Contents

1. 2020

1.1 October

Week 1 - Our Vision (2020-10-01 08:51) - davideb3

Do you too need a motivation for your daily tasks? We do! So we thought of an Android-App that you can use to motivate yourself by earning free time while finishing tasks.

With Work2Play you are able to create a To Do list to schedule tasks of your every day life. For each completed tasks you can earn credits. You can use these credits to reward yourself by"*buying*" free time for activities you enjoy.

If you have assignments with a deadline or repeating events you could plan them in a calendar and set a date and time to get an early reminder.

In addition to that, it is possible to share tasks with your friends or roommates to work on projects simultaneously.

This is the first post of our project Work2Play - Together. Further links and information will come soon.

Week 2 - Organizing and Setup (2020-10-08 17:50) - rbnsch00

Let‘s get started!

We want to introduce our Development-Team of Work2Play - Together. We are organized in following roles:

Discipline Role Responsible
Project Management Project Manager David Eymann
Deployment Deployment Manager David Eymann
Requirements Requirements Specifier Tom Wagner
Design Designer Tom Wagner
Implementation Implementer Robin Schmidt
Test Test Designer/Tester Robin Schmidt, Tom Wagner

This week we build the foundation for the development of our App by creating a [1]Doku-Repository, a [2]Code-Repository (both on GitHub) and a [3]YouTrack-Project as organization tool. Feel free to visit.

We plan to use:

- Java (Jetbrains IntelliJ IDEA)
- Cucumber and Mockito for testing

Keep in mind that everything here is work in progress and we will post updates regularly. Stay put and we hope to see you next week.

1. https://github.com/david2628/Work2Play
2. https://github.com/rbnsch/Work2Play
3. https://dhbw-karlsruhe.myjetbrains.com/youtrack/issues/W2PT

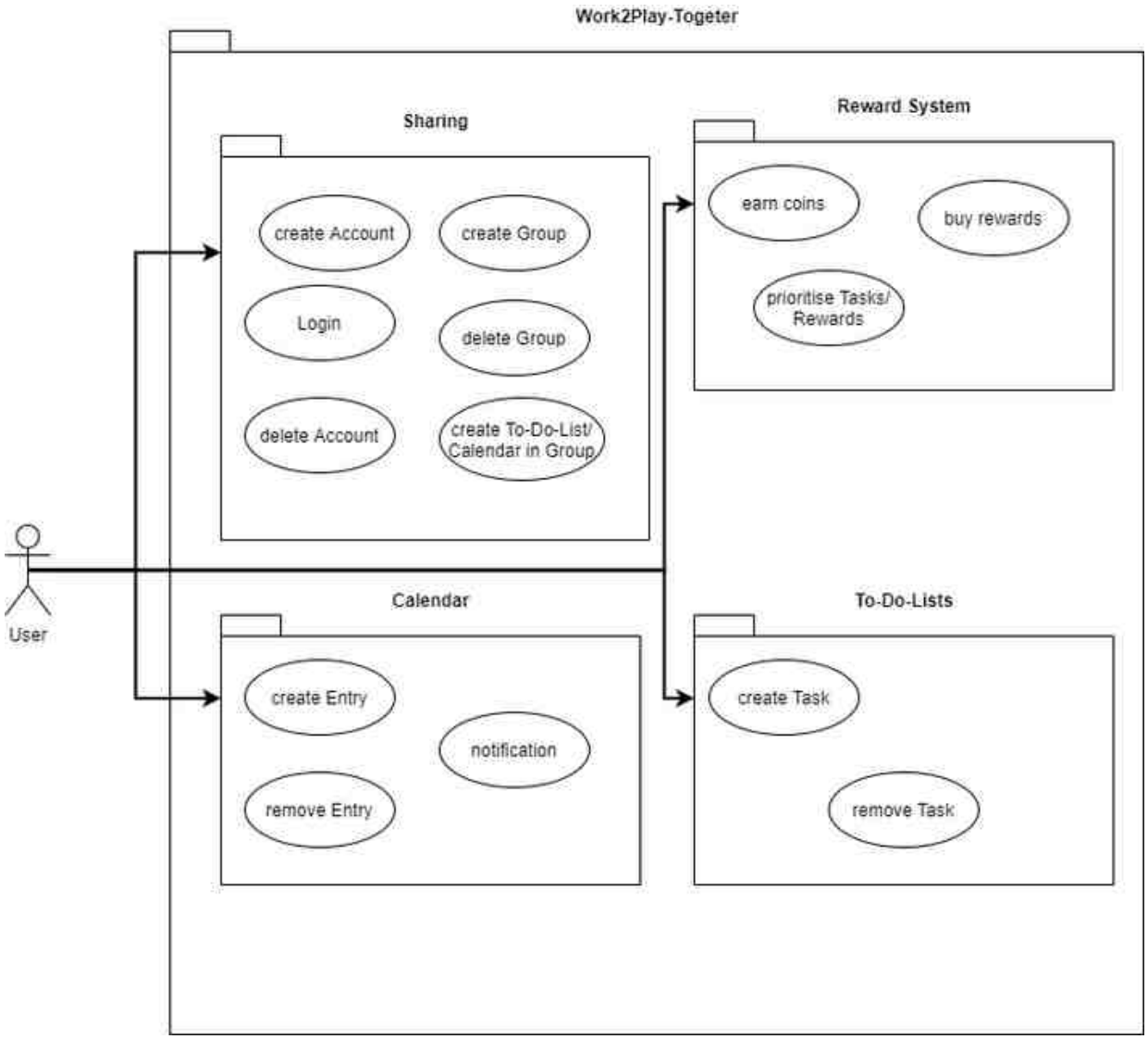
Week 3 - Software Requirements Specification (2020-10-19 18:53) - tomwb3

