

In today's digital world, automation is everything. To keep up with modern achievement-oriented society, people need to accomplish daily tasks and planning more efficiently. Therefore people are used to create or consume multiple personal productivity tools. Keeping an overview over your own pool of tools might occur difficult sometimes. Usage is complex, platforms are different and a common design is screwed up. Inspired by this efficiency loss, we create dashup to give you the ultimate tool to get back on track again.

Dashup helps you to get through daily tasks easier by providing various widgets for typical private productivity usage scenarios on a central platform. Design your platform according to individual needs, take use of a wide range of dashup-widgets from our store or create your own productivity widget and add external APIs to feed it with data to evaluate. This means dashup could as well be used as a central platform to consume or administrate own web services. You can even publish your widget to a broad community. Dashup saves you from spending too much time on analyzing data or using complex applications. You can take use of each widget restricted to the essential set of features on one central platform, thus leading to more efficiency. Each widget will be displayed as its own panel according to global design policies on the central dashboard.

All in all, dashup will provide you the best experience and overview over all daily productivity tools on a central platform.

Product Range

Our product covers the following features

- Wide range of widgets for increased productivity scenarios such as stock price visualization, finance and weather tracking, todo list editing and more...
- Central platform for displaying analyzed data provided by external APIs
- Global design policy and customizable layout
- Marketplace offering all available widgets
- Possibility to develop custom widgets and to publish them to the dashup marketplace

If you are as excited as we are, then join us on our journey!

SoftwareEngineeringEnthusiast (2018-10-10 09:11:50)

Hey Guys, you have a great and challenging idea. We are looking forward to get more insights into your project. What tools you will be using, what features you will provide, what architecture you will go for? We are looking forward to hear more about your project in the next blog entries. Keep up the good work. Your travelling Software Engineering Enthusiasts

Felix Hausberger (2018-10-10 09:46:30)

Dear SOFTWAREENGINEERINGENTHUSIAST, Thanks for your feedback. We will provide another blog post in future containing information about our technology stack, productivity tools, built tools and more!

MNZ-Team (2018-10-10 09:25:49)

Hey guys, first of all, we really like your idea and your vision sounds great. However, I didn't quite understand how exactly you are trying to automate multiple personal productivity tools by having them on shared platform. I'd appreciate it if you could elaborate on that a bit more. It would also be nice, if you could tell us a bit more about your general setup and the individual member roles. We are looking forward to hearing more about your project next week. Keep up the good work. Your MNZ Team.

Felix Hausberger (2018-10-10 10:03:34)

Dear MNZ-Team, Thanks for your feedback. There will be another blog post in future containing information about our technology stack, productivity tools, built tools and more! In addition, we edited our blog post according to your concerns.

Christian Schweigel (2018-10-10 09:34:04)

Hello dashup team, your vision sounds great. A platform to integrate own microservices would be great for lots of developers to get their applications available for more people. Just one little thing I have to mention: As far as I know, dashup is not available and finished yet, but your blog post sounds more like dashup is already created and finished. In my opinion, it would be better to be clear and set the tense to future if something is not ready yet. This would make it easier for readers to identify which features are finished and which are in progress. I will keep track on your progress. Sincerely, Christian Schweigel

Felix Hausberger (2018-10-10 10:01:29)

Dear Christian, thanks for this information. We edited the blog accordingly.

Midterm Summary – dashup (2018-12-22 14:50:43)

[...] Project vision [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:10)

[...] Project vision [...]

Distribution Of Roles, Technology Stack And More... (2018-10-14 12:30)



Quick introduction to Rational Unified Process (RUP)

Many of you should have heard of software development models like waterfall or scrum. RUP is probably the new state of the art way of managing projects in the software development industry. What makes it so special, is the considerable amount of roles, that are being specified in this model.

Generally, you distinguish between **breadth view** and **depth view** roles. Whereas the breadth view type of personality became familiar with keeping the overall focus while working quickly, intuitively and resiliently to change, a typical depth view worker puts much more effort into detail resulting in high-quality work.

In order to find out, which kind of personality each individual belongs to, Myers-Briggs Type Indicator (MBTI) personality types give a better understanding of eachs characteristics, based on a comparison between **Intuitives** and **Sensors** as well as **Judgers** and **Perceivers**. For further information, you can experience a deeper insight into RUP using [\[1\]this link](#).

We as team dashup put much effort into getting to know and specifying our own personalities in order to distribute the RUP roles accordingly to achieve the best possible outcome.

Distribution of roles

Joshua Schulz

Felix Hausberger

Raphael Müßeler

Sven Leonhard

Requirements specifier

- Responsible for requirement use cases
- Communicating requirements to the project manager
- Publishes release notes on wordpress website

Raphael Müßeler

Software Architect/Designer

- Decides on the technology stack
- Central person of contact for questions on the technology
- Analysis runtimes and technology requirements for a single set of use cases

Joshua Schulz

Felix Hausberger

Raphael Müßeler

Sven Leonhard

Implementer

- Implements use cases according to requirements
- Documents finished features and progress

Joshua Schulz

Felix Hausberger

Raphael Müßeler

Sven Leonhard

Test Designer

- Implements automated tests for features that are still in the validation
- Runs the tests and approves inconsistencies

Joshua Schulz

Deployment Manager

- Oversees deployment for all deployment units
- Administration of web application and database

Felix Hausberger

Raphael Müßeler

Project Manager

- Tracks project progress
- Responsible for sprint planning
- Delegates tasks and change requests
- Ensures high-quality standards

Felix Hausberger

Tool Specialist

- Administers project tools
- Creates guidelines for using the tools

Sven Leonhard

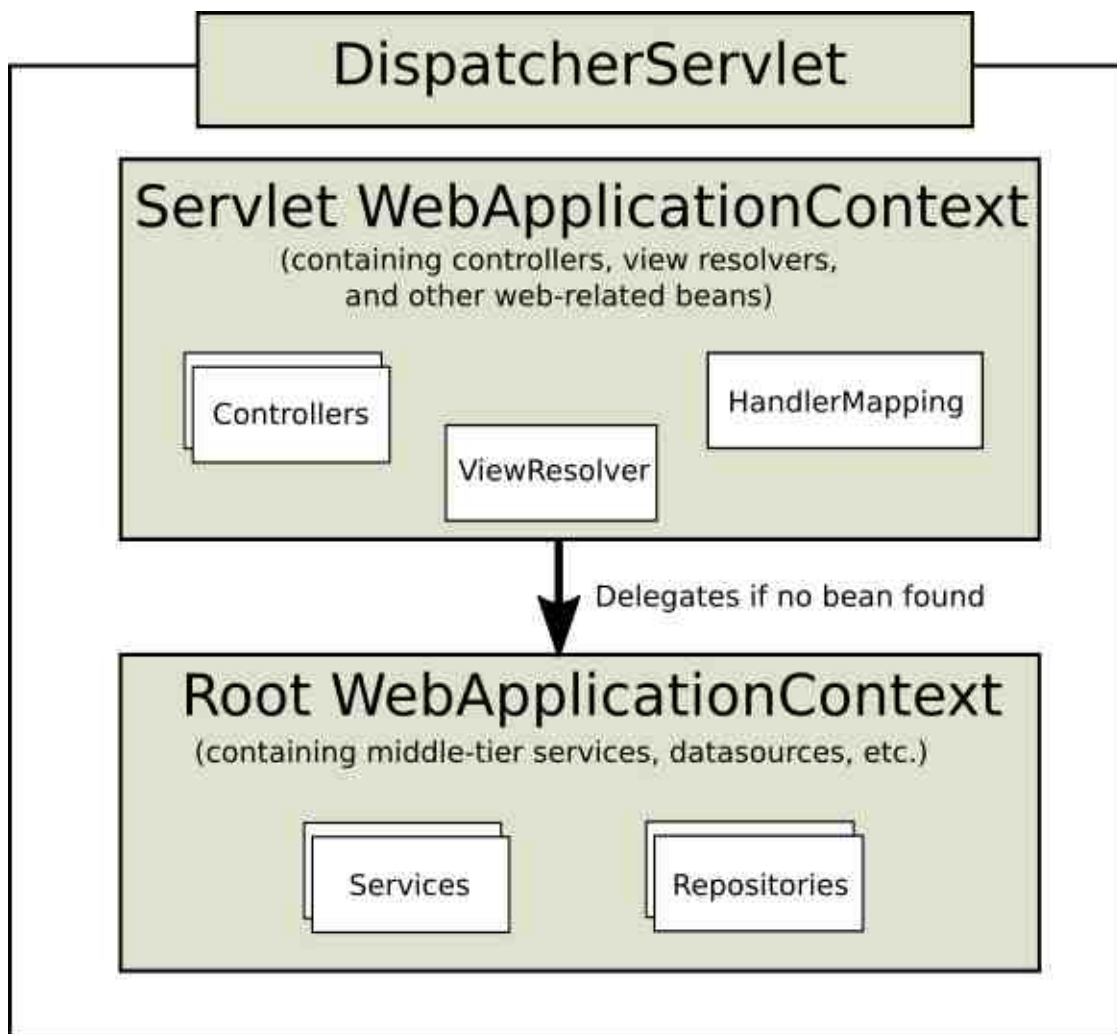
Configuration/Change Control Manager

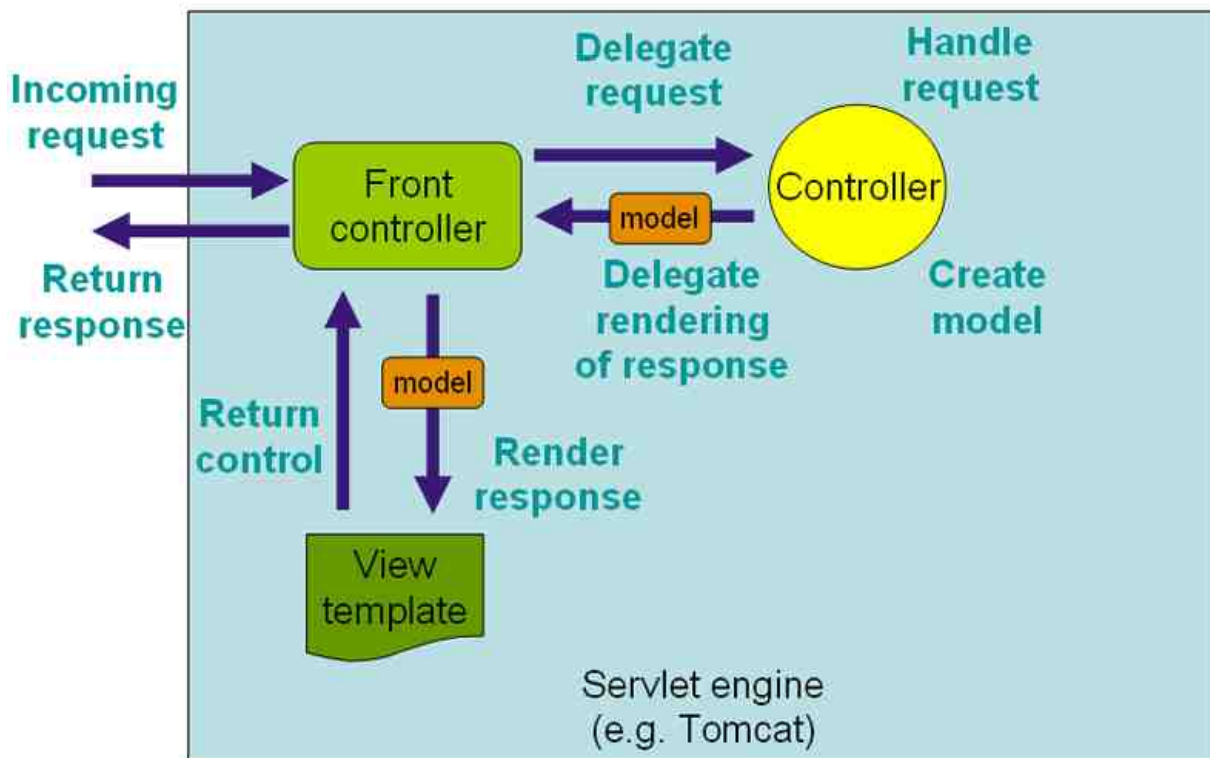
- Creates deployment units and reports on configuration
- Reviews and manages change requests by community

Technology Stack

Spring Web MVC

As nearly all of us are passionate Java Developers, we will use Spring MVC as our central architecture for dashup. See the pictures below, to get a deeper insight into the architecture itself, or visit the official [2]Spring Web MVC documentation.





Polymer

Polymer is an open source library from Google to create fast and lightweight web components. We will make use of polymer for frontend development purposes.

JUnit5 & Selenium

With both of them being strong testing frameworks, we use JUnit5 for unit and smoke testing, whereas selenium tests are of course used for UI tests.

MySQL

Of course, in a successful software development project, a database connection should not be missing. We decided to use a classical MySQL database to store our data in.

Maven

Maven as a software project management and comprehension tool gives us the opportunity to automate project's build, reporting, and documentation. It is quite essential in every bigger Java project, which is why it is necessary for us to make use of it.

Jenkins/Blue Ocean

Jenkins is a continuous integration tool, written in Java, and gives us the opportunity, together with Blue Ocean to build pipelines for automated building, testing and deployment simply and fast. Have a [3]look!

Sonar

Sonar's dashboard gives an overview of the current code situation with in-depth metrics. Serious errors and grievances can be detected quickly. The dashboard is public and can be accessed [4]here.

YouTrack

JetBrains offers a big amount of tools for programmers, among them YouTrack, an easy to use project management tool. Integrated into our web server, you can receive insights into our current project progress behind [5]this link.

IntelliJ IDEA

Besides common IDEs like Eclipse, we decided to use new innovation driving environments. In line with YouTrack, IntelliJ IDEA is another flagship of the JetBrains software range. As there are YouTrack integration plugins available for IntelliJ, we ensured, that all internal processes run as automated as possible.

Git

Git provides a proper version control system, which makes it possible to differentiate between different development branches. Once finished with a specific feature, you can merge your progress to the master branch and simply push it to a remote repository. Our code is completely open source, just visit [6]GitHub.

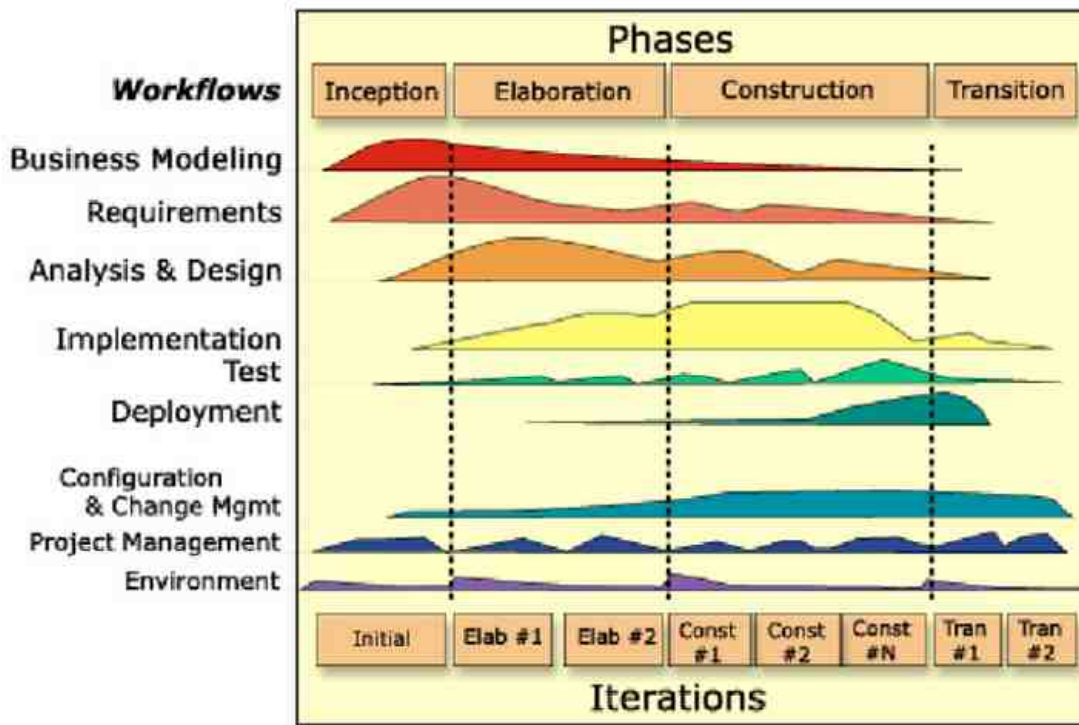
SonarCloud & Codacy

To ensure high code quality and coverage, we use both SonarCloud and Codacy to give us continuous reporting on our recent commits and advice on how to improve our coding.

We have just listed the most important technologies, that we are going to use. Of course, there are others like jQuery, AJAX, Materialize and so forth, who obviously will always be included in every web application project!

Project organization

According to RUP, we will divide our project into four main workflows *Inception*, *Elaboration*, *Construction*, and *Transition*.



Each issue in YouTrack will be tagged by a phase (seen in the picture above), workflow, use case and of course the sprint. As well, continuous time measuring plays a great role in our project. Therefore, we thought about publishing [7]burndown charts, but generally, we will make use of the most functions, that YouTrack can offer us.

If you have any questions left regarding today's blog post, let us know in the comment section below. Stay tuned!

1. <https://www.ibm.com/developerworks/rational/library/apr05/crain/>
2. <https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>
3. <http://jenkins.rafael-muesseler.de/job/dashup/>
4. <https://sonarcloud.io/dashboard?id=dashup>
5. <https://youtrack.dashup.de/issues>

6. <https://github.com/raphaelmue/dashup>
7. <https://youtrack.dashup.de/reports/burndown/123-5>

Team EventLAB (2018-10-17 10:09:15)

Hey Guys, we like how you explained your team structure, roles and technology in detail and we really enjoyed all the graphics you included to clarify your ideas. For now, there are no open questions from our side, but we will check out your next blog posts in great excitement. Keep up your good work, we are really looking forward to seeing the first prototype of your application! Cheers, Your EventLAB team

Felix Hausberger (2018-10-17 10:27:05)

We're glad to hear about your excitement. Our next blog post will be published soon, stay tuned!

Christian Schweigel (2018-10-17 10:13:29)

Hello dashup-Team, thanks for the insights in your team organization. I just have one thing to mention. As far as I know, in RUP it's better to name one person to be responsible for a specific role. This person doesn't need to do all by itself, but if there are too many responsibilities, the organization gets complicated. The chosen technologies look very promising, I'm looking forward to see your progress. Sincerely Christian Schweigel@digiWill

Felix Hausberger (2018-10-17 10:37:26)

Dear Christian, the reason why we listed the whole team for the roles Requirements specifier, Implementer and Test Designer is rooted in our idea of what our application will consist of. As all microservices, that we are going to implement, are mostly independent from each other, each of us will have one's own use case being implemented as a microservice. This means, that project structure will not get complicated as complexity is restricted to each developer. In order to make project management more efficient, we will use two project managers, one representing the product owner (Raphael Müßeler) and the other being the scrum master (Me, Felix Hausberger). Your sincerely, Felix Hausberger (Team dashup)

Malaber (2018-10-18 08:23:52)

Hey Guys, While reading your blog post I haven't seen any major things missing. I think it is great that you gave a quick summary of what RUP is and how it works for people that don't know it. Also as you stated who does what in great detail, everyone should be able to exactly ask the right person if they have an issue with dashup. One little thing I have is, that maybe you could make the link to your YouTrack more prominent, it took me a few seconds to find it. But that is a really minor point ;). Keep up the great work! Your Daniel from turnie.re

Anita Julitz (2018-11-28 15:24:20)

We like your blog, it has nice content, Many Thanks.