The foundation is set!

This week we started to set up our SRS (Software Requirements Specification). This will help to develop our ideas and give you a more detailed look on what we are planning to do with Work2Play - Together. Feel free to visit our SRS here:

https://github.com/david2628/Work2Play/blob/main/SRS.md

As you can see, nothing is set in stone yet, for the lazy guys who don #t want to read through it all, here is our UCD (Use Case Diagram):



Thats it for this week. Thanks for reading and we hope to see you in near future.

Your Work2Play-Team

Week 4 - First Use-Cases (2020-10-26 21:04) - tomwb3

Hello Guys, welcome back to week 4!

Today we want to show you the concepts of the first two Use-Cases for Work2Play - Together. Feel free to check them out.

1) Create a new task for your To-Do list:

https://github.com/david2628/Work2Play/blob/main/UC-Create _Task _CRUD.md

2) Spend earned Coins to buy Rewards:

https://github.com/david2628/Work2Play/blob/main/UC-Buy _Rewards.md

As always, these are only ideas and concepts so they can and possibly will change in the future.

We would love to hear your opinion and feedback.

See you next week,
your Work2Play - Team

1.2 November

Week 5 - Feature Files (2020-11-02 18:15) - rbnsch00

Hey Guys, welcome back to week 5!

Remember our Use-Cases from last week?
We added our first Cucumber .feature Files to them. You can find the links to the UseCases below and the .feature Files under paragraph 2.1

[1]CRUD-Habit

[2]CRUD-Task

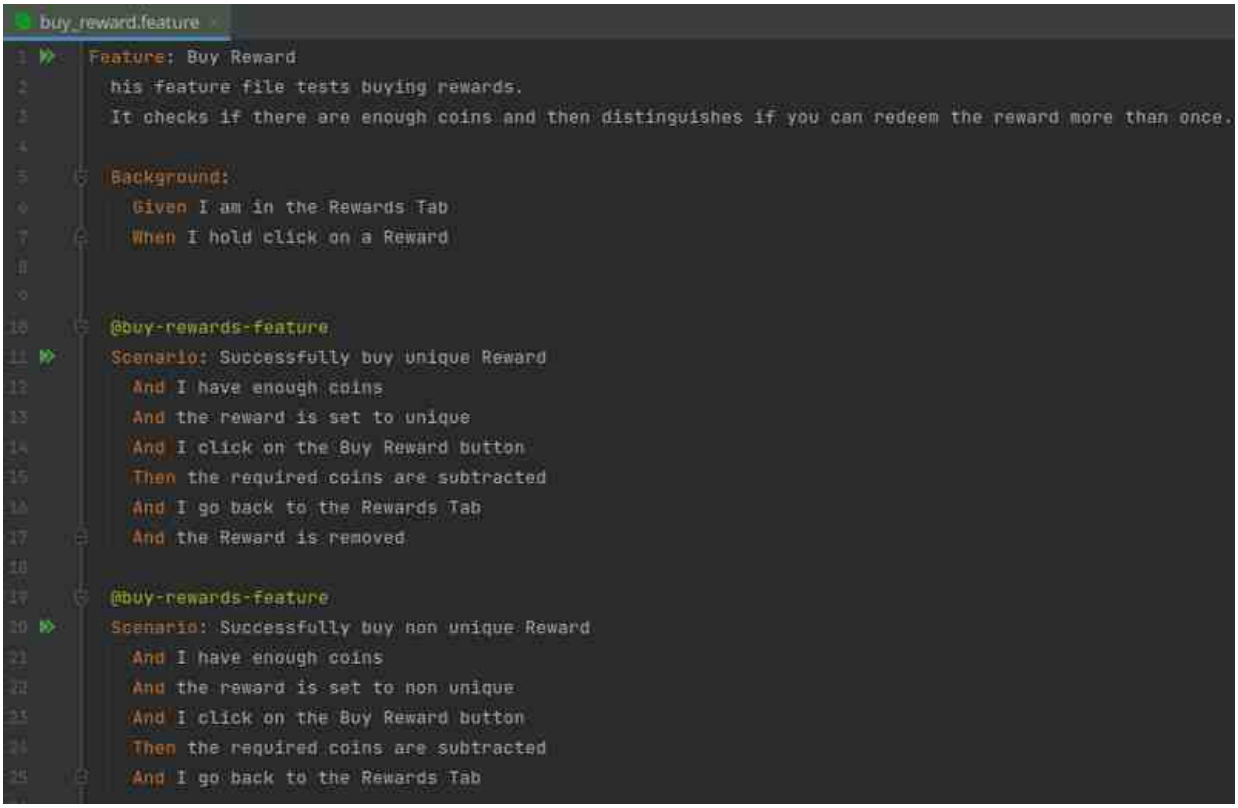
[3]CRUD-Reward

B[4]uy-Rewards

Fini[5]sh-Habit

[6]Finish-Task

To implement the feature-files in our Project, we first installed in our IDE the Gherkin Plugin. Now we have a nice Syntax-Highlighting in our Feature-Files:



Feature-Files in Android Studio

To implement the Step Definitions we used Espresso.

The following video shows the Feature-Files from Buy-Rewards and Finish-Taks running:

https://www.youtube.com/watch?v=lMHLe8PuL6A

As always, these are only ideas and concepts so they can and possibly will change in the future.

We would love to hear your opinion and feedback.

See you next week,
your Work2Play - Team

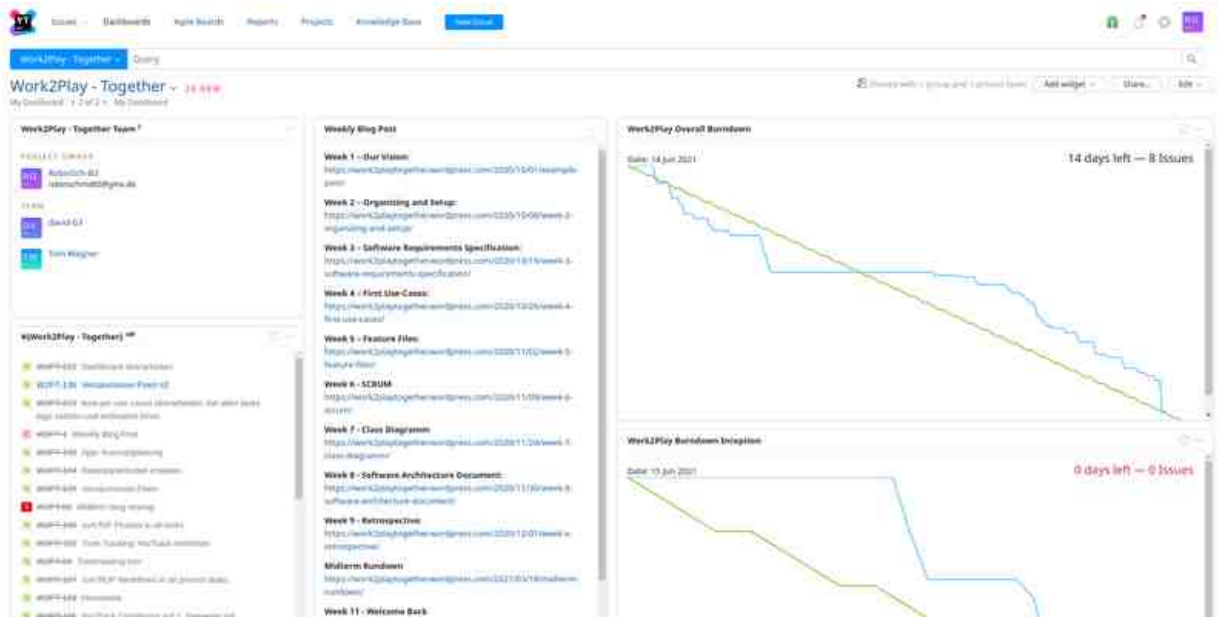
1. https://github.com/david2628/Work2Play/blob/main/UseCases/CRUD-Habit.md
2. https://github.com/david2628/Work2Play/blob/main/UseCases/CRUD-Task.md
3. https://github.com/david2628/Work2Play/blob/main/UseCases/CRUD-Reward.md
4. https://github.com/david2628/Work2Play/blob/main/UseCases/UC-Buy_Rewards.md
5. https://github.com/david2628/Work2Play/blob/main/UseCases/UC-Finish_Habit.md
6. https://github.com/david2628/Work2Play/blob/main/UseCases/UC-Finish_Task.md