Midterm Summary – dashup (2018-12-22 14:50:44)

[...] Distribution of roles, Technology Stack, RUP [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:12)

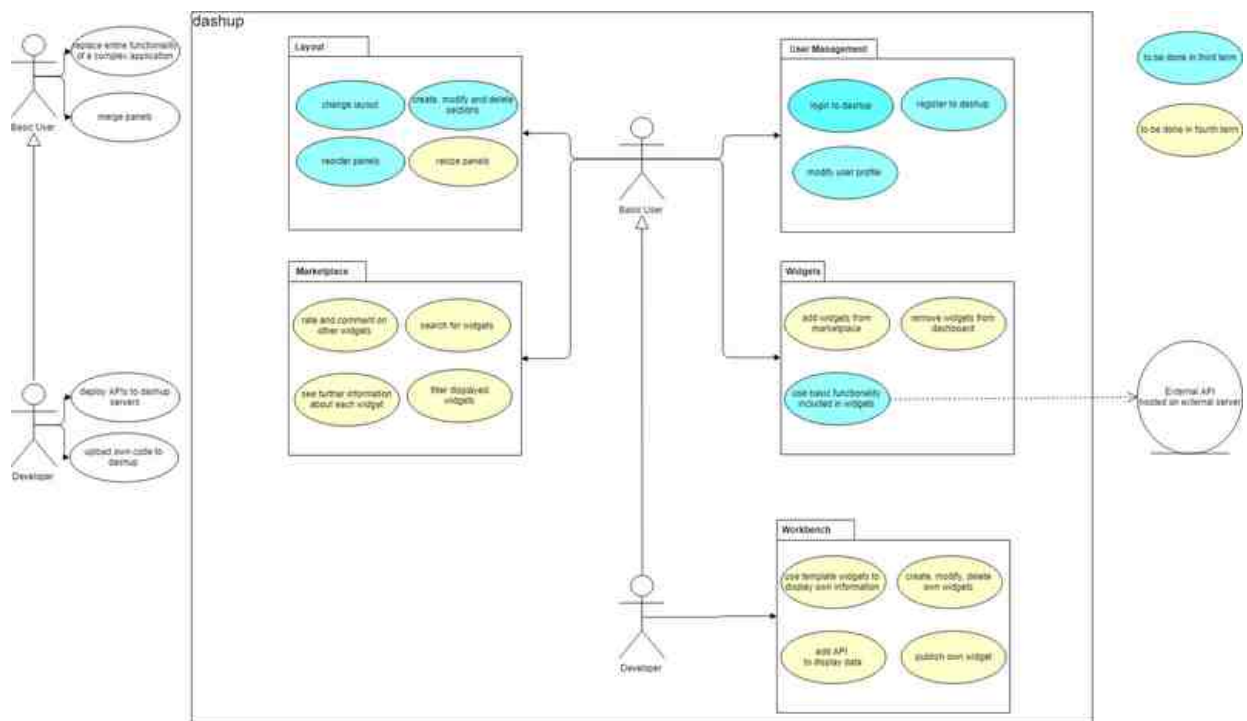
[...] Distribution of roles, Technology Stack, RUP [...]



Hey there! Have a look at our [1]software requirements specification!

This specification will be updated on a regular basis during the project workflow. If you think anything is missing in our specification, any features, that you would like to be added to dashup, let us know!

Furthermore, we created a [2]use case diagram, giving you a basic overview of the user characteristics, features, constraints, associations, and dependencies within the project scope. Non-functional requirements are put outside the system boundaries.



Stay tuned for our next blog post about the use case specifications!

1. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/srs/SRS.md>
2. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/srs/UCD.png>

Tennsikönig (2018-10-22 15:08:45)

Hey dashup, im looking forward when the Software requirements specification is completed. In the first version it looks very finished. Maybe you could have linked the UseCase Diagram so i can view it in a bigger version. Greeting Tenniskönig

Felix Hausberger (2018-10-24 08:09:02)

Hey Tennsikönig, thanks you for your feedback. We edited the blog post accordingly, now providing a link to the UCD in a bigger version. Your sincerely Felix - Team dashup

learnityourselfdhbw (2018-10-23 20:29:26)

Your SRS was very pleasant to read and very detailed. Good job there +1. However, I did not understand who you actually mean by the actor "developer" in your UC-Diagram. Can you apply as a "developer", or are users generally allowed to use the functionalities of a "developer" if they want to? Or to put the question differently: How can I become a so-called "developer"? Furthermore, you need to rescale your UC-diagram in your blog post or provide a link to the corresponding image. Sincerely, JannikAdam@learnityourself

Felix Hausberger (2018-10-24 08:31:46)

Hey Jannik, the developer user is intended for those users, who consider the provided microservices as insufficient for their daily use case scenarios and would appreciate to integrate their own microservices into dashup. Therefore they can create their own panels, based on predefined panel templates or a diverse pool of panel components that dashup provides. Lateron, they simply add data and functionality for these components by providing their personal REST Service, whose responses must comply to a specific format. Furthermore, we added a link for the UCD in a bigger version. ;) Your sincerely Felix - Team dashup

learnityyourselfdhw (2018-10-24 09:30:37)

Hello Felix, thanks for making me understand the actor "developer" better :) However, how can a user "upgrade" to be a developer? kind regards, JannikAdam@learnityyourself

Felix Hausberger (2018-10-28 15:45:04)

Hey LEARNITYYOURSELFHDHW, you are automatically a developer, if you have the knowledge to develop your own microservice. But our system makes no difference between a normal user and a developer. But the normal user probably won't use development functionalities as he can't develop. Your sincerely Felix from Team dashup

Midterm Summary - dashup (2018-12-22 14:50:46)

[...] Software Requirements Specification [...]

Indeed, it was a very adventurous journey - dashup (2019-06-17 11:29:14)

[...] Software Requirements Specification [...]

Use Case Specifiactions (2018-10-28 23:38)



After the software requirements specification has come into effect since last week, we were working hard on delivering the corresponding use case specifications. Don't be afraid to have a closer look:

Layout

- [1]Change Layout
- [2]Change Panel Structure

User Management

- [3]Change Profile
- [4]Login / Register

Marketplace

- [5]Marketplace

Workbench

- [6]Workbench

Widgets

- [7]Widget

In the [8]SRS there are references to all items in the list. See you next week!

1. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_layout/UCS_change_layout.md
2. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_panel_structure/UCS_change_panel_structure.md
3. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/change_profile/UCS_change_profile.md
4. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/login_register/UCS_login_register.md
5. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/marketplace/UCS_marketplace.md
6. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/workbench/UCS_workbench.md
7. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/widgets/widget/UCS_widget.md
8. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/srs/SRS.md>

Team DigiWill (2018-10-30 19:09:08)

Hey there, I really like your use case documentation! Your mock ups they give a great overview over the planned interfaces and I really like the look and feel of them! I did note a few things that you might want to look at: The versioning of the SRS is kind of unnecessary when using git. The tables in all your markdown files look a little weird, maybe you should use actual markdown tables instead of html ones ;) Change Layout: - Spelling mistake "This use-case allows -to-(the) user" - In your brief description you say "First the user[...]" but never mention what else they can do - color and font might be considered the style of the layout but "change basic layouts of the board, e.g. the main color or font" doesn't really make sense "change the basic appearance of the board, e.g. the main color or font" would make more sense - the activity diagram could be more percise. Display which settings, adjust which settings in which way? (slider/text field/color picker/...) - What's the difference between "Display dashboard with adjusted layout" and "Display dashboard with adjusted settings"? Is there a difference? If not why do you have it twice. If the user can only adjust the color and font how is the layout affected? Change Panel Strcuture: - sounds better: "fill these sections -by-(with) the panels" - when deleting a section you could ask the user to confirm the deletion - why is the renaming of sections not in your activit diagram but referenced in 2.1.3 - word missplayed: "The user can exit the change mode at all (times) without saving -times-" Marketplace: - Really nice diagram! - The Text in the diagram could be a little bigger for readability - What action can the user take in "navigate through result set" to leave the use case? (The line doesn't have text) - "navigate through result set" is a little confusing since there are so many arrows interacting with it Overall your use case documentation is really good and I'm happy seeing your project take shape! Best regards Jannik@DigiWill

Joshua Schulz (2018-10-31 09:08:35)

Hey Jannik, thanks for your helpful tips. I tried to realize all of them in our UCS and improved the activity diagrams. But there are some questions regarding some tips: "- word missplayed: 'The user can exit the change mode at all (times) without saving -times-': I think this correction is messing up the sense of this sentence, but i tried to find a more precise expression: "The user can leave the change mode at any time without saving". "- when deleting a section you could ask the user to confirm the deletion": We included the possibility to roll back the changes without changing. So we think it would be some kind of annoying if you have to apply this stuff twice. This is just my thought at this point. I hope you can understand my doubts on some of your points, but all in all your extensive feedback was really useful to improve your UCS. Best regards Joshua

Felix Hausberger (2018-10-31 10:32:10)

Hey Jannik, in addition to joshuas comments, I would like to add, that we added a button to navigate back to the main menu thanks to your review. This is now the default action to end the use case. The "navigate through result set" action is indeed a little bit complicated, as it is the action, from which other actions are triggered. Your sincerely, Felix from team dashup

EventLAB Team (2018-10-31 08:51:46)

Hi Dashup Team, you've really done a great job creating the specifications for your three use cases. Your mock-ups provide a good overview about your user interface and clearly specify the actions that can be taken on each screen. Nevertheless, we have got some tips for improvement on your work: In the second mock-up of your Marketplace Use Case, the description of each micro-service is: "This is a cool contact app, which has a very nice description, although it has only 3.5 stars of 5." This should be changed for each service accordingly (contacts, sharevalue, ...) Regarding the requirements, you write: "a default amount of panels provided by the community should be available". How many micro services do you plan to provide yourself? And is there already any kind of description how micro services can be implemented? What kind of interface will be used? Best regards, Your EventLAB Team

Felix Hausberger (2018-10-31 10:38:09)

Dear Team EventLAB, thanks for your feedback. We took your advise regarding the mockup into account. In Addition, there will be a default amount of minimum 4 microservices given by dashup natively (calendar, shopping list, ...). The rest of all microservices are provided by the community as a REST API, that they specify when uploading custom panels to dashup (another use case that will be documented soon) Your sincerely Felix

MNZ-Team (2018-10-31 08:52:08)

Hey guys, we really like your use case specifications and mock ups. Maybe you think about changing the way you like to implement the rating function. We think a simple selection in 1 - 5 stars is more user-friendly as a slide bar. Best regards MNZ-Team.

Felix Hausberger (2018-10-31 10:39:38)

Hey Team MNZ, thanks for your feedback, we will take your advise into account and adapt the mockup and specification regarding this topic. Your sincerely Felix from Team dashup

Midterm Summary – dashup (2018-12-22 14:50:48)

[...] Use Case Specifications [...]

Welcome back! – dashup (2019-04-07 12:42:32)

[...] Now that we have the new scope, we defined all use cases and we specified them in the known pattern. You can find the updated specifications here. [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:15)

[...] Use Case Specifications [...]

1.2 November

Gherkin Feature Files And Selenium Testing (2018-11-04 17:41)



Dear community,

we are happy to announce that we just created our .feature files using Gherkin. These files use human-readable language and are later on used to run our selenium tests on. This is a major step towards behavior driven development.

We created the following feature files:

Layout

- [1]Change Layout
- [2]Change Panel Structure

User Management

- [3]Change Profile
- [4]Login / Register

Marketplace

- [5]Marketplace

Workbench

- [6]Workbench

Note that a narrative for the widgets use case cannot be written. The logic behind a widget for data processing is hosted on an external server and provided by its consumable web service. As most widgets from dashup are developed by its community and users, it is uncertain how a widget looks like and what its purpose is. This is why the functionality of a widget won't get tested. The web components widgets are built with and their interaction with other web components will be tested in internal prototypes though.

Furthermore, IntelliJ offers as always great highlighting and formatting of the gherkin code! Have a look:

```
Scenario Outline: Filter result set
  Given User is located on marketplace menu
  When User clicks on filter icon
  Then Filter menu will pop up
  When User changes filter "<filter>" to value "<value>"
  Then Result set will be restricted to all widgets that match "<filter>" equals to "<value>"

Examples:
| filter           | value           |
| rating           | 4.0             |
| categories       | money           |
| tags             | productivity     |
| publisher        | dashup          |
| date of publication | 04/11/2018      |
```

The tests, that are specified using gherkin, are implemented as Selenium tests and run automatically before each build. As our continuous integration tool, we use [7]Jenkins and Blue Ocean, which makes it possible to combine automated building, testing, and deployment in a pipeline. See the videos of some extracts of our selenium tests below:

3rd Term

<https://youtu.be/Fs-f6Olf9os>

4th Term

<https://www.youtube.com/watch?v=NSU4QPSm5cY>

For more detailed information, please read through the [8]test logs or have a look at any build and click on "Tests" at the top right corner on out [9]Jenkins.

Hope you liked today's blog post, stay tuned!

1. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_layout/narratives/change_layout.feature
2. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_panel_structure/narratives/change_panel_structure.feature
3. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/change_profile/narratives/change_profile.feature
4. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/login_register/narratives/login_register.feature
5. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/marketplace/narratives/marketplace.feature>
6. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/workbench/narratives/workbench.feature>
7. <http://jenkins.rafael-muesseler.de/job/dashup/>
8. https://github.com/raphaelmue/dashup/tree/master/docs/logs/surfire_test_logs
9. <https://jenkins.rafael-muesseler.de/blue/organizations/jenkins/dashup/activity/>

EventLAB Team (2018-11-05 17:23:44)

Hey guys, we really liked the work you did this week. The feature files you created are very detailed and it was good that you defined examples to test different settings of your scenario. If we are right in all scenarios you have the requirement that an user has to be logged in to do something. Maybe it would also be interesting to create a test that handles the fact of an user that is not logged in. That for the user some of the features or sides are not available. It would be great if you think about this point. All in all you did an extensive job and we are interested to see your progress in the next weeks. Keep up the good work!
Your EventLAB team

Felix Hausberger (2018-11-07 09:49:09)

Dear team EventLab, great idea, we added these scenarios to the feature files :) Your sincerely, Felix (Team dashup)

MNZ Team (2018-11-07 08:57:28)

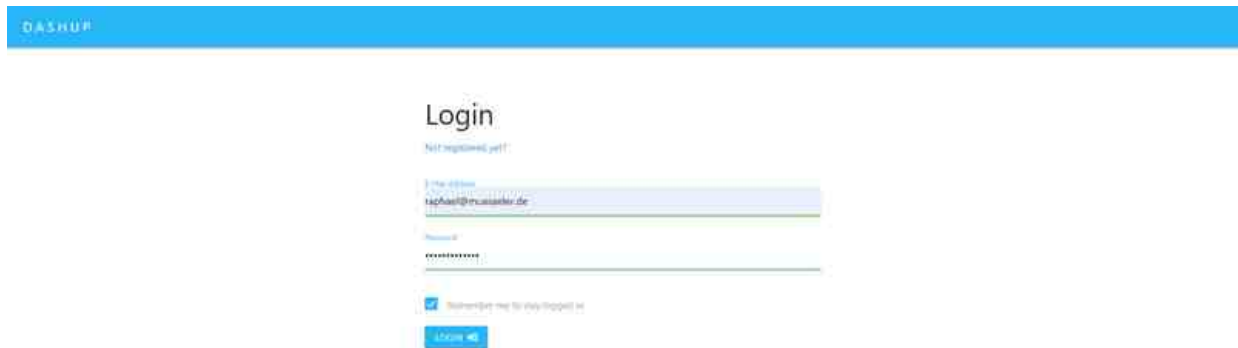
Hey guys, your feature files are very detailed and well thought through. We like that you want to give your users the opportunity to customize so many things by reordering and resizing. Amazing work - keep it up!

Midterm Summary – dashup (2018-12-22 14:50:50)

[...] Gherkin Feature Files And Selenium Testing [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:17)

[...] Gherkin Feature Files And Selenium Testing [...]



In this week's episode of dashup, we present you several achievements, that we managed to reach this week.

First, we started implementing the kernel of dashup by setting up the Spring MVC architecture. Furthermore, maven now helps us to organize our project structure better with modules and manages required dependencies for us.

We implemented basic controllers and request handlers, especially the main controller delegating all incoming requests to the corresponding controllers (like for the central dashboard, the entry handling, i.e. login and registration, error handling, etc.).

Secondly, we set up our database and connected it to the application, which made it possible for us to implement the login scenario.

In addition, we included cucumber dependencies and implemented step definitions for some scenarios already.

Furthermore, we managed to create several JSP Views. Have a look at our login page:

Login

[Not registered yet?](#)

E-Mail Address

testuser@gmail.com

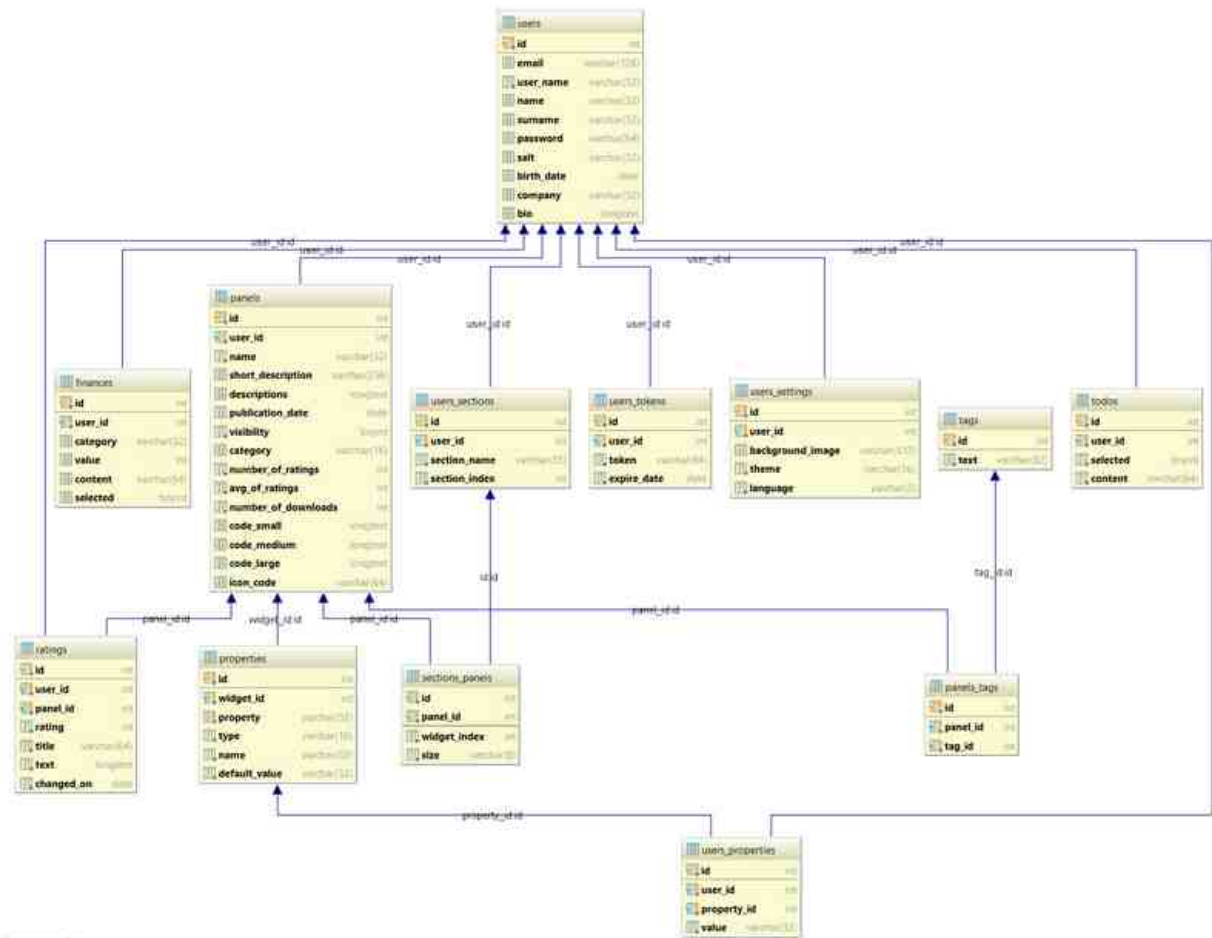
Password



Remember me to stay logged in

LOGIN ➔

As well, we set up the [1]database scheme shown below.



And now here comes the most exciting point. We tried to create a class diagram for our application and recognized, that this project definitely opens up a new dimension of complexity (at least for us).



And this [2]class diagram is not even finished yet, so stay tuned ;)

Hope you liked today's blog post, see you soon!

Your dashup core developer team

1. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/sad/img/database_scheme.png
2. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/sad/img/dashup_class_diagram_mvc.png

flashcardcommunity (2018-11-12 20:59:54)

Hello Dashup, your progress sounds really good so far. Your login page looks very nice and clean. The only point I would like to mention is your grey font on a grey/black background. Please make sure that you adjust those colors correctly to avoid a poor readability. Your database schema is well structured. You could add the datatypes to your fields, but your idea is easy to understand. One question I have about your User: Will one only be able to have a first-last-name-combination or a simple username as well? I'm interested in finding out more about your application, so I'm looking forward to your class diagramm. Regards Benjamin@FlashCardCommunity

7lofficial (2018-11-14 09:10:21)

Hello Benjamin, thank you for your Feedback. We will rework the page for a better readability. In our database schema we deliberately renounced datatypes because in our view a schema comes without them. An unique username is not planned because it does not bring an additional value to our application. Best Regards, Sven (dashup team)

mljenne (2018-11-13 22:22:54)

Hello Dashup, your login page is quite nice and look good. The class diagram is also easy to understand and clean structured. I like they way you'll implement this. I'm very interested in finding out more about your project and I'm looking forward to the further course. P.S. in the first section, the letter r is missing in the word "structure". Kind regards, FyF-Team

Midterm Summary – dashup (2018-12-22 14:50:52)

[...] DB Scheme And Class Diagram [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:19)

[...] DB Scheme And Class Diagram [...]

Scrumming With YouTrack And Project Management (2018-11-18 09:52)



Hey there,

today we would like to introduce you into our scrumming tool [1]YouTrack. Our sprints are running on a weekly basis. This is how a typical sprint backlog item looks like:

Implement cross browser testing for selenium tests

Test dashup application in at least chrome, firefox and edge.

Click to attach, drag files, or press Ctrl-V to paste an image

Activity settings

Joshua Schultz committed changes 25 Mar 2019 10:31
 cucumber DASHUP-97 implemented min-profiles to enable cross browser testing

25 Mar 2019 10:31
 cucumber DASHUP-97 encapsulated driver setup

25 Mar 2019 10:31
 cucumber DASHUP-97 adapted jenkins file to run cross browser testing

Joshua Schultz 25 Mar 2019 10:32
 Status: Open → In Progress

JS Joshua Schultz commented 25 Mar 2019 14:09
 Only Chrome and Firefox possible due to linux OS on Jenkins
 Status: In Progress → To Verify

Joshua Schultz committed changes 25 Mar 2019 14:45
 Infrastructure: 2 hours for 25 Mar 2019
 Spent time: 7 → 2h

Joshua Schultz 25 Mar 2019 12:44
 Received date: None → 25 Mar 2019

Felix Hausberger 14 hours ago
 Time Tracker: No time tracker → 25 Apr 2019 08:30

Write a comment, @mention people

Attach file... Get help attaching images, files and files. Mention *

Add comment Visible to: Same as issue * X Add spent time **Add commit**

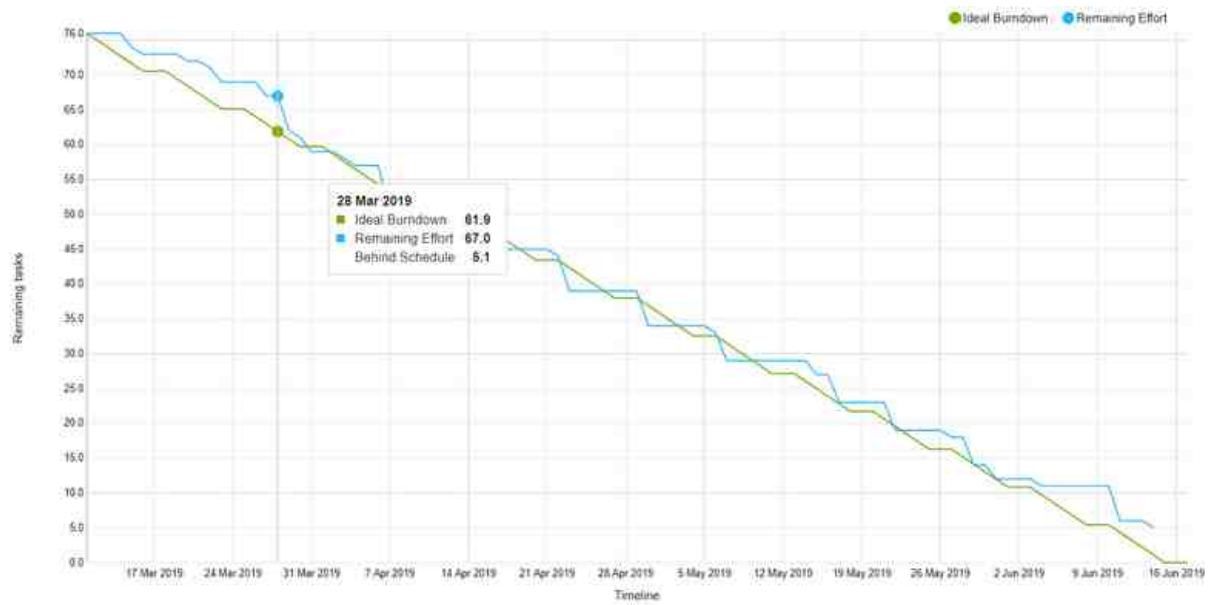
Metadata Sidebar:

Project	dashup
Priority	Normal
Type	Task
Status	Done
Assignee	Joshua Schultz
Phase	Construction
Sprints	Practical Phase
Spent time	2h
Estimation	4h
Workflow	Test
Use Case	No use case
Time Tracker	25 Apr 2019 08:30

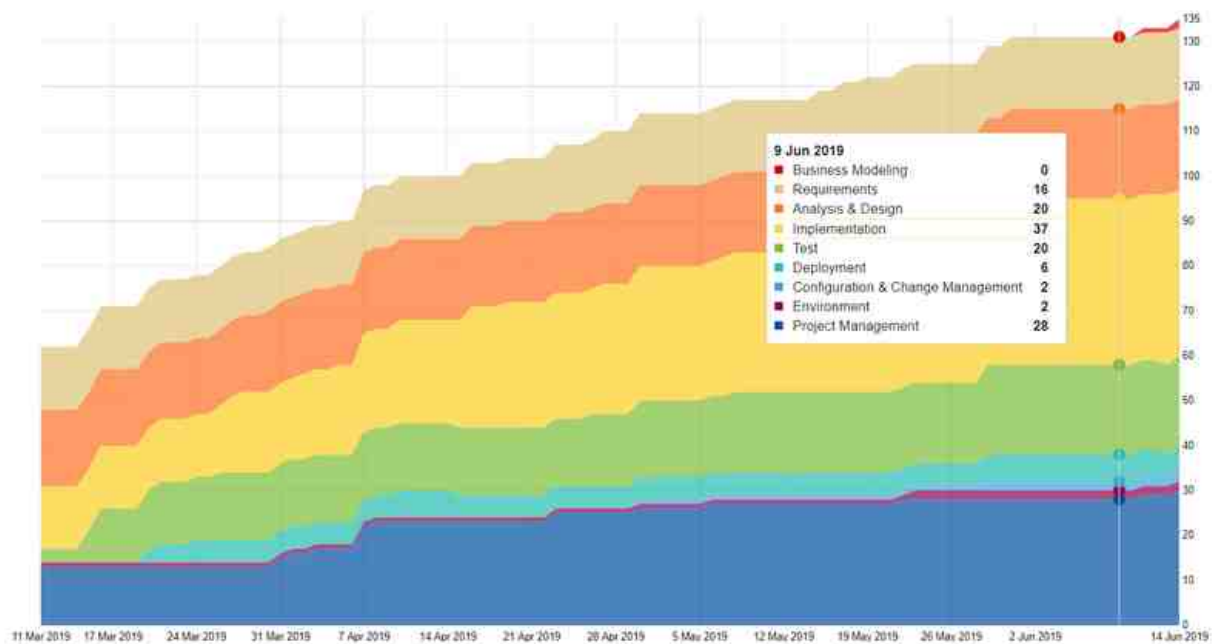
Watchers: Watch issue
 Boards: Add to board

Each item is linked to an assignee, a phase and a workflow according to RUP, and furthermore a time estimation, the sprint, and the use case. These information help us to generate diagrams to better track our project progress in YouTrack. Have a look:

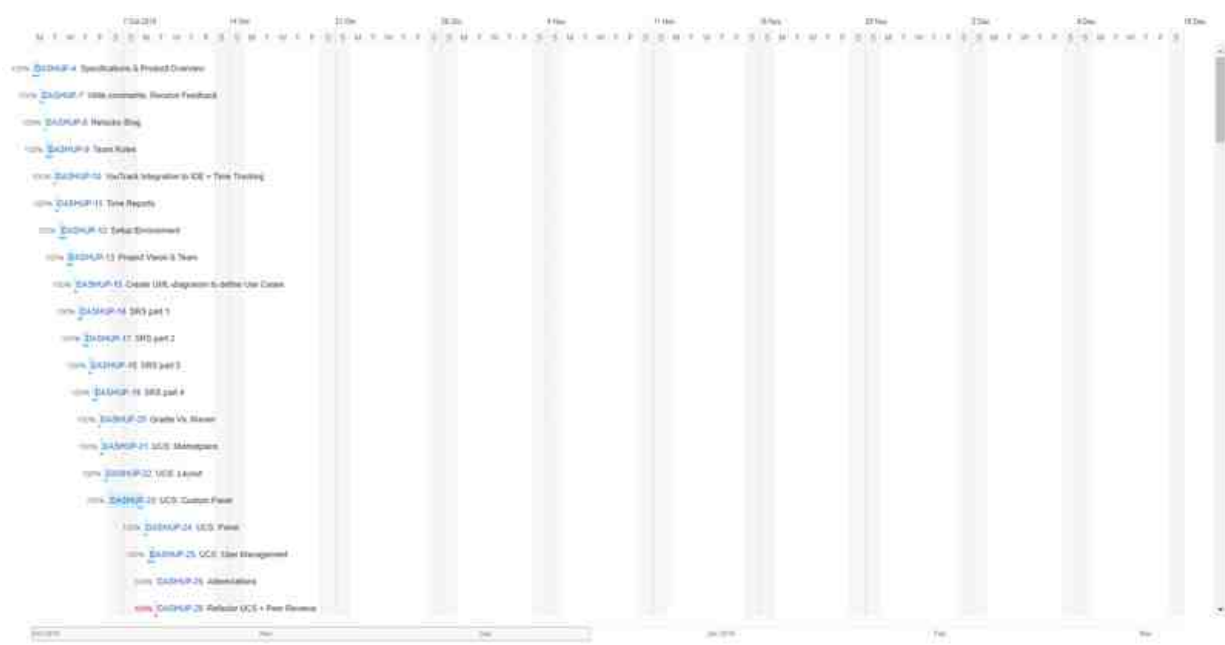
Burndown Chart (Issues)



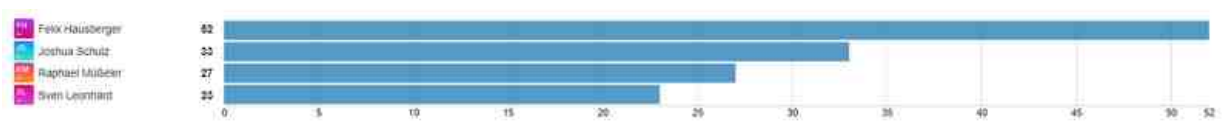
Cumulative Flow (Workflow)



Gantt Chart



Issues Per Assignee



Time Report Per Person

PROJECTS , GROUP BY ASSIGNEE		TIME ESTIMATED	TIME SPENT
felix.hausberger		267h	284h10m
DASHUP	dashup		284h10m
joshua.schulz		163h32m	180h2m
DASHUP	dashup		180h2m
raphael.muesseler		169h5m	183h5m
DASHUP	dashup		183h5m
sven.leonhard		140h15m	191h5m
DASHUP	dashup		191h5m

Total time spent: **838h22m**
out of 739h52m

Time Report Per Phase

USERS , GROUP BY PHASE		TIME ESTIMATED	TIME SPENT
Construction		660h45m	762h38m
Felix Hausberger			257h55m
Joshua Schulz			183h43m
Raphael Müßeler			158h
Sven Leonhard			163h
Elaboration		31h2m	31h54m
Felix Hausberger			11h15m
Joshua Schulz			5h49m
Raphael Müßeler			9h
Sven Leonhard			5h50m
Inception		39h35m	33h20m
Felix Hausberger			15h15m
Joshua Schulz			9h
Raphael Müßeler			3h5m
Sven Leonhard			6h
Transition		8h30m	10h30m
Felix Hausberger			2h30m
Sven Leonhard			8h

Total time spent: **838h22m**
out of 739h52m

Time Report Per Use Case

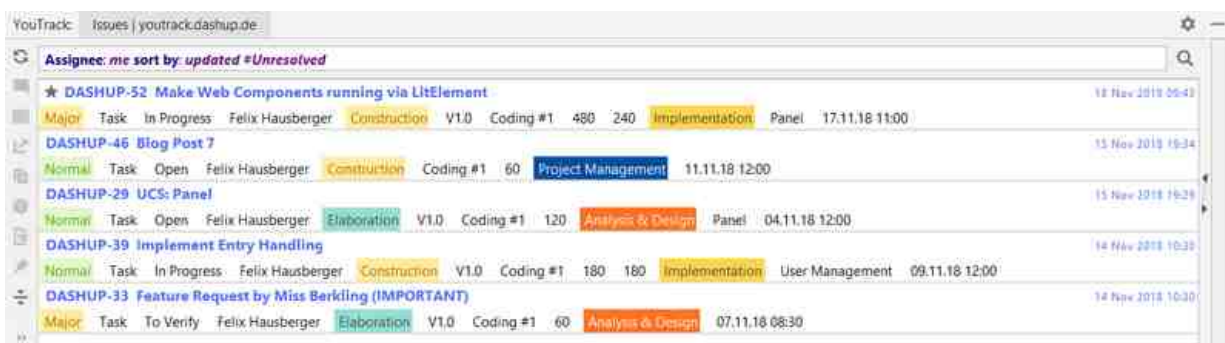
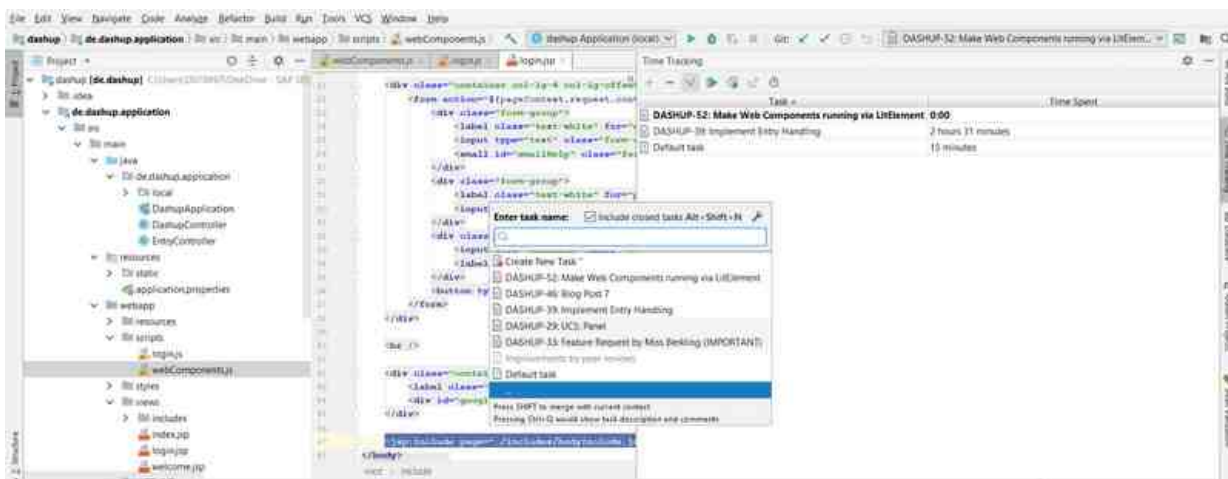
USERS	GROUP BY USE CASE	TIME ESTIMATED	TIME SPENT
Layout		91h47m	151h37m
	Felix Hausberger		23h15m
	Joshua Schulz		32h2m
	Raphael Müßeler		4h
	Sven Leonhard		92h20m
Marketplace		126h	121h
	Felix Hausberger		16h
	Joshua Schulz		64h
	Raphael Müßeler		3h
	Sven Leonhard		38h
No use case		260h50m	260h20m
	Felix Hausberger		86h
	Joshua Schulz		77h15m
	Raphael Müßeler		48h5m
	Sven Leonhard		49h
User Management		52h45m	64h15m
	Felix Hausberger		16h
	Joshua Schulz		19h15m
	Raphael Müßeler		29h
Widgets		139h	168h40m
	Felix Hausberger		132h40m
	Joshua Schulz		6h
	Raphael Müßeler		27h
	Sven Leonhard		3h
Workbench		69h30m	72h30m
	Felix Hausberger		13h
	Raphael Müßeler		59h
	Sven Leonhard		30m
		Total time spent:	838h22m
			out of 739h52m

Time Report Per Workflow

USERS - GROUP BY WORKFLOW	TIME ESTIMATED	TIME SPENT
Analysis & Design	46h17m	46h9m
Felix Hausberger		16h15m
Joshua Schutz		5h4m
Raphael Mùßeler		19h
Sven Leonhard		5h50m
Business Modeling	10h	11h30m
Felix Hausberger		3h30m
Sven Leonhard		8h
Configuration & Change Management	20h	27h
Felix Hausberger		9h
Sven Leonhard		18h
Deployment	35h	35h30m
Joshua Schutz		4h30m
Raphael Mùßeler		23h
Sven Leonhard		8h
Environment	8h	8h
Felix Hausberger		4h
Raphael Mùßeler		4h
Implementation	343h15m	429h10m
Felix Hausberger		134h25m
Joshua Schutz		69h45m
Raphael Mùßeler		96h
Sven Leonhard		129h
Project Management	119h15m	119h45m
Felix Hausberger		104h
Joshua Schutz		3h45m
Raphael Mùßeler		12h
Requirements	46h5m	38h50m
Felix Hausberger		13h45m
Joshua Schutz		9h
Raphael Mùßeler		10h5m
Sven Leonhard		6h
Test	112h	122h28m
Felix Hausberger		2h
Joshua Schutz		105h28m
Raphael Mùßeler		6h
Sven Leonhard		8h
Total time spent:		838h22m
		out of 730h52m

Have a closer look through all our reports [2]here.

Note that YouTrack does not just offer you excellent git integration, but even integration into our IntelliJ IDE. We do not have to switch between the tool and the IDE in order to create new tasks. Even time tracking works automatically.



We hope, that we were able to convince you of YouTrack as the only true scrumming tool out there. :)

If there are any questions left, feel free to contact us. See you!