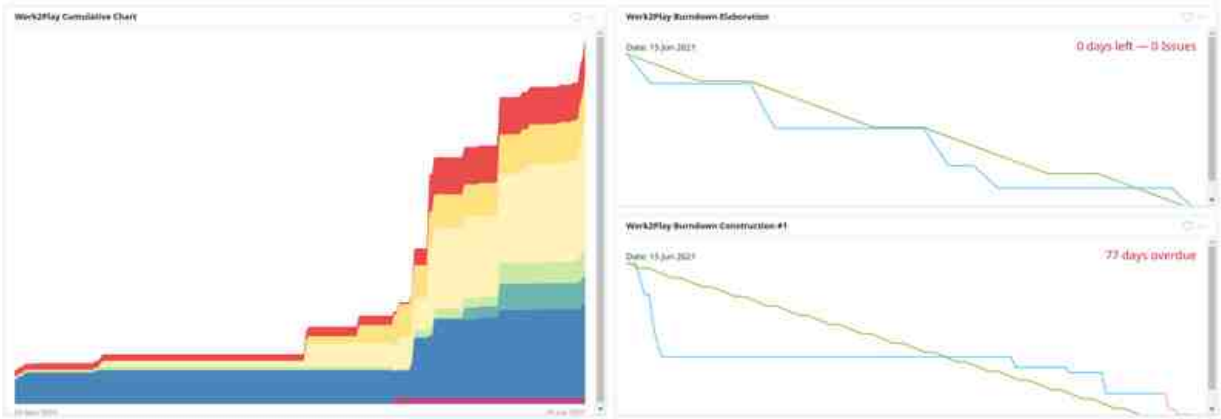
Week 6 - Scrum (2020-11-09 18:37) - tomwb3

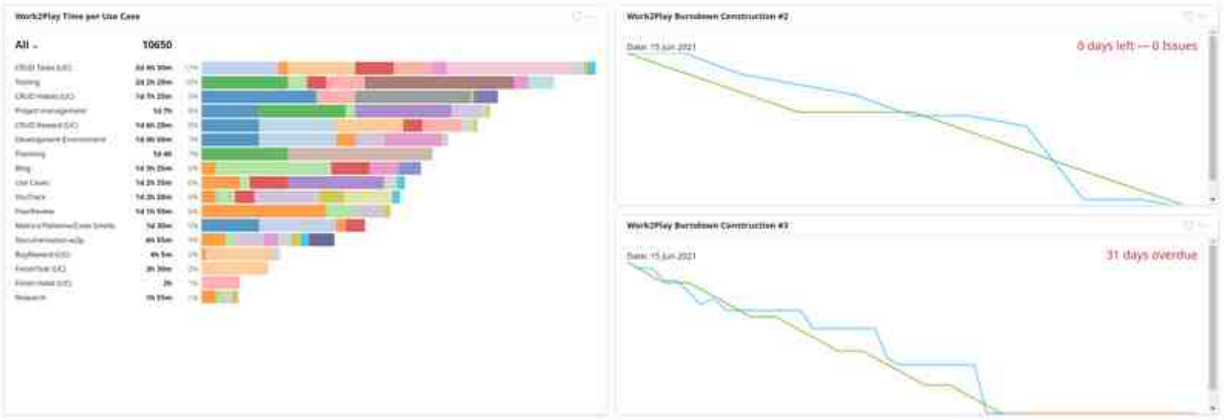
Welcome back to our Blog.