1. <https://youtrack.dashup.de/issues>
2. <https://youtrack.dashup.de/reports>

MNZ Team (2018-11-20 22:57:51)

hi dashup-Team, like the teams you use YouTrack for your project, you integrated this tool very good into your IDE and workflow. In the burndown chart we got a good overview about your working progress. Also the Git integration in your IDE looks very good. keep up your work your MNZ-Team

Christian Schweigel (2018-11-21 08:43:10)

Hi dashup-Team, Your integration looks very good. I'm convinced that it will help you to become even more productive. The charts give a good overview how far you are in the development and how good your time management and planning is. Maybe you should add a link to your YouTrack board and your issue list directly in the blog entry, so it would be easier to access them. Keep going with your work. Sincerely, Christian@DigiWill

DHBWieWarsEssen (2018-11-22 09:41:05)

Howdy dashup-Team, thank you very much for your update. You did great work this week. I take it you haven't managed to generate a GANTT diagram, since it is not mentioned in your post. You might want to clarify that. Looking forward to seeing your next update. Best regards Luis@DHBWieWarsEssen

Joshua Partogi (2018-12-17 09:00:11)

How is it better than JIRA?

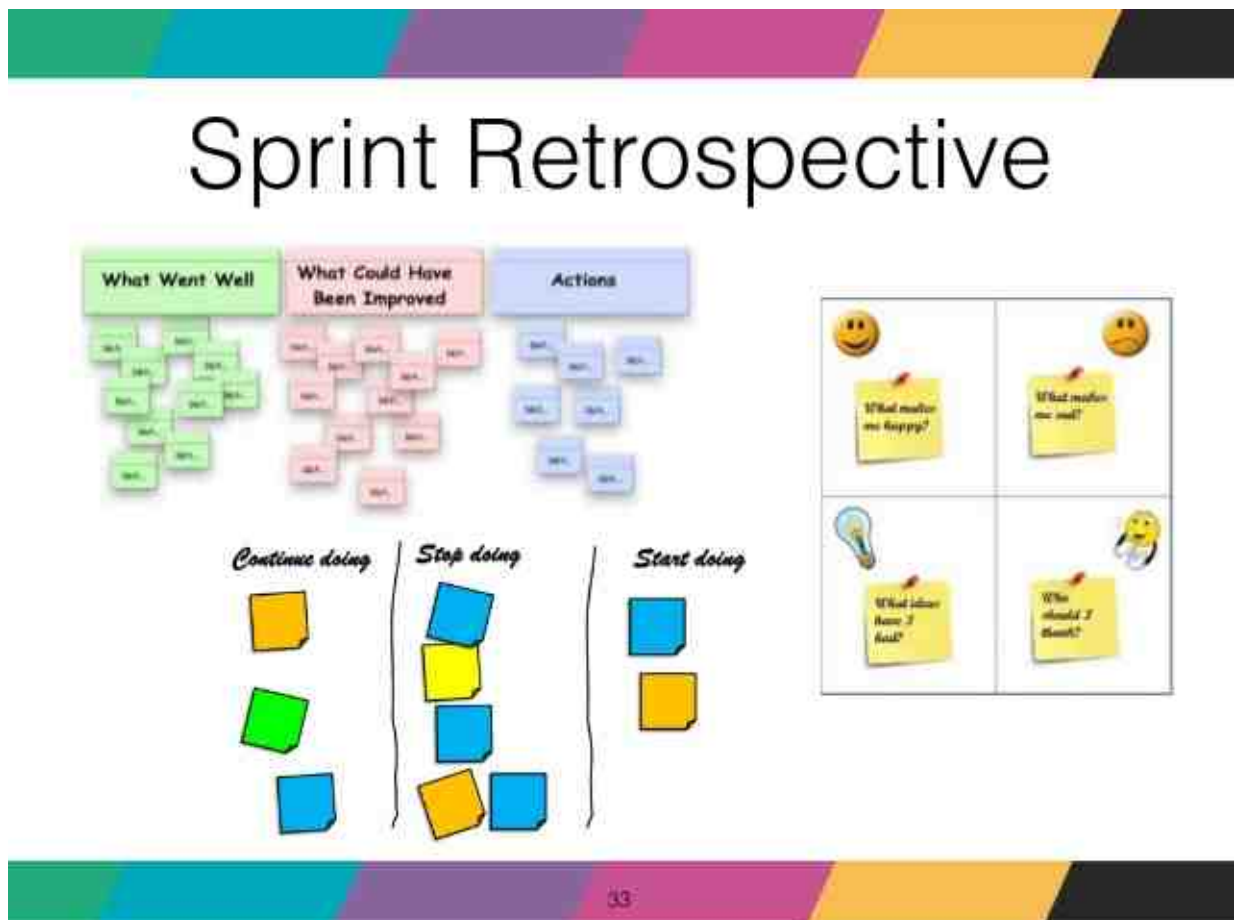
Midterm Summary – dashup (2018-12-22 14:50:53)

[...] Scrumming With YouTrack [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:20)

[...] Scrumming With YouTrack And Project Management [...]

Scrum Retrospective (2018-11-23 09:43)



Hey there,

this week we had a retrospective with an experienced moderator to help us to scrum even better and more efficient. We would like to introduce you to our results and possibilities for improvement.

First, we talked about things that went well in our team:

- **The idea:** We were able to find an interesting vision for our project very fast and everybody contributed to that vision. Because of that, we are working on an innovative and creative idea.
- **The organization:** The creation and distribution of tasks went quite well in our team. We were able to avoid duplicate work and unnecessary effort. The communication via YouTrack and other tools was effective and fast.
- **The punctuality:** Every piece of our work was completed on time and uploaded for review. To our team meetings, everyone was punctual so that we can start the work fast and easy.

Second, we tried to figure out things that did not go that well and needs to be improved in the future:

- **The design decisions:** We experienced some conflicts when deciding about technology, patterns, and designs we want to use and which not. There were very different options and some members did not really accept some of these opinions. This lead to a bad team atmosphere in which it is difficult to keep up the good work.
- **Empathy:** In our team, there were some misunderstandings regarding problems that appeared while working or planning. Some team members felt left alone with their problems.
- **The support:** In case there is a problem that needs to be solved by a team member the other members did not support him to the right degree. Some problems needed a long time to be solved, even though they could have been solved very fast with the help of a second team member.

Last points we talked about were possibilities to improve our scrumming to avoid the problems we had in the past:

- **More communication:** We will try to communicate more and more often to be aware of the different opinions and thoughts about certain decisions and tasks. First, this should lead to fewer conflicts and constructive decision making. Second, this should lead to a deeper understanding of the problems of the other members and the willingness to support them.
- **Fewer prejudices:** We need to get rid of some prejudices against certain technology and designs to consider every possibility and opinion in a neutral and constructive way. This should avoid mistrust and a bad team atmosphere and should state that every member and his options an equal and nice to have.

To conclude we can state that this retrospective worked out very well for us and we were able to find some necessary improvements to become a better scrum team.

Feel free to share your thoughts with us, by commenting on this blog post. We are looking forward to some constructive criticism and new ideas. See you!

MNZ Team (2018-11-28 08:53:47)

Hi Dashup-Team, I think you did a great job this week. I hope your Empathy and support problem could be solved. At the beginning of your text was a word repetition. ("...retrospective with a with an experienced moderator...") See you soon your MNZ-Team

Midterm Summary – dashup (2018-12-22 14:50:55)

[...] Scrum Retrospective [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:22)

[...] Scrum Retrospective [...]

1.3 December

MVC Architecture (2018-12-02 18:26)



Hey guys,

this week, we've been working on our Software Architecture Document, which you can check out [1]here. We also have rebuilt our [2]class diagram, because some things have changed:



Spring


We are using Spring MVC as our main framework. This allows us to create an interactive and dynamic Web Application, that uses the Model-View-Controller logic and it offers the possibility of coding in Java.

Database Access

As we are persisting our application data in a MySQL (or rather MariaDB) database, we found our own possibility to access the database. By dint of the *Database* class, we can access our database. It offers all CRUD actions as instance methods, in which the respective SQL statement is put together, sent to the database and finally, it returns the result. You can check out this framework [3]here.

Application

You can check out our application right [4]here. SSL support is coming soon.

We would love to hear your suggestions and comments, so feel free to comment. 

1. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/sad/SAD.md>
2. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/sad/img/dashup_class_diagram_mvc.png
3. <https://github.com/raphaelmue/dashup/blob/master/de.dashup.model/src/main/java/de/dashup/model/db/Database.java>
4. <http://dashup.de/login>

EventLAB Team (2018-12-04 10:17:40)

Hi guys, we have had a detailed look at your Software Architecture Documentation and your diagrams. You have done a great job and it's nice to see all the progress you've made. We are also happy to see that you could apply your acquired knowledge from our database course and created a beautiful entity relationship model! We have also got some tips for improvement of this week's blog post: It might be useful if you included the images in your blog post directly from your GitHub repository instead of uploading them to wordpress. Then you do not have to update them each time you change something. Regarding the grading criteria of this week, it might be useful if you described your chosen framework (spring) in your blog post and how you can easily generate a CRUD with a few commands using this tool. Finally, we would appreciate if you added a link to the current version of your application where we could try it out. A screenshot would also be nice to have. Keep up your good work! Best regards, Your EventLAB Team

Raphael Müßeler (2018-12-04 21:36:16)

Hi EventLAB, that you really much for your comment! We appreciate your remarks. We changed your enhancements and adjusted our blog entry. Kind regards – dashup Team

Team DigiWill (2018-12-05 09:03:40)

Hi, nice Blog post! - I think it would be beneficial to additionally include links to smaller versions of your UML diagram which don't have the methods / fields to get a better overview. - "MySQL (or rather MariaDB) database" why write MySQL if it's actually MariaDB? - I really like that you included a link to your Database class! - You are exposing your weather API key and probably shouldn't make the API call in the frontend but in the backend - In your SAD you wrote "Besides is the Controller and Model language Java, so that we do not have to care about serialization." what does using Java have to do with making serialization? Could you clarify? - "Since we wont our users" should be "want" - I really like your Data View, it's really easy to read - In your segment about Jenkins could you be more percise about what Jenkins actually doe sin your project, which tasks it performs and which tests are run by it? Hope everything I wrote is clear, else fell free to ask! Jannik@DigiWill

Felix Hausberger (2018-12-22 11:32:33)

Dear Jannik, thanks a lot for your feedback, we took your advices into account. The panel for which the API Key is exposed to in the frontend, is a custom user panel, therefore only the user who created the panel for his own dashboard will be able to see it. Moreover, this is just for demo purposes. The UC "custom panel" will be developed next year, therefore the user should have the possibility to add an api key by using an extra menu, have a look at our "Add _API" UC. Then the API key will never be exposed anymore. I also forwarded all listed points about architecture to our chief architecture developer. Your sincerely, Felix (Team dashup) By the way: It would be great of you could give us the feedback next time BEFORE we hand in our homework :) Thanks!

DB Scheme And Class Diagram – dashup (2018-12-22 14:03:55)

[...] see a current version of our class diagram, have a look at our new blog post MVC Architecture. [...]

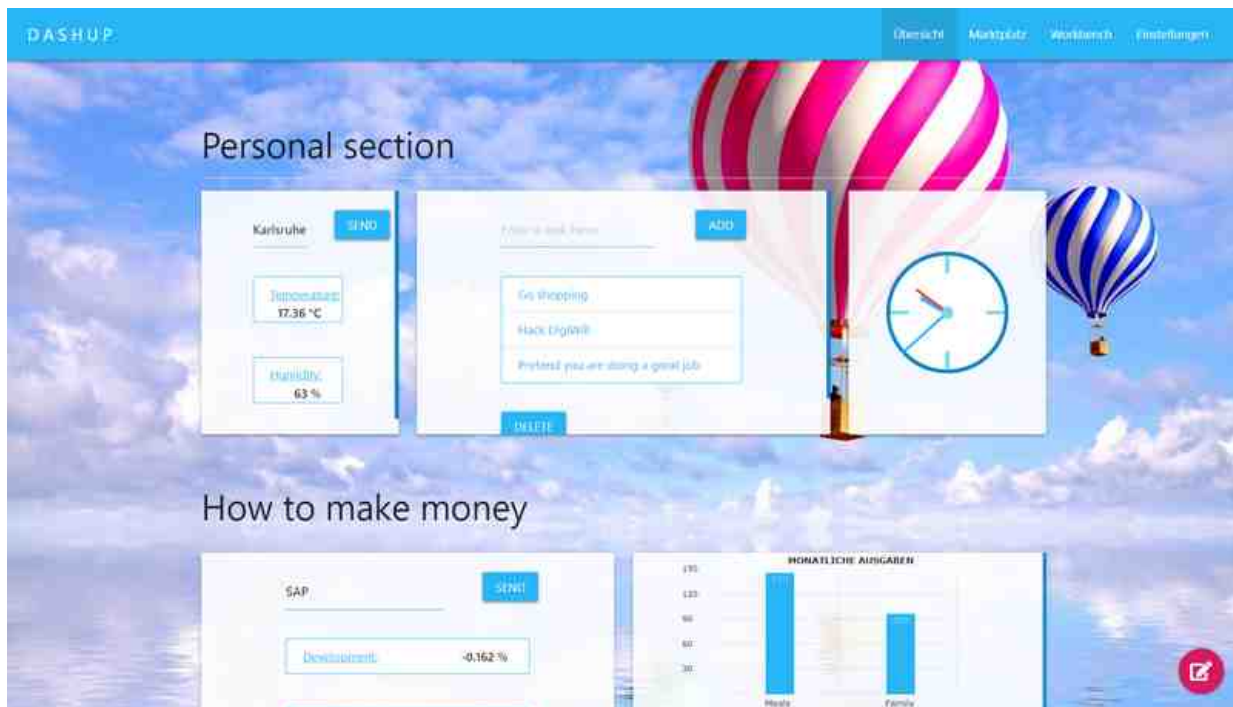
Midterm Summary – dashup (2018-12-22 14:50:56)

[...] MVC Architecture [...]

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:29:23)

[...] MVC Architecture [...]

Midterm Summary (2018-12-22 14:50)



Wow, it's unbelievable, that half of the time for our software project dashup should already be over. But the things we achieved, the individual phases we went through and the knowledge we gained was worth the hundreds of hours we spent. And with hundreds, we mean [1]HUNDREDS of hours. Not all of them are always tracked, as our philosophy puts passion over everything else. Therefore we even spend hours if we wouldn't need to, for us, for dashup and for all our fans out there... To make our lives easier :)

Here is a list of all goals we reached this term:

1. [2]Project vision
2. [3]Distribution of roles, Technology Stack, RUP
3. [4]Software Requirements Specification
4. [5]Use Case Specifications

5. [6]Gherkin Feature Files And Selenium Testing
6. [7]DB Scheme And Class Diagram
7. [8]Scrumming With YouTrack And Project Management
8. [9]Scrum Retrospective
9. [10]MVC Architecture

If you want to, you can download our midterm presentation [11]here.

Don't forget to visit us on [12]GitHub, [13]YouTrack or [14]YouTube!

But don't think that we will stop developing for our project. Of course, the contributions and commits for dashup will decrease in the next 3 months, but we are sure, that every now and then somebody is longing to get back to the spirit of dashup. See us next term in April 2019!

Your dashup team :)

1. <https://youtrack.dashup.de/reports/time/116-30>
2. <https://dashup2k18.wordpress.com/2018/10/10/our-project-vision/>
3. <https://dashup2k18.wordpress.com/2018/10/14/distribution-of-roles-technology-stack-and-more/>
4. <https://dashup2k18.wordpress.com/2018/10/21/software-requirements-specification/>
5. <https://dashup2k18.wordpress.com/2018/10/28/use-case-specifications/>
6. <https://dashup2k18.wordpress.com/2018/11/04/gherkin-feature-files/>
7. <https://dashup2k18.wordpress.com/2018/11/11/db-scheme-and-class-diagram/>
8. <https://dashup2k18.wordpress.com/2018/11/18/scrumming-with-youtrack/>
9. <https://dashup2k18.wordpress.com/2018/11/23/scrum-retrospective/>
10. <https://dashup2k18.wordpress.com/2018/12/02/mvc-architecture/>
11. <https://github.com/raphaelmue/dashup/blob/master/docs/presentations/midterm/Midterm%20Presentation%20dashup.pptx>
12. <https://github.com/raphaelmue/dashup>
13. <https://youtrack.dashup.de/issues>
14. https://www.youtube.com/channel/UCkzyPZ1hoasZHxEwxLfDu2w?view_as=subscriber

2. 2019

2.1 April

Confirmation Of Scope (2019-04-02 14:34)

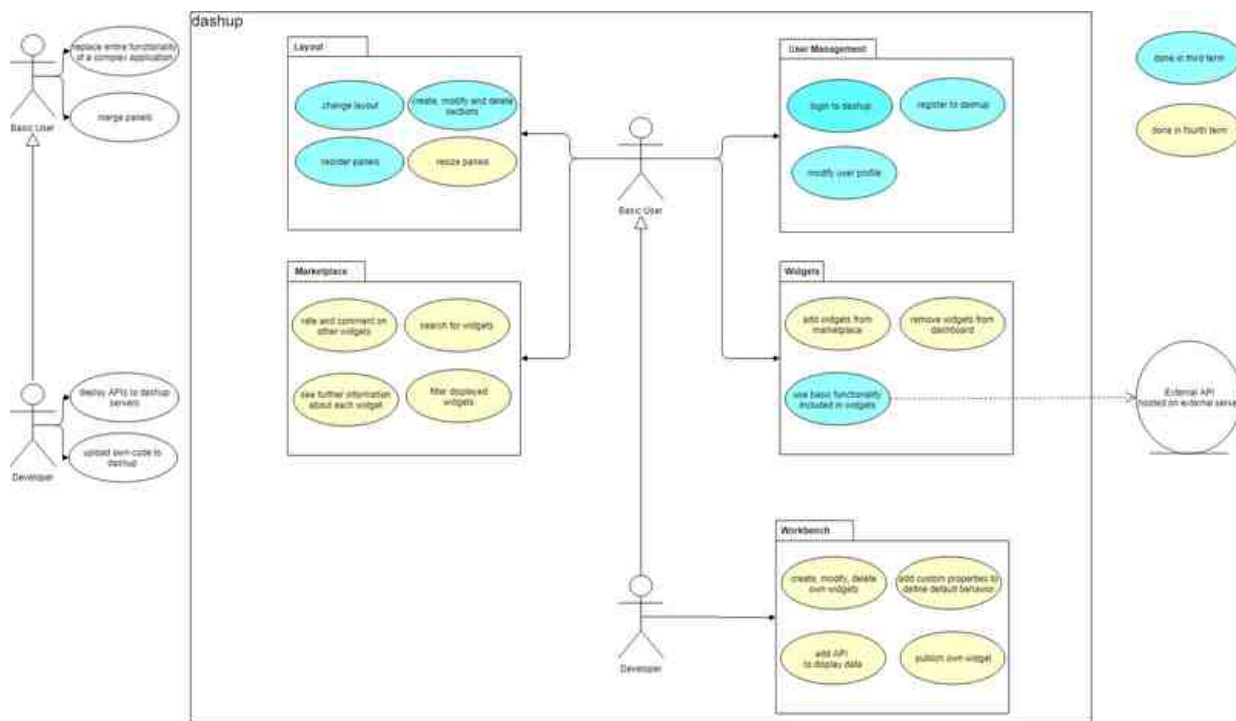


Hi all,

here we are again! This is the official start of the second half of our nice and innovative project "dashup". We will develop a bunch of new features this semester and we want dashup to be a finished and working productivity tool to make your life easier.

First of all, we want to confirm our vision and our plans for dashup. We are still convinced that this tool and vision is worth working on. Luckily we can also announce that there are no changes to our team so you can expect to see all of us contributing to the project regularly.

When it comes to the scope we needed to make some changes. Our overview of the project is a lot better than it was at the beginning of last semester. You can see our new use case diagram [1]here. We also updated the original blog post accordingly.



We needed to exchange some use cases between the semesters because we needed some features more than the ones we originally planned. Some wording changes were made, so a "panel" is now called "widget" whereas a "panel" is actually the container for a "widget" to avoid confusion. And last but not least we had to remove some use cases like "Developer can view statistics about his panel", because from the current perspective they would cause too much effort and jeopardize the success of the project.