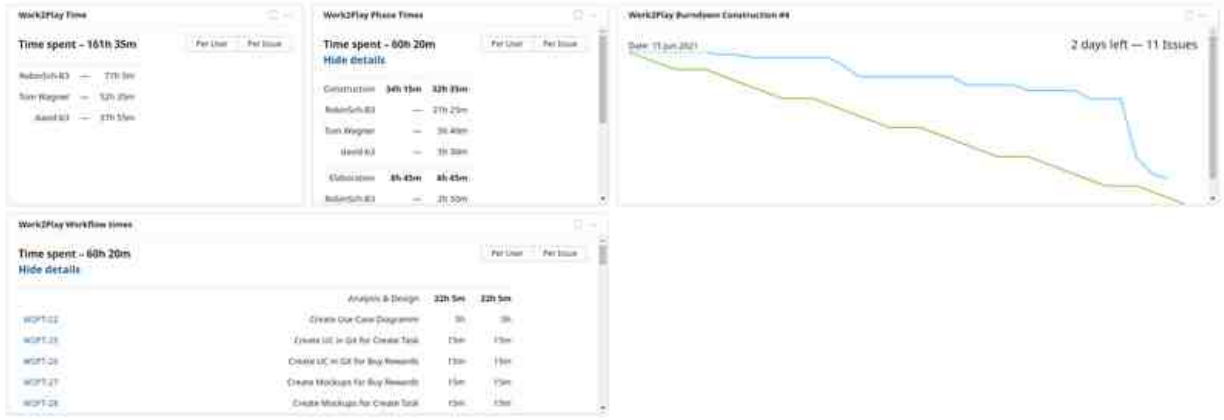
For this week, we started to organize our Sprints with the Scrum Framework.

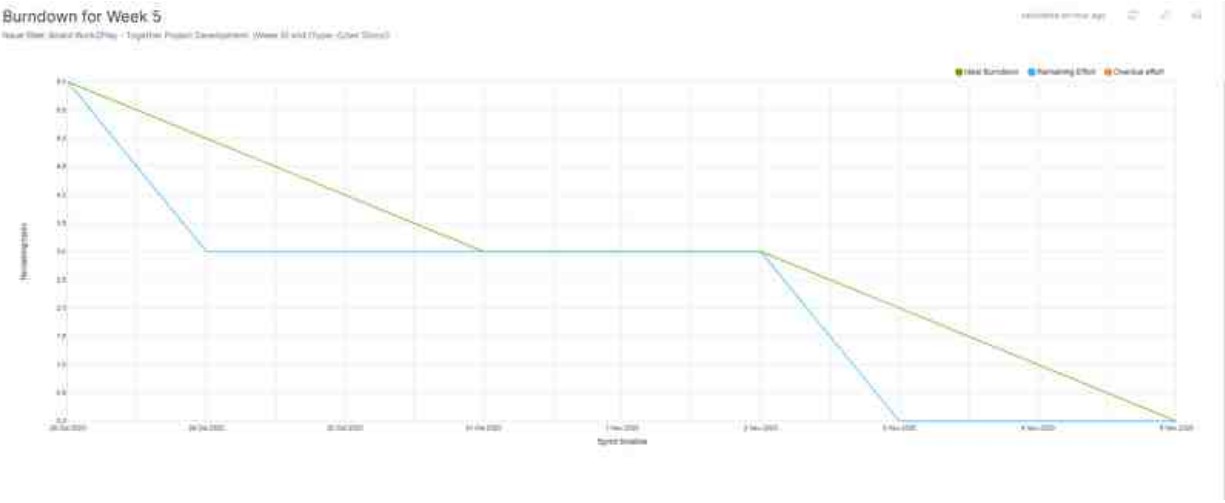
Therefore we created a YouTrack Dashboard and Agile Board where we write down our tasks and track our progress.