Now that we have the new scope, we defined all use cases and we specified them in the known pattern. You can find the updated specifications [2]here.

Furthermore, to achieve the challenge to enable users to create own widgets, we added a new technology that helps us to develop reusable web components for our users. [3]LitElement, a simple base class for creating lightweight and efficient web components from Googles [4]Polymer Project, will give us a boost in developing web components.

We hope we were able to provide you a quick overview of what is planned for the next twelve weeks. We are looking forward to working on this fantastic project and keep in touch with you guys!

And as always: If there are any questions left, feel free to contact us.

See you!

1. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/srs/UCD.png>

2. <https://github.com/raphaelmue/dashup/tree/master/docs/specifications/ucs>

3. <https://lit-element.polymer-project.org/>

4. <https://www.polymer-project.org/>

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:13)
[...] Confirmation Of Scope [...]

Risk Management (2019-04-07 13:39)



Hey there!

Free climbers are living on the edge and are always looking for the ultimate kick. Dashup is a pioneer in its field and can in some aspects be compared to free climbers: The risks.

There are a lot of risks that we need to keep track of, some of them self-inflicted others arise through external dependencies. In this week we took some time to think about these possible risks. See the result below or through this [1]link.

Risk name	Risk description	Risk probability of occurrence	Risk impact / timing	Risk mitigation	Process in place	Risk score
Side tasks	Every task that comes up besides project tasks representing additional out-of-scope effort	100%	8	Explicit time scheduling	AI	8
Bad time estimation	Tasks take longer than expected leading to delay of milestones	50%	8	Reestimate time plan regularly, reassign tasks if needed	False	8
Documentation not detailed enough	Usage of web components is unclear to users with debugging cycle	70%	5	Independent library	True	3.5
Hardware problems	Laptop crashes multiple times while working, impacting workflow severely	25%	10	Contacting IT support as soon as possible	Unknown	2.5
Implementation bugs, code issues	Not working code leads to unhappy users	20%	8	Automated tests, code reviews through pull requests	AI	1.8
Security issues	When uploading an HTML file, all security risks have to be considered	10%	10	White-hacker, Codeay	Highend	1
Complex web specifications	Due to the extent of specification keeping the big picture in mind proved to be difficult	25%	3	Feature list, regular meetings ensuring the project progress	Unknown	0.75
Framework limitations	Extent of implementing features is limited due to framework limitations	5%	10	Usage of other framework, fallback to native implementation	Highend	0.5
Browser limitations	Whether browser supports or changes in specifications of web platforms help	10%	4	Regular checks for new announcements	False	0.4
Incompatible technologies	Collisions of different frameworks	5%	6	Read through documentation, do not use too many frameworks	AI	0.3
Maintenance of service providers	Availability of services depends on external providers	3%	10	Backups	True	0.3

This list will be updated on a regular basis.

To prevent some of the risks mentioned above we create time sheets listing the time spent for each use case grouped by work item types. Have a look:

PROJECTS	GROUP BY USE CASE	WORK TYPE	TIME ESTIMATED	TIME SPENT
Layout			67h47m	127h37m
DASHUP	dashup	Documentation		19h22m
		Development		108h15m
Marketplace			126h	121h
DASHUP	dashup	Documentation		18h
		Development		102h
		Testing		1h
No use case			287h50m	292h20m
DASHUP	dashup	Meeting		17h30m
		Documentation		104h22m
		Infrastructure		76h28m
		Development		50h
		Testing		44h
User Management			49h45m	56h15m
DASHUP	dashup	Documentation		17h
		Development		36h15m
		Testing		3h
Widgets			139h	168h40m
DASHUP	dashup	Meeting		9h
		Documentation		21h30m
		Development		138h10m
Workbench			69h30m	72h30m
DASHUP	dashup	Documentation		13h30m
		Development		56h
		Testing		3h
			Total time spent:	838h22m
				out of 739h52m

Hope you like today's blog post. As always we are open for any kind of notes or advice. See you next week!

1. https://github.com/raphaelmue/dashup/blob/master/docs/project_management/risk_management/risk_management.png

PerfectTimeCrew (2019-04-15 10:45:02)

Hey guys, Your list with the risks is well structured. The time sheet gives a nice overview of your efforts. Well done! The only thing you forgot is to put the warmup time into your time sheet. Keep up the good work. We are looking forward to your next steps. Your PerfectTimeCrew

Felix Hausberger (2019-04-15 12:47:37)

We know that warm-up time is missing because at the beginning of the 3rd semester we were not aware that we as well need to track warm-up time. We have now added it as a custom time classification field in YouTrack. We will not backtrack our warm-up time because of two reasons. 1. As far as I know, it is only used to separate invalid time spent from the function point analysis. 2. The only tasks where warmup time was needed were not related to a specific use case like our Jenkins CI/CD Pipelines. (See the Infrastructure - No Use Case). Spring Boot was very simple to understand and use, whereas Frontend Technologies like Polymer were also known among the frontend developers. Your dashup core development team

learnityourselfdhbw (2019-04-15 19:19:04)

Hello Dashup Team, I really like your risk management tables, they offer detailed information on every risk I could think of. However the table is really hard to see, maybe you could narrow a few columns so it becomes readable? (see our blogpost as an example) Sincerely, Mert @learnityourselfdhbw

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:15)
[...] Risk Management [...]

Time Estimation With Function Points (2019-04-22 18:43)



Hey there,

this week was a lot of work. On one hand, our backend developers coded endless hours into the night, whereas on the other hand time estimation efforts were made in order to track our progress and workload in the future. In our software engineering course, we have had an introduction into function points, a better way of estimating efforts.

A **function point** is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points are used to compute a functional size measurement (FSM) of software. The cost (in dollars or hours) of a single unit is calculated from past projects.^{[1][1]} The complexity factor in the domain characteristic table is measured by the amount of record element types, data element types and file type references according to this [2]table:

Tables	ILF and EIF Complexity Matrix		
RETs	1-19 DETs	20-50 DETs	51+ DETs
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

Visible groupings	EI Complexity Matrix		
FTRs	1-4 DETs	5-15 DETs	16+ DETs
0-1	Low	Low	Avg
2	Low	Avg	High
3+	Avg	High	High

EO and EQ* Complexity Matrix			
FTRs	1-5 DETs	6-19 DETs	20+ DETs
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Have a look at the results in our use case specifications: Layout

- [3]Change Layout
- [4]Change Panel Structure

User Management

- [5]Change Profile

- [6]Login / Register

Marketplace

- [7]Marketplace

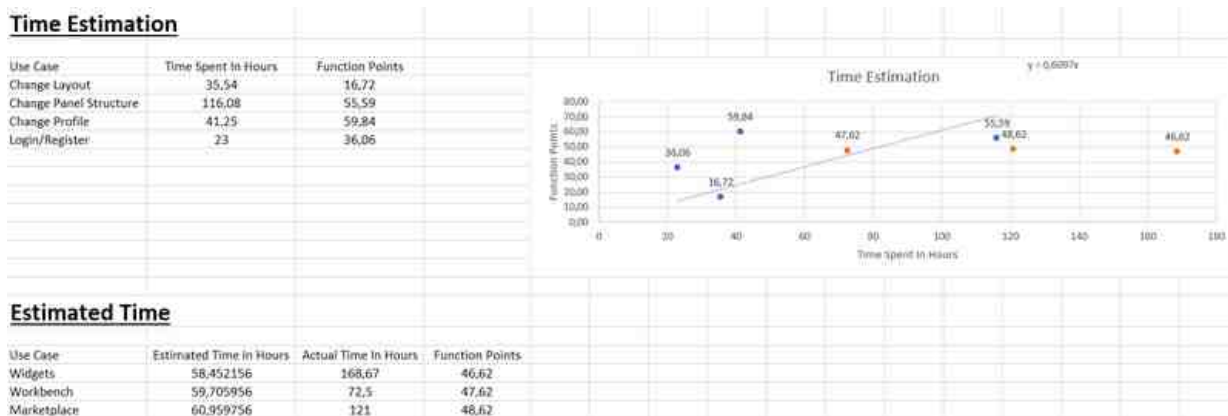
Workbench

- [8]Workbench

Widgets

- [9]Widget

If we plot the time spent on previous use cases with the calculated function points, we receive the following [10]results:



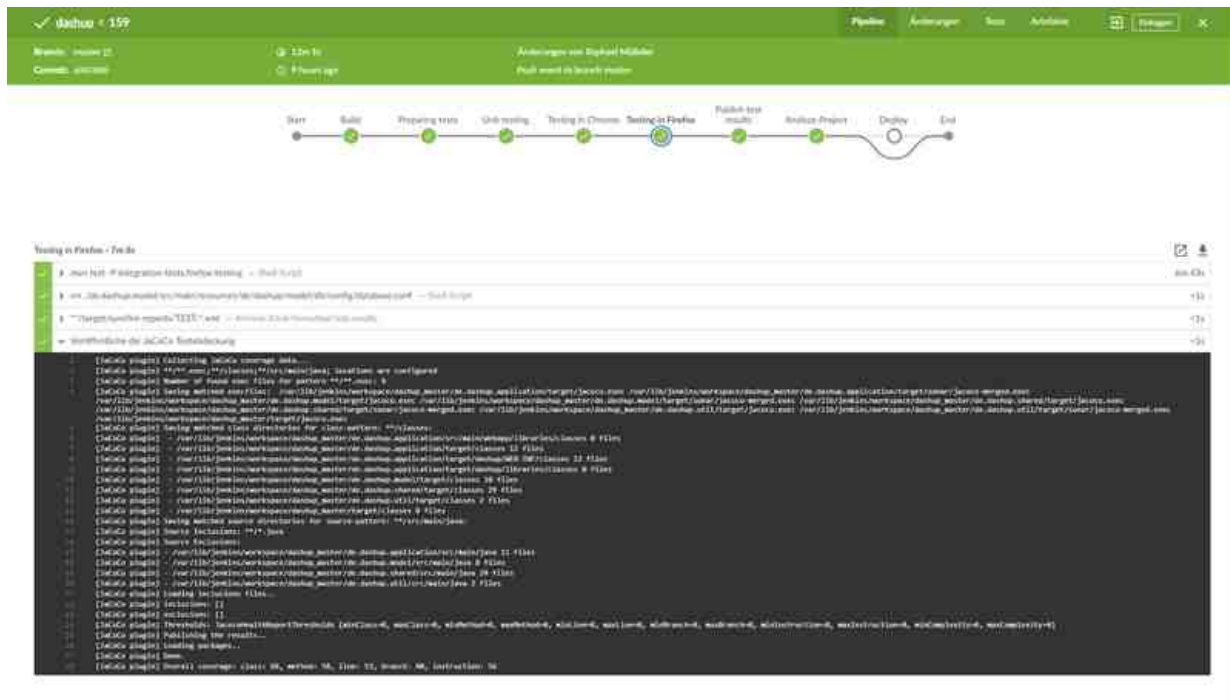
The data set colored blue belongs to the use cases already done in the last semester. As you can see there are two major outliers. The first one belongs to the login and registration use case, which only needs implementing basic form interaction and persisting data. There is not much calculation going on in the backend, but a lot of user functionality, that is later used in the user management and marketplace use case, is created. Same goes for the change profile use case within the user management use case. This is why we can handle a lot of features without much time effort. Generally, we were able to estimate the time we will need for future use cases to about 118.19 hours. As we managed to achieve our scope set for the 3rd semester in about [11]191.48 hours and effort is distributed equally according to the function points, we are confident to finish our scope in time. Even if the statistics are true and time estimations should always be doubled, 236.38 hours still would not bring us into big difficulties. Anyways, we will not underestimate this [12]risk! Stay tuned and see you next week! UPDATE Now, at the end of the 4th semester, we have updated the time estimation with the actual time spent for the use cases, that were left to implement. It can be seen by the data set colored orange. Once again we recognized to major outliers, but this time into the other direction. The first one, belonging to the use case Marketplace, can be explained due to the complex search and filter algorithms as well as the selenium tests, that were more extensive than in other parts of the dashup application. This is why this use case took more time as expected to implement. The probably biggest outlier of the whole chart belongs to the widget use case. There are several reasons, why this use case took nearly three times that long as expected. As the dashup widgets are built out of dashup web components, that are later on used to built own custom widgets in the workbench, a concept of how different components interact and communicate in still a generic way with each other had to be developed. There were multiple approaches to this issue, starting already during the practical phase. With Polymer, we later on finally found a solution. Then, developing the components, making them responsive to theming and sizing, specifying a generic interface and data formats, documenting how to use the components, writing own

dashup APIs for the default dashup widgets, reviewing, improving metrics, and so on... As you may see, there were as well a lot of efforts besides pure development, that go along with the widget use case. And working out a generic "architecture" for such components was one of the biggest challenges we had, which is how to actual time spent could be explained. So in the end, it took us far more than only 118.19 hours. But the result of [13]630.88 hours of hard work, dashup is not only a huge but as well a very competitive and high-quality application that all of our developers are very proud of.

1. https://en.wikipedia.org/wiki/Function_point#cite_note-1
2. <http://www.softwaremetrics.com/fpafund.htm>
3. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_layout/UCS_change_layout.md
4. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_panel_structure/UCS_change_panel_structure.md
5. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/change_profile/UCS_change_profile.md
6. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/login_register/UCS_login_register.md
7. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/marketplace/UCS_marketplace.md
8. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/workbench/UCS_workbench.md
9. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/widgets/widget/UCS_widget.md
10. https://github.com/raphaelmue/dashup/blob/master/docs/project_management/time_estimation/time_estimation.xlsx?raw=true
11. <https://youtrack.dashup.de/reports/time/116-13>
12. https://github.com/raphaelmue/dashup/blob/master/docs/project_management/risk_management/risk_management.xlsx?raw=true
13. <https://youtrack.dashup.de/reports/time/116-19>

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:17)
[...] Time Estimation With Function Points [...]

Testing (2019-04-26 12:01)



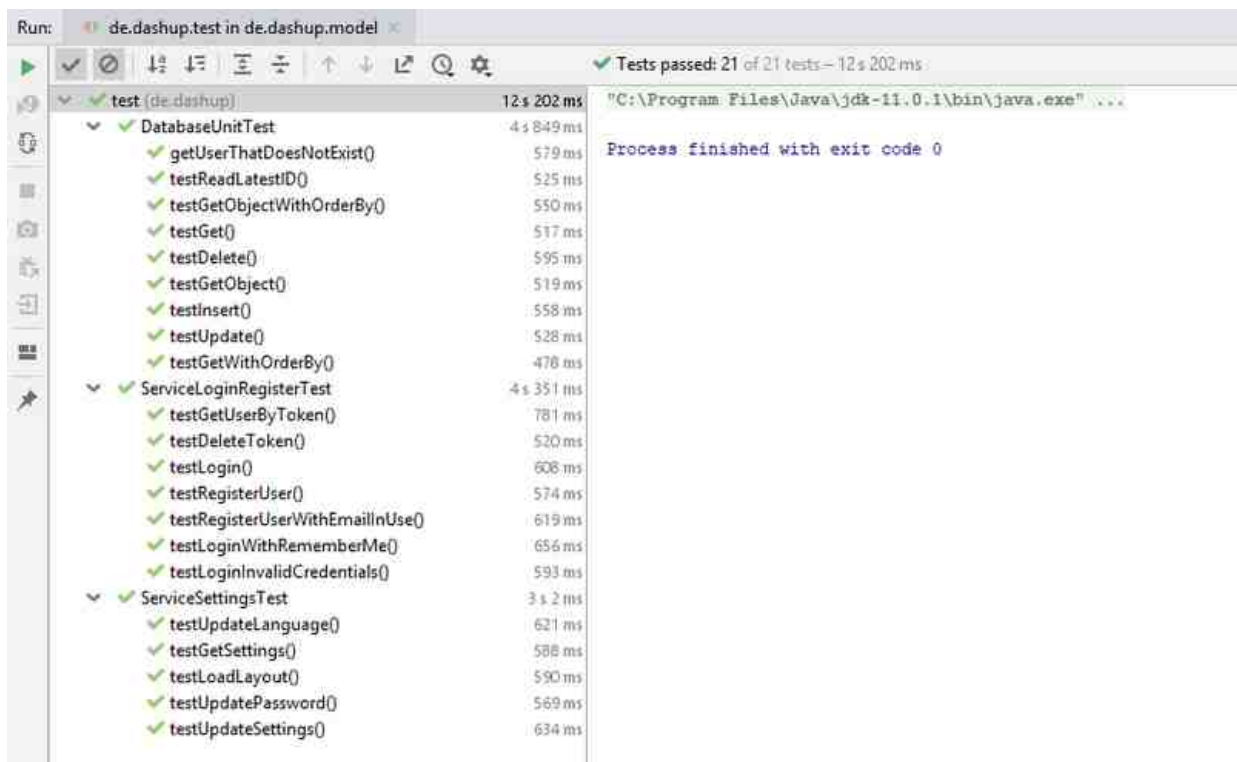
Hi guys!

This week was all about testing. We created a test plan for our application, which you can find right [\[1\]](#)here.

This document contains all the information about our handling of tests and the scope regarding tests.

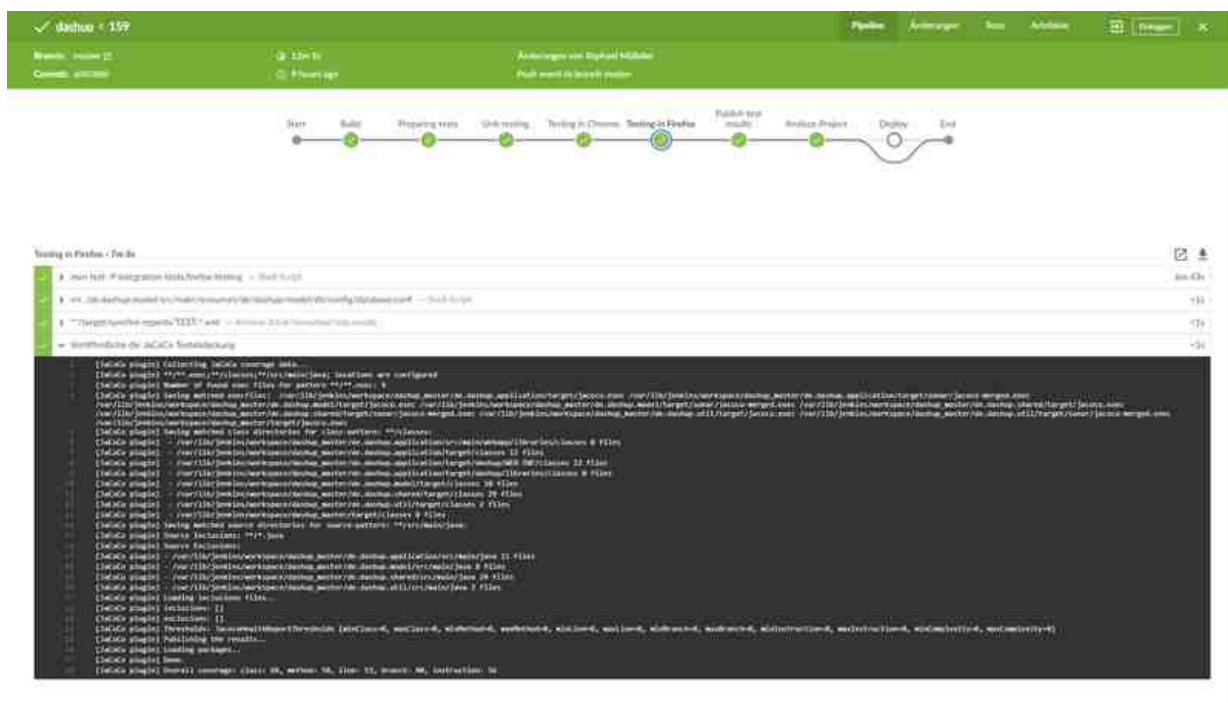
Testing

So far, we have implemented [\[2\]](#)JUnit Tests as well as [\[3\]](#)Selenium Tests as far as we could. For our JUnit tests, we are using the [\[4\]](#)JUnit Jupiter engine. Have a look:



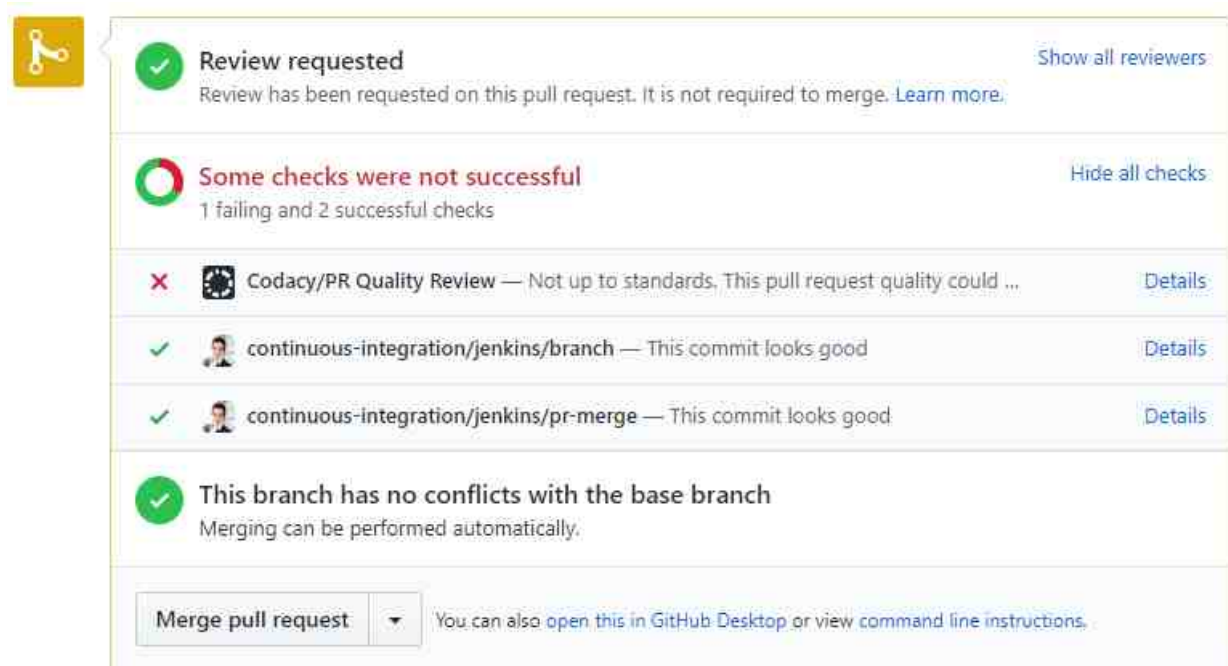
More tests will be merged into the master branch soon. Find all test logs [5]here or on our [6]Jenkins server. We mirror the productive database into the test database on a regular basis to guarantee testing on a real-world level. Testing tries to be compliant with the activity diagrams as good as they can, have a look at [7]this example (find the feature file linked in the document).

To keep track of our tests we set up a continuous integration service. We decided that Jenkins would fit our project's needs. You can check out multi-branch pipeline [8]here. We configured our build steps and stages in the [9]Jenkinsfile on our GitHub repository.



Furthermore, we use [10]Codacy to track the quality of our code. We, therefore, added a badge to our [11]README, so that you can easily check out the status of our code quality.

We also managed to integrate Jenkins and Codacy into GitHub's pull requests. Each time, a commit is pushed to the pull request, all checks will be executed. This helps us to identify errors and bugs directly after pushing.



Code Coverage

By means of the Maven plugin Jacoco, we are able to gather code coverage data. After our Jenkins server has executed all tests successfully, this data will be published to [12]Codacy. But as we had some trouble integrating Codacy to receive and interpret the test data correctly, we, later on, switched to [13]SonarCloud, which perfectly analyzes test coverage with test data provided by Jacoco and is integrated into our CI pipeline. Check out its badge in our GitHub repository. In our Test Plan Document, we defined to have a code instruction coverage of 50 %, which we archived by now.

Feel free to give us feedback on our Test Plan Document as well as on our testing landscape.

1. https://github.com/raphaelmue/dashup/blob/master/docs/architectures/testing/test_plan.md
2. <https://github.com/raphaelmue/dashup/tree/master/de.dashup.model/src/test/java/de.dashup/test>
3. <https://github.com/raphaelmue/dashup/tree/master/de.dashup.application/src/test/java/de.dashup.test/steps>
4. <https://github.com/raphaelmue/dashup/blob/master/de.dashup.model/pom.xml>
5. https://github.com/raphaelmue/dashup/tree/master/docs/logs/surfire_test_logs
6. <https://jenkins.raphael-muesseler.de/blue/organizations/jenkins/dashup/activity>
7. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/change_profile/UC_S_change_profile.md
8. <https://jenkins.raphael-muesseler.de/blue/organizations/jenkins/dashup/activity>
9. <https://github.com/raphaelmue/dashup/blob/master/Jenkinsfile>
10. <https://app.codacy.com/project/dashup/dashup/dashboard>
11. <https://github.com/raphaelmue/dashup/blob/master/README.md>
12. <https://app.codacy.com/project/dashup/dashup/dashboard>
13. <https://sonarcloud.io/dashboard?id=dashup>

CommonPlayground (2019-04-28 20:53:16)

Dear Team dashup, I like your test plan document and your already working CI/CD pipeline based on Jenkins. Feedback regarding the GC: Can you please provide links to your test code and the test library integration (maven file). Furthermore it would be nice to have a screenshots for test run in your IDE directly visible in your blogpost (the screenshot already exists in your test plan document:) Keep up the good work! Kind regards Nils@CommonPlayground

Felix Hausberger (2019-04-29 22:54:44)

Thanks a lot for your comment. You were absolutely right, we added the missing information.

MNZ (2019-04-29 21:39:08)

Dear Team dashup, Your test plan document and pipelines looks very nice. On your codacy page if I hover over code Coverage there are 80 uncovered Files is this because I don't see the real code Coverage or because you didn't configure it yet. Your MNZ-Team

Raphael Müßeler (2019-04-30 07:45:20)

Hi MNZ-Team, we are working on this issue. Unfortunately, our script for publishing our coverage to Codacy is not working correctly. But our actual code coverage can be seen on our Jenkins server on each branch (like <https://jenkins.raphael-muesseler.de/job/dashup/job/master/>).

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:18)
[...] Testing [...]

2.2 May

Refactoring (2019-05-05 15:08)

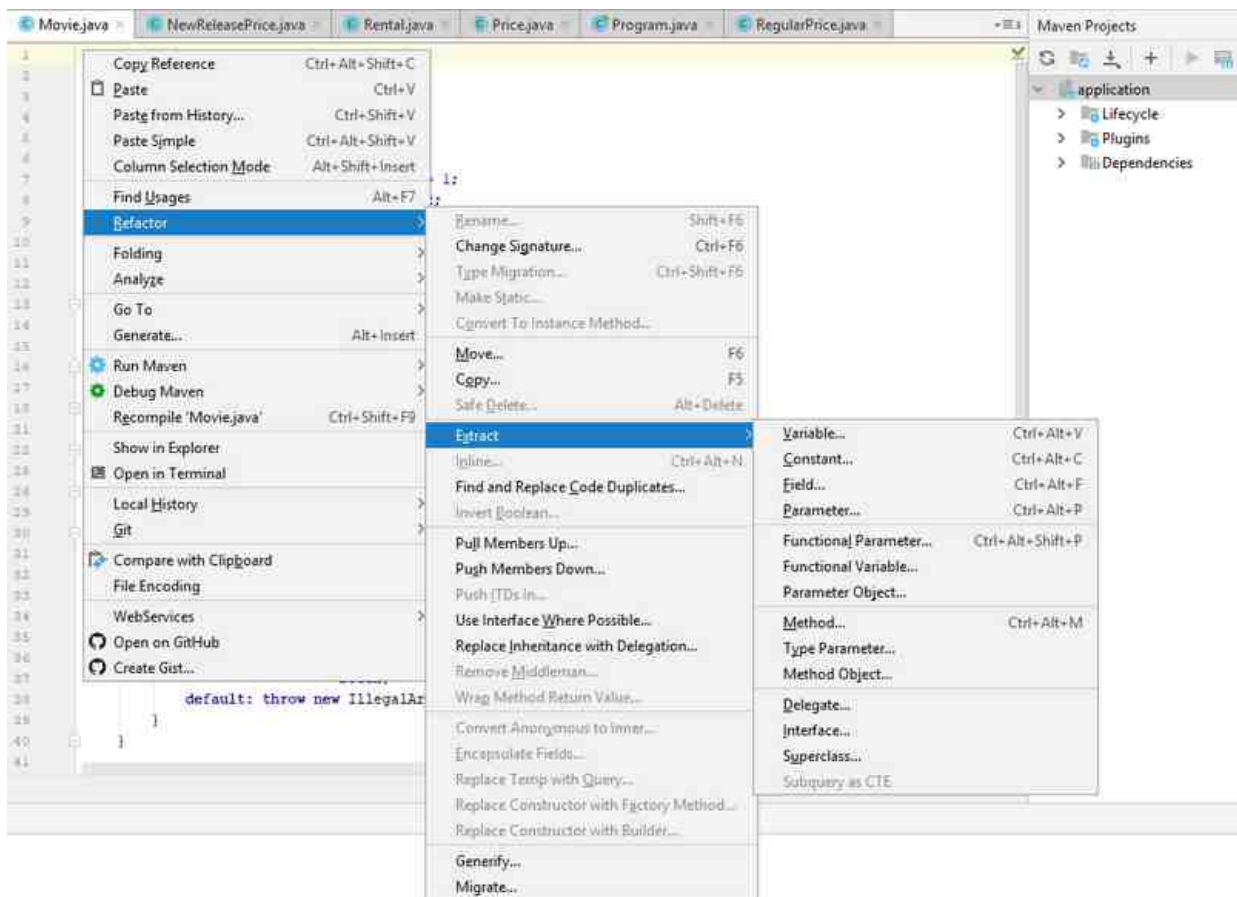


Hey there,

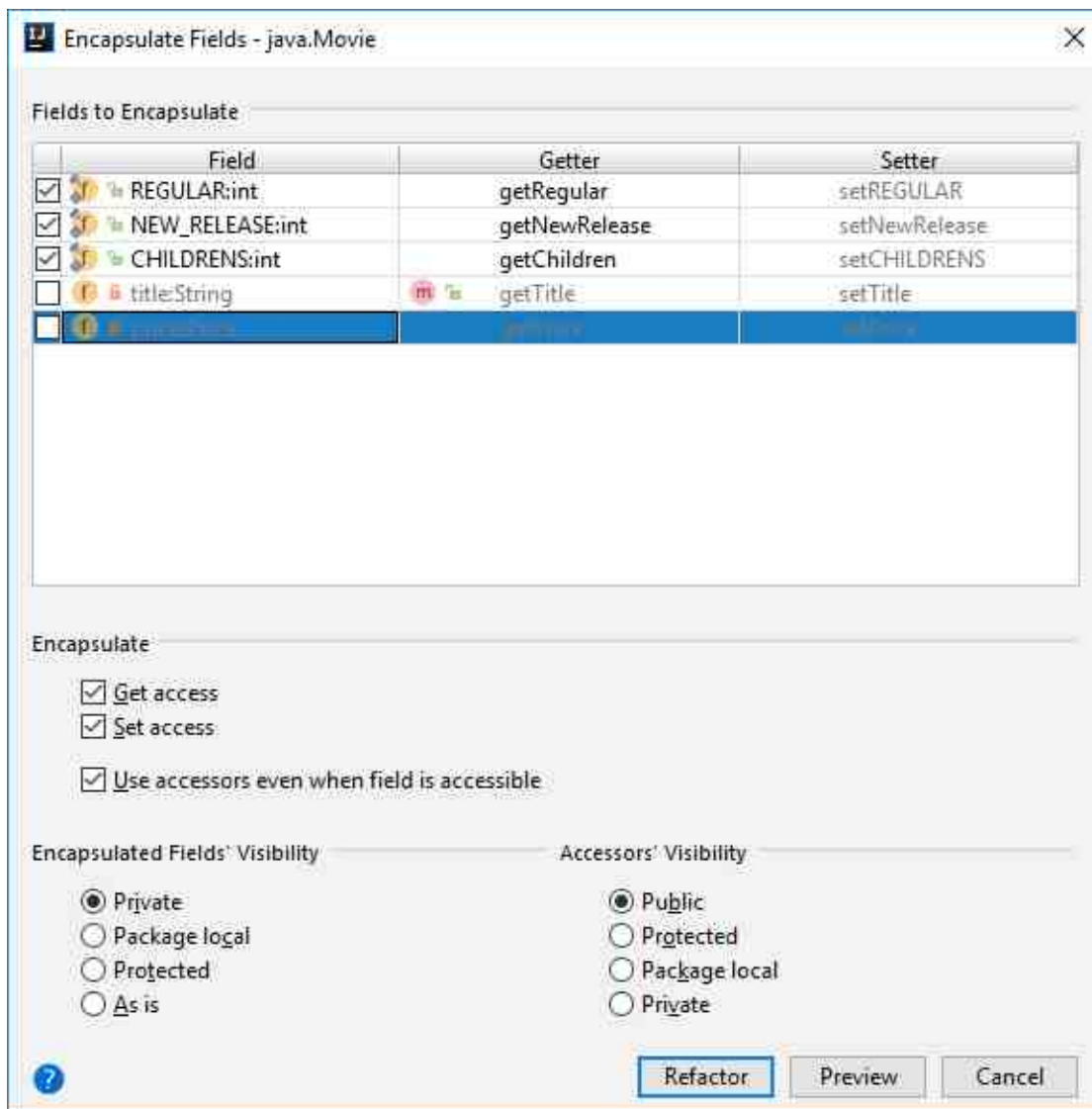
this week we had the opportunity to read through chapter one of Martin Fowler's famous [1]Refactoring: Improving the Design of Existing Code!

Each of us had a more in-depth look at what refactoring actually means and how to use the tools provided by IntelliJ to make refactoring as automated as possible.

IntelliJ offers a wide range of refactoring tools:



For instance, extracting logical units to independent methods, moving these methods to other classes, encapsulating fields using self-defined accessors, renaming entities and much more can be done with simple context menus. Heres an example of encapsulating fields:



You can describe how constants or variables should get accessed by defining own getters and setters. By ticking the "Use accessors even when field is accessible" checkbox, you can make sure that attributes are only accessed using the defined accessor methods and not directly, which is the thought behind encapsulating fields.

The first chapter of Refactoring dealt with an overview of different topics regarding refactoring, such as decomposing, redistributing, encapsulating, polymorphism and the famous state pattern from the Gang of Four.

See ur result here: [2]Felix [2], [3]Joshua, [4]Sven, [5]Raphael.

Each commit should have the name of the code smell that was removed in its commit message. We have even used [6]Codacy to track our progress in code quality after each refactoring step. JUnit tests grant, that the core functionality remains the same after each refactoring step.

Hope you liked today's blog post. See you next week!

2. <https://github.com/fidsusj/Fowler-Refactoring>
3. <https://github.com/joshuaschu/Fowler>
4. https://github.com/svenleonhard/Fowler_Refactor
5. <https://github.com/raphaelmue/fowler>
6. <https://app.codacy.com/>

PerfectTimeCrew (2019-05-06 18:35:01)

Hi guys, we really like your work this week. In your repositories, it is easy to see the different steps of refactoring and your commit messages are understandable. Joshua and Raphael, we could not find any tests for your code. Which tool did you use for testing your code before and after refactoring? Keep up the good work. We are looking forward to your next steps. Your PerfectTimeCrew

Felix Hausberger (2019-05-13 16:44:04)

Dear PerfectTime, thanks for the feedback, both of them have added test code (JUnit tests) in their repositories. Your dashup team

flashcardcommunity (2019-05-07 07:43:13)

Hello there, I really like your blog post. You explained exactly what tools you used and why in particular you used them. Furthermore I like that you mentioned what IntelliJ is capable of in terms of refactoring! Sadly, I couldn't find any JUnit Tests at Joshuas and Raphaels repository and why exactly did Felix add the Codacy badge four times? :) Other than that, great job! Greetings Moritz@FlashCardCommunity

Felix Hausberger (2019-05-07 07:47:01)

Because I thought it said somewhere the more batches the better the grade ;)

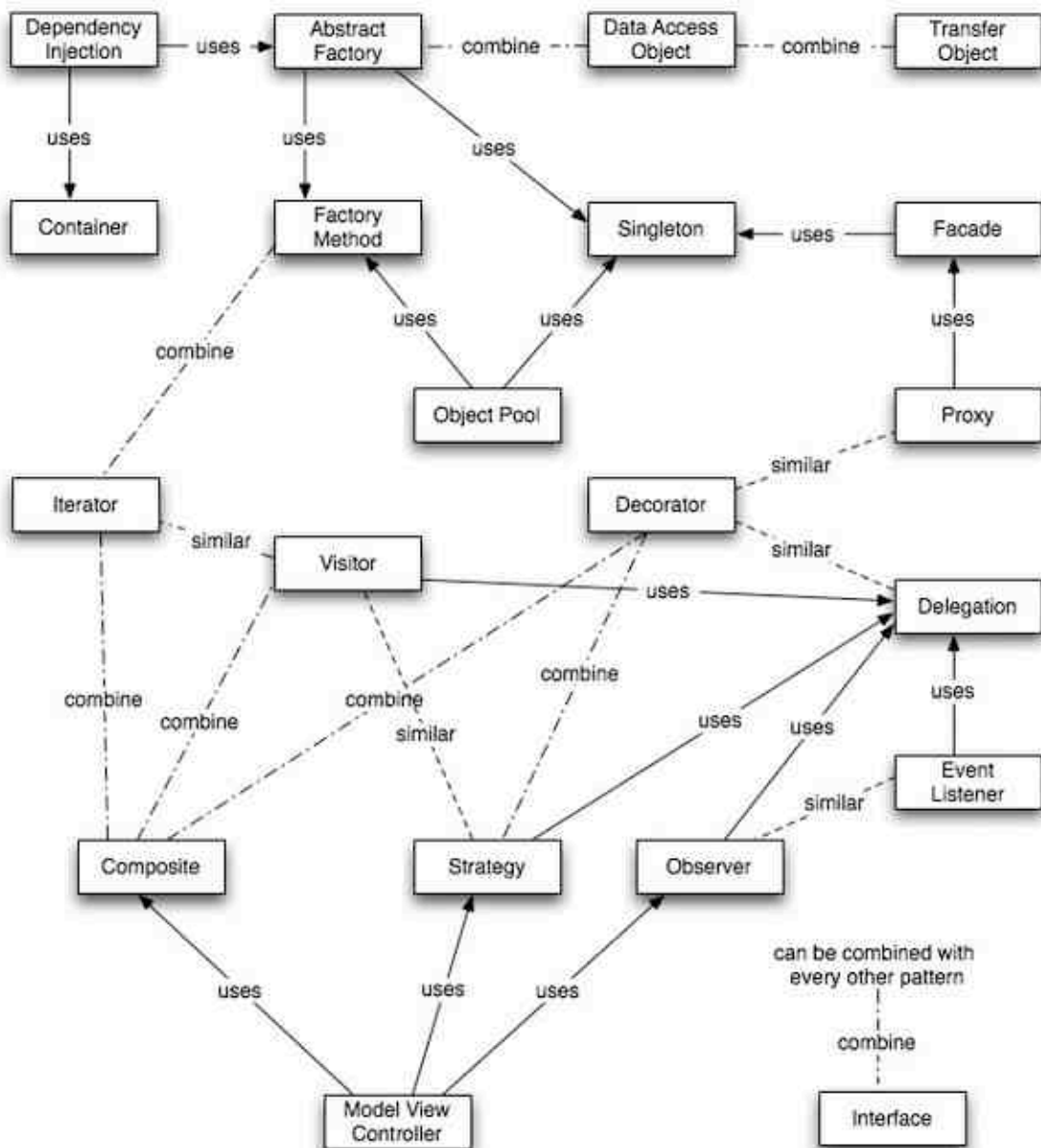
Felix Hausberger (2019-05-13 16:45:24)

Dear FlashCardCommunity, thanks for the feedback, both of them have added test code (JUnit tests) in their repositories. Your dashup team

MNZ-Team (2019-05-07 07:43:31)

Hi, I agree with the PerfectTimeCrew. Your repositories were nice to understand. Also Felix and Svens batches were really nice. your MNZ-Team

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:20)
[...] Refactoring [...]