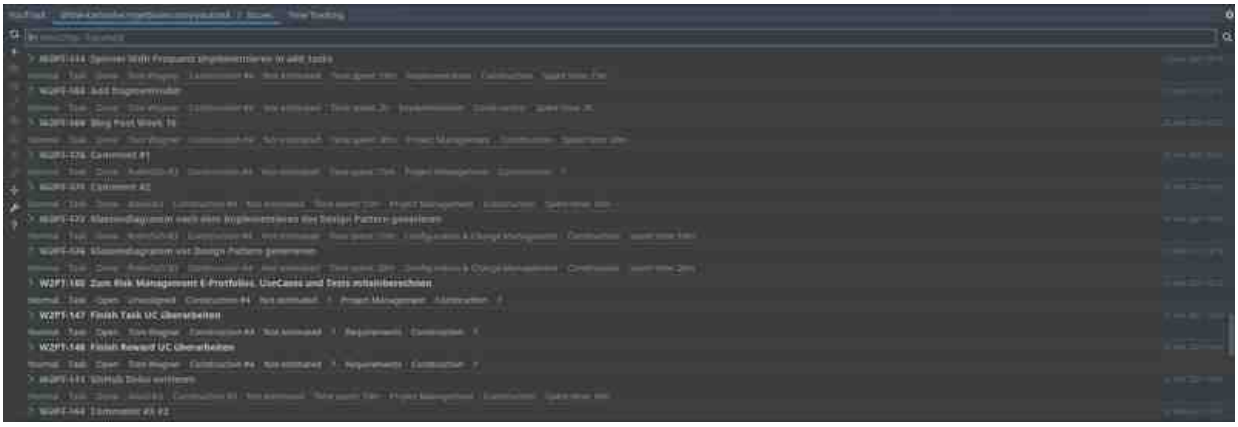




Feel free to check it out:

https://dhbw-karlsruhe.myjetbrains.com/youtrack/dashboard?id=186e6d48-444e-48 30-af90-6e1e9a86bfd8

Edited 11.11.2020:
We finaly got IntelliJ and YouTrack connected. Check out this tutorial to get around using a token. [1]https://www.jetbrains.com/help/youtrack/ standalone/YouTrack-Integration-Plugin.html



As always, everything can be changed in the future and we appreciate your feedback.

Thanks for reading, see you next week.

Your Work2Play - Team

1. https://www.jetbrains.com/help/youtrack/standalone/YouTrack-Integration-Plugin.html

Week 7 - Class Diagramm (2020-11-24 09:34) - rbnsch00

Hey Guys, welcome back to week 7!