As the Gang of Four has taught us, high-quality software has its roots in well-considered design patterns, that are applied in the software itself. When it comes to design patterns, we have to differentiate between multiple styles of patterns:

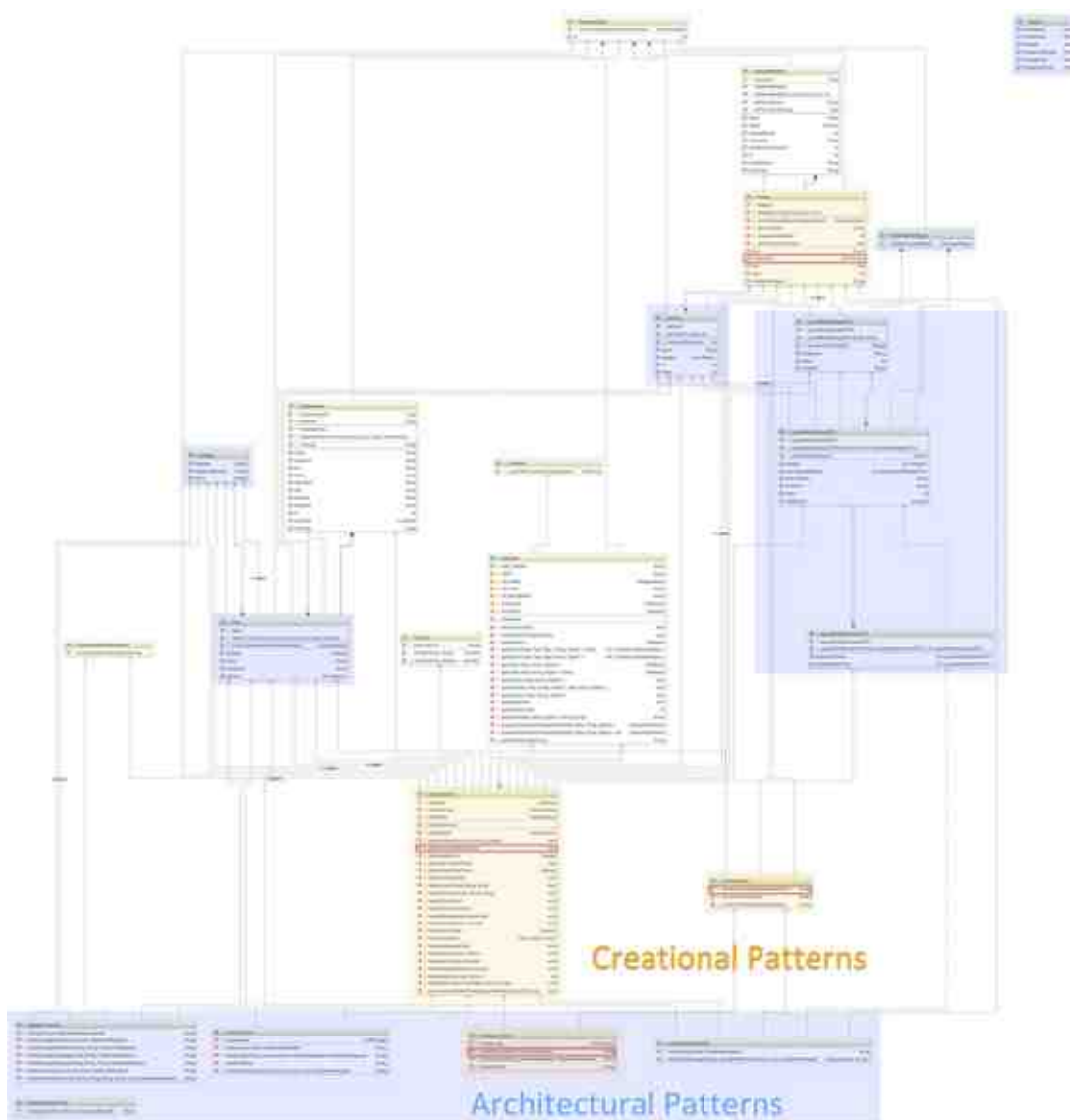
- Creational Patterns
- Structural Patterns
- Behavioral Patterns
- Functional Patterns

- Concurrency Patterns

- Architectural Patterns

As dashups codebase is growing more and more, it was needed to have a closer look at the different kinds of design patterns. Some of them were already implemented in dashup, especially architectural design patterns like MVC, Front Controller, Data Access Objects and Data Transfer Objects, but anyways we found another design pattern, that was really worth implementing. With it came several other patterns, that were required to realize it.

Dashup provides a central dashboard with widgets for private productivity usage. Users can create custom widgets using dashups web components. Before having a look at dependency injections and lazy initialization, creational design patterns, a client requested every dashup web component available, as he could possibly need it for his widgets to be rendered on the dashboard. Of course, that meant, that a whole lot of web components are pushed to the client in vain, increasing average response time and decreasing performance. With these two patterns, only the web components needed at the moment are loaded lazily. New web components will only be loaded, when users add new widgets to their dashboard. Let's have a look at the [1]class diagram first:



Classes belonging to an architectural design pattern are highlighted blue, but they weren't the main reason for this blog post. To implement dependency injection to realize lazy initialization, we first had to make the dependencies to inject as modular as possible. Fortunately, web components are naturally designed modular, each web component is imported as a [2]JavaScript module into the browser session, so no need to add extra logic for the so-called module pattern. The [3]web platform APIs already offers a way to include JavaScript Modules instead of normal scripts into the browser session.

To now implement the dependency injection pattern, further changes were needed. The pattern regulates the dependencies of an object at runtime. If, for instance, an object needs another object during its initialization, this dependency is stored at a central location - it is therefore not generated by the initialized object itself. To load the required dependencies, new tables in the database were needed, that store the relationship between a widget and its required components. So when each widget is now loaded into a data access object, additional information on the required components are now stored in a list within this object. This is done by a service method that runs as soon as

a client requests the central dashboard and requests the widgets. Within a builder class, the dependencies located in the data access objects are then injected into a JSP View, which is sent back to the client. The client then knows exactly, which components to load and therefore reduces the payload. So in the end, widgets do no longer need to take care of loading and initializing their dependencies, as this is already done by service and builder methods in the backend and pushed to the client.

This thought hits the true point of lazy initialization or lazy loading, as dependencies get only injected and initialized the moment it is really needed and not from the very beginning. So when a client adds a new widget to his dashboard, then and only then the required new web components get loaded.

Now, this sounded very hard to implement, but actually, it was quite simple. Here some references to our code:

- [4]Before
- [5]After

Just have a look at the classes highlighted in orange in the above class diagram or the [6]pull request created for the design pattern changes. The class diagram before the changes belonging to the creational design patterns looked the same (as no structural or architectural changes like in other patterns were made). Best would be to start from the Front Controller called "DashupController" if you want to go through all necessary coding, as the pull request doesn't contain all of it (some logic was already implemented).

So what do you think? Do you agree with us, that these three/four creational patterns really helped us improving performance?

Your dashup team

1. <https://github.com/raphaelmue/dashup/blob/patterns/docs/architectures/patterns/patterns.png>
2. <https://github.com/raphaelmue/dashup/tree/master/de.dashup.application/src/main/webapp/components>
3. <https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Statements/import>
4. <https://github.com/raphaelmue/dashup/tree/a1fa554c18ffa72067ee076ccecf3850691f0ddb4>
5. <https://github.com/raphaelmue/dashup/tree/a296b6dac8a8469d9f5b5d93b5d4d3245044fe93>
6. <https://github.com/raphaelmue/dashup/pull/99>

Christian Schweigel (2019-05-14 07:32:19)

Hi dashup-Team, your work looking good so far. Your class diagram shows very good, which classes are parts of which design pattern. Also the decision for using the dependency injection pattern seems very practicable for your project. For the comparison of your code it would be very handy to create a pull request with your changes so we can see directly, which files where changed in which way instead of looking in two different states of your project. I think your patterns will help you to make your project run better and have cleaner code, so keep up with them. Best regards, Christian@DigiWill

Felix Hausberger (2019-05-14 07:47:40)

Hey Christian, thanks for mentioning, we added a pull request with the adjustments we did for further transparency! Nevertheless, not everything that is needed in the pattern is contained in the pull request, as some parts were already

implemented. This is why I marked the specific methods in the class diagram for you. :)

MNZ Team (2019-05-14 08:44:29)

Hi dashup-Team, it's really nice to see what you have done this week. Looks like the Design Pattern you chose fits best for you and will help you write cleaner code. One thing we would like to mention is that the picture of your class diagram is not readable and can also not be magnified. Besides that, great work this week. Best regards MNZ-Team

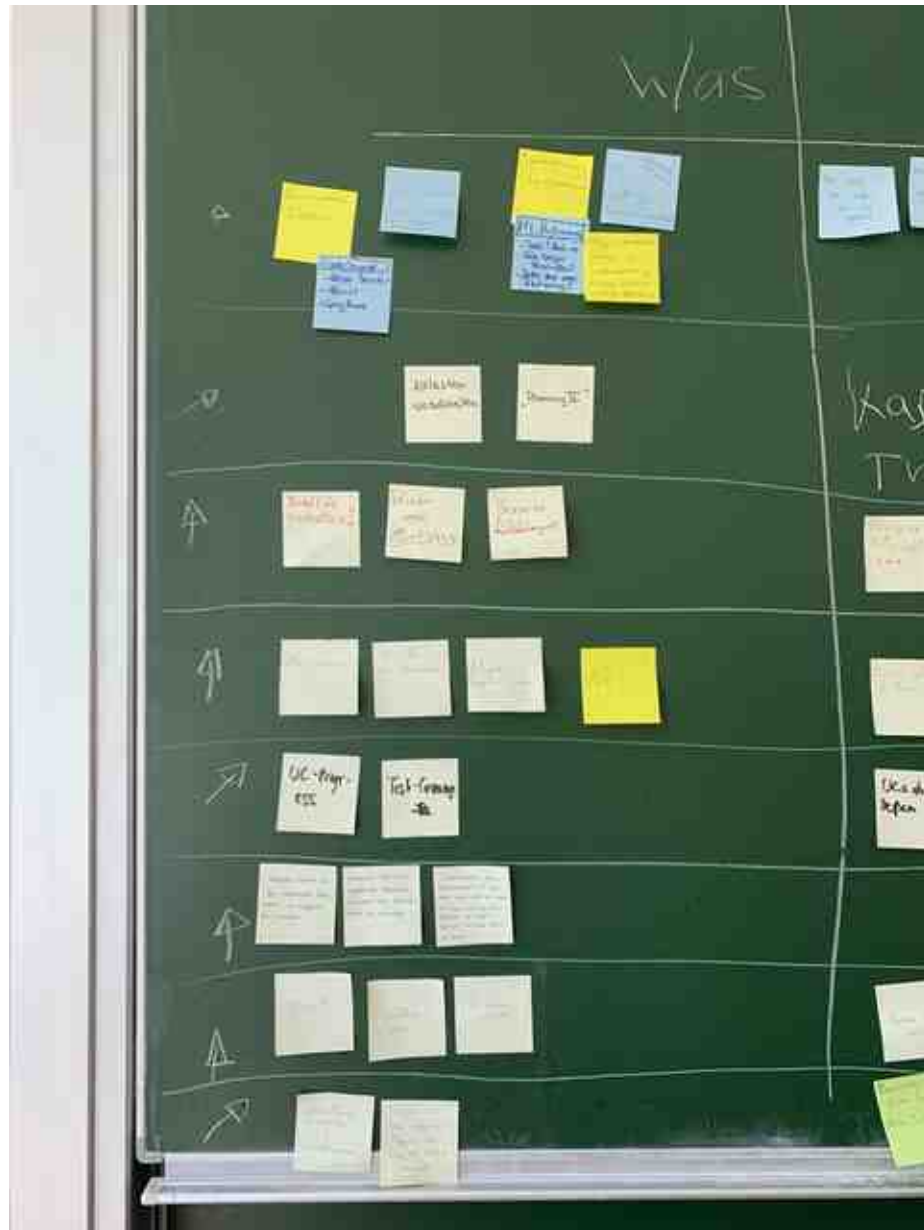
Felix Hausberger (2019-06-13 17:51:00)

That's why we added the link to the class diagram ;)

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:21)

[...] Design Patterns [...]

Scrum Retrospective II (2019-05-17 14:43)



Hey there,

like the last time we did the retrospective, we used today's week scrum retrospective to think about what went wrong, what went good and what to improve. As a reference point, we compared our current status with the results we worked out last time in the scrum retrospective and came to the following conclusion:

At the beginning of the project, we had some difficulties to all agree on technology and architectural designs of our application. Some wanted to bring in new technologies, whereas others rather insisted on conventional methods. We managed to get through this time, achieving a consensus of all team members in the end. It took us some time to find out the true advantages of the new technologies, which we then integrated into the project, whereas in backend related tasks, we stuck to our conventional designs. Informing about the advantages and

disadvantages first and later on evaluate in a reasonable discussion of what suits best, is probably the best strategy you could choose in such issues. That's one way to respect everybody's opinion, prove them right or wrong to avoid prejudices and achieve a general consensus.

Another aspect was empathy towards other peoples problems, which were almost ignored in the beginning. This was solved by using pull requests and issues in GitHub, making people work more together. Almost magical how much this supported team spirit and a better understanding of the whole application. This behavior furthermore fixed the last aspect of the last retrospective: the lack of support. And with the beginning of feeling more about each others well-being, helping others is not seen as an obligation anymore, but self-evident! So the more communication in the team, the better the support gets. In regular meetings, we communicated problems and possible risks and everybody received the support he needed in the end.

Normally you should differentiate between which goals/habits to start with, which to keep or which to drop. Fortunately, the above-mentioned, new habits should all remain the same until the end of the project and grant a successful end result.

Of course, we agreed on new aspects to work on:

- Set clear goals, when to have something done. Creating tasks in YouTrack and assigning them to people could lead to issues, that will never be done in the end. So the scope of our work due to the end of the week should in the future always be clear. This way, we make sure, that work on the project is distributed equally among the time we have.
- Because we now tracked many of our issues in GitHub, some of us neglected our actual tool for project management: YouTrack. To create proper reports, for instance for time estimation, it's necessary to always create corresponding issues in YouTrack as well. To do so, our agenda of the weekly meetings will now follow the issues tracked in YouTrack.
- Communication with our community has almost been done by one single person. Now the whole team should be involved in this process in order to bring the project members closer together with the community.

Furthermore, we took the time to think about, which areas of the project can be transferred to the industry and which not. And to be honest, most of the things we did in the project, will not be the same as in business. In this project, we are the only ones who can influence the design, features, and scope of the project, whereas, in the industry, customers will decide, what suits best for their personal needs, even if this is sometimes not the most reasonable and easiest to implement. Of course, this project as its difficulties and level, but it's more like you should compare a playground with a battlefield and find out the similarities. Do you see some? The only point in this project is to do as many mistakes as possible to not do them in business because that's where it costs a lot of time, resources and money.

So all in all, this retrospective helped us to once again work out our current progress and problems, gave us some time to talk to each other and to think about the progress of the project, but not to find similarities with the business out there! But still, it was quite a good get together on a more in-depth level. So in comparison to the last retrospective, we made huge progress, not only in our team spirit but as well in the progress of the project.

Let us know what you think! See you next week.

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:23)
[...] Scrum Retrospective II [...]

Test Coverage (2019-05-20 14:01)



Hi guys,

this week, we've been working on our test coverage. You can check out our current test coverage [1]here. We even managed to publish this coverage to Codacy as well, but we are still having some issues to properly automate this process. This is why in the future, test coverage in Codacy might not be up to date all the time, whereas on SonarCloud you should always get the latest metrics about test coverage.



A Project certification

Quality evolution

Last 7 days

Last 31 days



We also have implemented the following test types into our test workflow:

1. [2]Unit tests
2. [3]UI Tests
3. [4]Smoke Tests

These three testing practices help us to keep our application away from any new issues that arise when changing the code. Besides those are described in our [5]test plan document as well.

In our case, we only integrated unit and UI tests into our [6]Jenkins multi-branch pipeline, since smoke tests are meant to be executed locally so that we can easily verify whether the main functionality of our application is working or not before contributing.

We would enjoy receiving your comments about our testing infrastructure.

1. <https://sonarcloud.io/dashboard?id=dashup>
2. <https://github.com/raphaelmue/dashup/tree/master/de.dashup.model/src/test/java/de/dashup/test>
3. <https://github.com/raphaelmue/dashup/tree/master/de.dashup.application/src/test>
4. <https://github.com/raphaelmue/dashup/tree/master/de.dashup.model/src/test/java/de/dashup/test>
5. https://github.com/raphaelmue/dashup/blob/master/docs/architectures/testing/test_plan.md
6. <https://jenkins.raphael-muesseler.de/blue/organizations/jenkins/dashup/activity>

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:24)
[...] Test Coverage [...]

Metrics (2019-05-26 19:25)



Hey there,

this week we had a closer look at some metrics to improve code quality on our project. Metrics are used to numerically evaluate the code. It helps to find critical code sections, examine them more precisely and eliminate them in the best case.

For this purpose, we have integrated [1]sonarcloud.io into our CI process. This can be seen on the screenshot of our Jenkins Pipeline at the step **Analyze Project**.

[2]



Sonar's dashboard gives an overview of the current code situation. Serious errors and grievances can be detected quickly. The dashboard is public and can be accessed [3]here.

For more detailed metrics around the code, the IntelliJ Plugin Metrics Reloaded is used. Metrics of different categories can be automatically generated in a few seconds. In this way, you can get a lot of information about the code and act if necessary. The following is about the [4]Chidamber & Kemerer object-oriented metrics suite, which unites various metrics.

Two of these are Weighted Methods per Class (WMC) and Response for a Class (RFC). We will take a closer look at these two methods now. WMC counts the number of methods in each class. This value is relevant because you want to avoid too many methods. This makes it difficult to maintain and reuse the code. Somewhat more complex is RFC. The methods are also counted here. In addition to this value, the number of methods that are called within the methods of the class is also added. A high RFC value usually means that the class is more prone to errors. The complexity of the class is high. This can lead to difficulties in understanding and make both testing and debugging

more difficult.

If you look at the values for WMC and RFC, you will notice the *Database* and the *DashupService* class. There are clear anomalies here that need to be reduced. Due to the architecture of dashup, there are some difficulties. On the one hand, we want to provide the functionality for CRUD operations in one place, on the other hand, we want to establish a clear structure. In refactoring, we tried to combine both approaches. Some methods have been merged and outsourced. This has reduced the total number of methods and the number of method calls. The result is the [5]issue112 branch on GitHub. Here all changes can be traced by clicking the compare button on the top right corner. This [6]link leads to a commit before the refactor work. The table shows the before/after values.

Class	RFC before	RFC after	WMC before	WMC after
-------	------------	-----------	------------	-----------

DashupService	154	139	81	74
---------------	-----	-----	----	----

Database	56	53	36	32
----------	----	----	----	----

Another metric is Lack of Cohesion in Methods (LCOM). There are different versions from 1 to 4. According to documentation, the metric tool uses LCOM1. Basically, this metric measures the coherence of methods. This should be as high as possible. Therefore, it is important that the LCOM value is low. During the calculation, all possible method pairs are considered. If both methods access one or more equal attributes, a variable Q is incremented by one otherwise, a variable called P is incremented by one. If P is greater than Q at the end, the LCOM value will be P - Q, otherwise, LCOM is 0. If the values are high, you should consider splitting the affected class. More information about that can be found [7]here.

With a value of 12, the *DatabaseWidget* class has a particularly high value, which requires to take a closer look at that code. However, since the class represents the database entry of a widget, we decided not to make any changes here. The class is only needed to make CRUD operations possible in the software. So, it contains instance variables that can be read or written with the help of getters and setters only. We do not consider changes to be useful since some components require the functionality of this class. Furthermore, it makes no sense to split the

class, because it represents an atomic object in this context of the software.

All in all, it can be said that metrics are a good help to find weaknesses. However, you should not blindly make changes. It is better to evaluate the values accurately and make appropriate decisions that lead to improvement. We would enjoy receiving your comments about our refactoring and the used and unused metrics.

1. <http://sonarcloud.io/>
2. https://github.com/raphaelmue/dashup/blob/master/docs/architectures/testing/metrics_jenkins_screenshot.PNG
3. <https://sonarcloud.io/dashboard?id=dashup>
4. <https://www.aivosto.com/project/help/pm-oo-ck.html>
5. <https://github.com/raphaelmue/dashup/tree/issue112>
6. <https://github.com/raphaelmue/dashup/tree/fe4a4964f4358f86b3296718511ec6bc97874fd1>
7. <https://www.aivosto.com/project/help/pm-oo-cohesion.html#LCOM1>

perfecttime (2019-05-27 10:36:48)

Hey dashup Team, good job for this week! We like that you want to provide classes that are easily understandable and structured on the one hand. On the other hand you aren't just fixing all the issues your metric tool finds but think about the meaningfulness of changes. You Provided a clear overview of the used Methods and tools and also added this in your TestPlan. Grading Criteria seems to be fulfilled :D Best regards your PerfectTime Crew

Christian Schweigel (2019-05-28 07:54:42)

Hello dashup, looks like you did a great work this week, the grading criterias are fulfilled. You gave a detailed description of what you did and the picture and the table give a good overview over the benefits. Stay on with this good work. Best regards, Christian@digiwill

Tenniskönig (2019-05-28 09:49:43)

Hey guys, you have done the Excerice perfectly. It is very good that you show your git before and after the changes. Best regards, Tenniskoening

Indeed, it was a very adventurous journey – dashup (2019-06-17 12:27:26)
[...] Metrics [...]

2.3 June

Installation (2019-06-07 15:17)

This week we managed to deploy our application automatically as part of our continuous integration pipeline.

For our recent deployed version we created an extra branch called "deployment". When merging the master branch into this branch, its pipeline is triggered to start. After all tests and build processes went green, we have a special step that is only possible for this branch, where our Jenkins Server deploys the new version.

If you want to deploy the application on your own server, please follow the instructions in our [1]README. Since this is a full web-based Java application, it's a little harder to install and deploy this application on a server.

Furthermore, our application went online which you can check out [2]here.

We look forward to your feedback!

1. <https://github.com/raphaelmue/dashup/blob/master/README.md>
2. <http://dashup.de/>

Christian Schweigel (2019-06-11 08:32:58)

Hello dashup-Team, thanks for your installation description. I've tried myself to get through and was able to run the application. I just had a few issues. First of all it would be great to have an example for the database.conf to be sure that it is correct. Also the folders where the config needs to be were not present, so I had to create them, which was not mentioned in the installation. And finally the start script is a shell script so it can't be run on windows. I tried it with the Git Bash and it didn't work correctly, so I looked in the script and executed the application manually. So it would be great if you create another script in batch or describe what to do on Windows. Best regards, Christian@DigiWill

Raphael Müßeler (2019-06-11 10:04:34)

Hi DigiWill, thank you for your feedback. We have improved our installation instructions. Kind regards, – dashup

Indeed, it was a very adventurous journey – dashup (2019-06-17 11:14:30)
[...] Installation: Link [...]

Deployment, Continuous Integration & Tech Stack (2019-06-13 18:44)

Hi guys!

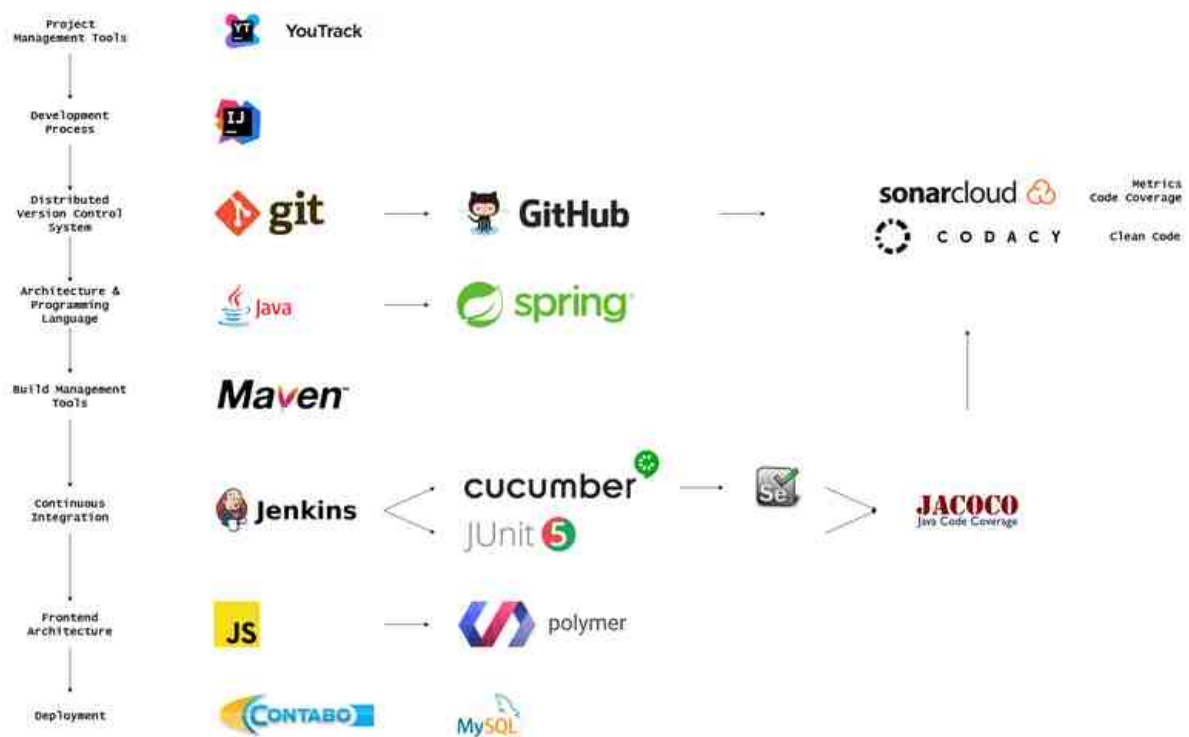
This week we would like to give you an overview of our deployment process, continuous integration services as well as our tech stack.

In the document called Configuration Management Project Plan (CMPP) - which you can check out [1]here - we summarized all information about these topics.

We also defined our contributing and developing process of our application, so that new contributors can easily adapt to these guidelines.

This process goes along not only with our setup for continuous integration but also for our deployment workflow. Both topics are part of our CMPP.

To give you a quick overview of our project and especially the tools, frameworks, and techniques we are using, we have created an overview of our technology stack:



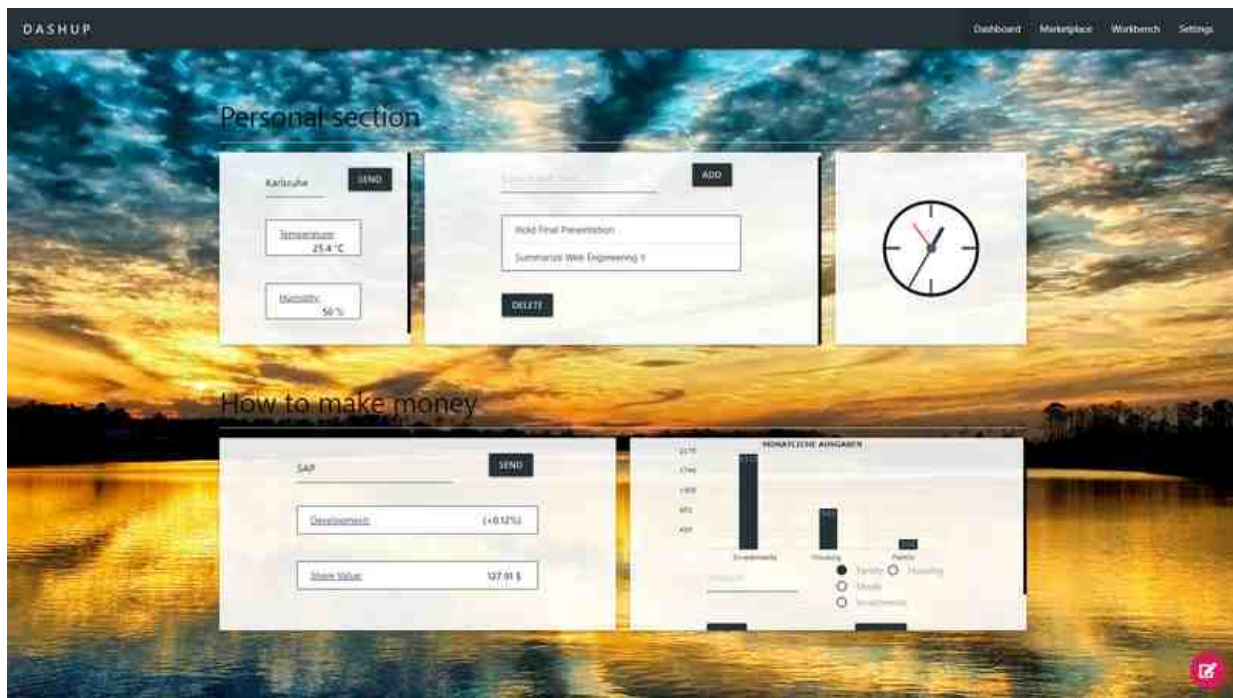
Find more information about our tech stack in our [2]test plan.

I hope, you now have a better understanding of how our application's architecture looks like as well as how our contribution and development process works.

1. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/cmpp/CMPP.md>
2. https://github.com/raphaelmue/dashup/blob/master/docs/architectures/testing/test_plan.md

Indeed, it was a very adventurous journey - dashup (2019-06-17 12:27:28)
 [...] Deployment, Continuous Integration, Tech Stack [...]

Indeed, it was a very adventurous journey (2019-06-17 01:21)



At the beginning of our journey, we stated our mentality:

If there is no enemy within, the enemy outside can do us no harm!

Now after around 850 hours of work, we are proud to the present you:

Dashup

https://www.youtube.com/watch?v=oP-JC_v-2vE

In the end, we had a lot of ups and downs within this project. Dashup was indeed a very adventurous journey we will never forget. The whole team learned a lot from each other and about software engineering in general. We could almost feel how we improve mentally, emotionally and team-spiritually on a day by day basis. Special

thanks to our community and peer review crew, who always supported us through the entire journey! But now the only easy day was yesterday, hail to the brotherhood! Long live [1]dashup!

Final Hand In

References

- [2]GitHub
- [3]YouTrack
- [4]Jenkins
- [5]SonarCloud
- [6]Codacy

Blog Posts

Please preferably use the blog posts listed below to receive the necessary information and documents to be handed in for grading. The blog posts should be more detailed than the list of shortcuts afterward.

1. [7]Project vision
2. [8]Distribution of roles, Technology Stack, RUP
3. [9]Software Requirements Specification
4. [10]Use Case Specifications
5. [11]Gherkin Feature Files And Selenium Testing
6. [12]DB Scheme And Class Diagram
7. [13]Scrumming With YouTrack And Project Management
8. [14]Scrum Retrospective

9. [15]MVC Architecture
10. [16]Confirmation Of Scope
11. [17]Risk Management
12. [18]Time Estimation With Function Points
13. [19]Testing
14. [20]Refactoring
15. [21]Design Patterns
16. [22]Scrum Retrospective II
17. [23]Test Coverage
18. [24]Metrics
19. [25]Installation
20. [26]Deployment, Continuous Integration, Tech Stack

Requirements

Use Cases:

- [27]Change Layout
- [28]Change Panel Structure
- [29]Marketplace
- [30]Change Profile
- [31]Login / Register
- [32]Widgets
- [33]Workbench

Software Requirement Specification: [34][Link](#)

Test Cases: Linked in each use case under "Narrative"

Test Logs:

- [35]Surefire Test Logs (Download and open in browser)
- [36]Jenkins (Choose any build you want and click on "Tests" at the top right corner or inspect the console log)

Test Coverage:

- [37]SonarCloud
- [38]Jenkins (Choose any build you want)
- [39]Codacy (Not up to date)

Test Plan: [40]Link (includes functional tests, units tests, and other tests)

Configuration Management Project Plan: [41]Link (Lifecycle management tools and tech stack)

Project Management

Function Point Calculation: [42]Link

Burndown Chart:

- [43]3rd semester
- [44]4th semester

Cumulative Flow:

- [45]3rd semester
- [46]4th semester

Gantt Chart:

- [47]3rd semester
- [48]4th semester

Time Report Per Person: (hours/team member breakdown)

- [49]3rd semester
- [50]4th semester

Time Report Per Phase:

- [51]3rd semester
- [52]4th semester

Time Report Per Use Case:

- [53]3rd semester
- [54]4th semester

Time Report Per Workflow:

- [55]3rd semester
- [56]4th semester

Ability To Execute

Demo: [57][Link](#)

Code: [58][Link](#)

Installation: [59][Link](#)

Deployment: [60][Link](#)

Quality

Software Architecture Document: [61][Link](#)

Configuration Management Project Plan: [62][Link](#) (Configuration management and environmental setup)

Metrics: [63][Link](#)

Risk Management: [64][Link](#)

Automated Testing:

- [65][Test Plan](#)
- [66][Configuration Management Project Plan](#)

Pattern: [67][Link](#)

Other

Presentation: [68][Link](#)

Handout: [69][Link](#)

Tech Stack: [70][Link](#)

Guides: [71][Link](#)

Dashup components documentation: [72][Link](#)

Your dashup core developer team

Raphael, Joshua, Sven and Felix

1. <http://dashup.de/login>
2. <https://github.com/raphaelmue/dashup>
3. <https://youtrack.dashup.de/issues>
4. <https://jenkins.rafael-muesseler.de/blue/organizations/jenkins/dashup/activity/>
5. <https://sonarcloud.io/dashboard?id=dashup>
6. <https://app.codacy.com/project/dashup/dashup/dashboard>
7. <https://dashup2k18.wordpress.com/2018/10/10/our-project-vision/>
8. <https://dashup2k18.wordpress.com/2018/10/14/distribution-of-roles-technology-stack-and-more/>
9. <https://dashup2k18.wordpress.com/2018/10/21/software-requirements-specification/>
10. <https://dashup2k18.wordpress.com/2018/10/28/use-case-specifications/>
11. <https://dashup2k18.wordpress.com/2018/11/04/gherkin-feature-files/>
12. <https://dashup2k18.wordpress.com/2018/11/11/db-scheme-and-class-diagram/>
13. <https://dashup2k18.wordpress.com/2018/11/18/scrummy-with-youtrack/>
14. <https://dashup2k18.wordpress.com/2018/11/23/scrum-retrospective/>
15. <https://dashup2k18.wordpress.com/2018/12/02/mvc-architecture/>
16. <https://dashup2k18.wordpress.com/2019/04/02/welcome-back/>
17. <https://dashup2k18.wordpress.com/2019/04/07/risk-management/>
18. <https://dashup2k18.wordpress.com/2019/04/22/time-estimation-with-function-points/>
19. <https://dashup2k18.wordpress.com/2019/04/26/testing/>
20. <https://dashup2k18.wordpress.com/2019/05/05/refactoring/>
21. <https://dashup2k18.wordpress.com/2019/05/13/design-patterns/>
22. <https://dashup2k18.wordpress.com/2019/05/17/scrum-retrospective-ii/>

23. <https://dashup2k18.wordpress.com/2019/05/20/code-coverage/>
24. <https://dashup2k18.wordpress.com/2019/05/26/metrics/>
25. <https://dashup2k18.wordpress.com/2019/06/07/installation/>
26. <https://dashup2k18.wordpress.com/2019/06/13/deployment-ci-tech-stack/>
27. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_layout/UCS_change_layout.md
28. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/layout/change_panel_structure/UCS_change_panel_structure.md
29. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/marketplace/UCS_marketplace.md
30. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/change_profile/UCS_change_profile.md
31. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/user_management/login_register/UCS_login_register.md
32. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/widgets/widget/UCS_widget.md
33. https://github.com/raphaelmue/dashup/blob/master/docs/specifications/ucs/workbench/UCS_workbench.md
34. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/srs/SRS.md>
35. https://github.com/raphaelmue/dashup/tree/master/docs/logs/surfire_test_logs
36. <https://jenkins.raphael-muesseler.de/blue/organizations/jenkins/dashup/activity>
37. <https://sonarcloud.io/dashboard?id=dashup>
38. <https://jenkins.raphael-muesseler.de/job/dashup/job/master/161/jacoco/>
39. <https://app.codacy.com/project/dashup/dashup/dashboard>
40. https://github.com/raphaelmue/dashup/blob/master/docs/architectures/testing/test_plan.md
41. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/cmpp/CMPP.md>
42. https://github.com/raphaelmue/dashup/tree/master/docs/project_management/time_estimation
43. <https://youtrack.dashup.de/reports/burndown/123-4>
44. <https://youtrack.dashup.de/reports/burndown/123-5>
45. <https://youtrack.dashup.de/reports/cumulativeFlow/124-6>
46. <https://youtrack.dashup.de/reports/cumulativeFlow/124-7>
47. <https://youtrack.dashup.de/reports/gantt/128-3?view=actual>
48. <https://youtrack.dashup.de/reports/gantt/128-2?view=actual>
49. <https://youtrack.dashup.de/reports/time/116-14>
50. <https://youtrack.dashup.de/reports/time/116-25>
51. <https://youtrack.dashup.de/reports/time/116-16>
52. <https://youtrack.dashup.de/reports/time/116-27>
53. <https://youtrack.dashup.de/reports/time/116-13>
54. <https://youtrack.dashup.de/reports/time/116-19>
55. <https://youtrack.dashup.de/reports/time/116-15>
56. <https://youtrack.dashup.de/reports/time/116-18>
57. <http://dashup.de/login>
58. <https://github.com/raphaelmue/dashup/archive/master.zip>
59. <https://dashup2k18.wordpress.com/2019/06/07/installation/>
60. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/cmpp/CMPP.md>
61. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/sad/SAD.md>
62. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/cmpp/CMPP.md>
63. https://github.com/raphaelmue/dashup/blob/master/docs/architectures/testing/test_plan.md
64. https://github.com/raphaelmue/dashup/tree/master/docs/project_management/risk_management
65. https://github.com/raphaelmue/dashup/blob/master/docs/architectures/testing/test_plan.md
66. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/cmpp/CMPP.md>
67. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/sad/SAD.md>
68. <https://github.com/raphaelmue/dashup/tree/master/docs/presentations/final>

- 69. https://github.com/raphaelmue/dashup/blob/master/docs/project_management/final_preparation/handout.pdf
 - 70. <https://github.com/raphaelmue/dashup/blob/master/docs/specifications/cmpp/CMPP.md>
 - 71. <https://github.com/raphaelmue/dashup/tree/master/docs/guides>
 - 72. <https://github.com/raphaelmue/dashup/tree/master/docs/specifications/ucs/widgets/components>
-