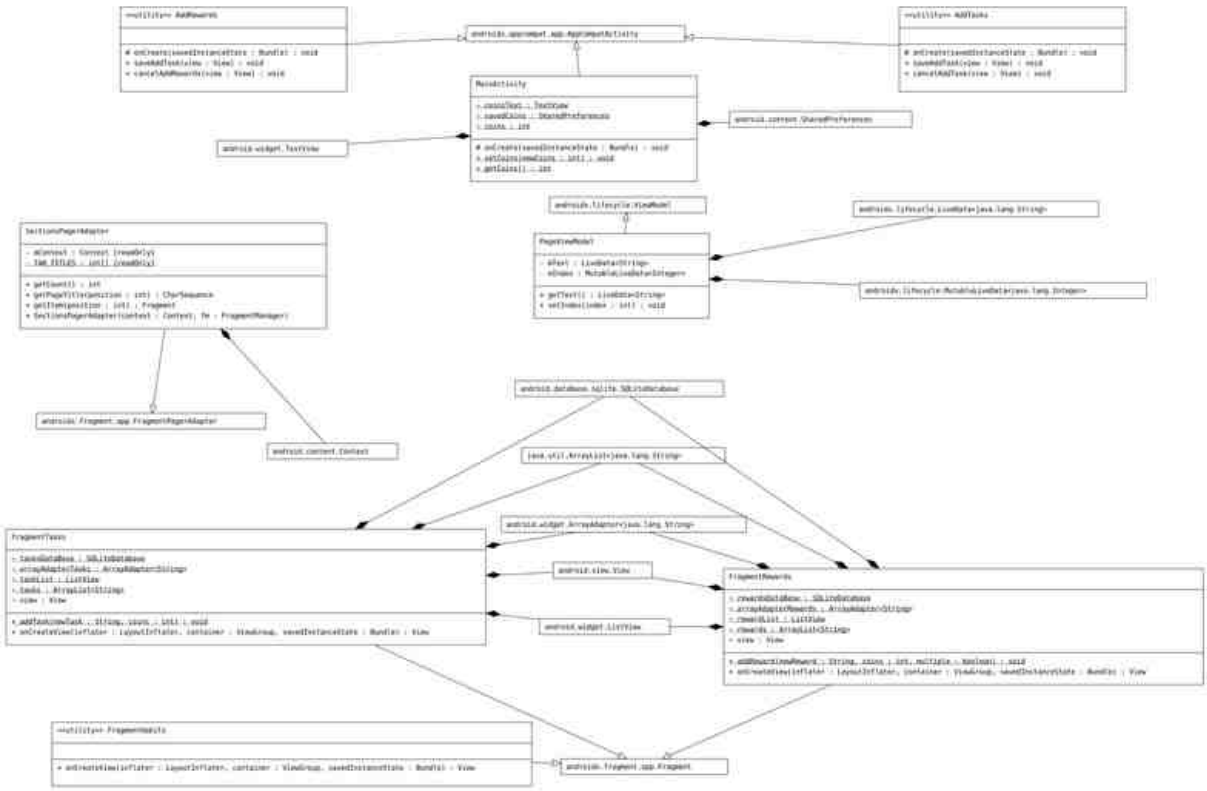
This Week we created a Class-Diagramm.

We creatred the diagramm like our other diagramms wirh DrawIO [1]https://app.diagrams.net/

In Android Apps there are different Activitys. You can imangine every Screen you see as a different Activity. In the Android Architecture every Activty extends of the AppCompatActivity. The "Coins", "Reward" and Taks" classes are handeling our data.

Update 4th Dec 2020:

Now we found a way to generate our class diagramms. For InteliJ there is a a Plug-In called "UML Generator". This is our new generated diagramm:

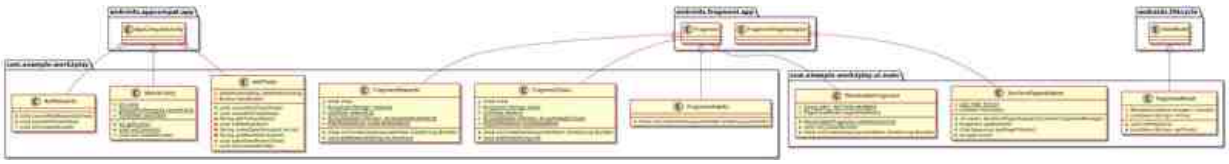


Update 7th Jan 2021:

The big disadvantage with "UML Generator" is that you have to sort your classes after generating, because a lot of them are overlapping and some useless stuff is also shown.

So now we switched to "PlantUML". With this Plug-In you can generate your class diagramm out of a text file where you have to manually write down all your classes and stuff. This is not very effective, so we found the Plug-In "PlantUML Parser". Now we only have to right click on our package and click on "PlantUML Parser" and a new PlantUML with all our classes is generated.

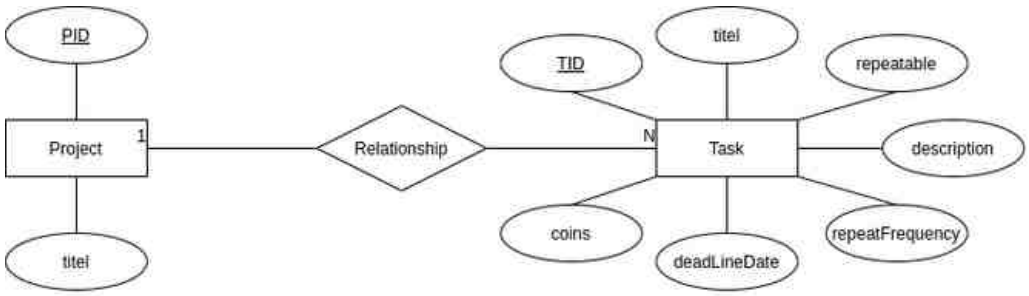
Here is our latest diagramm:

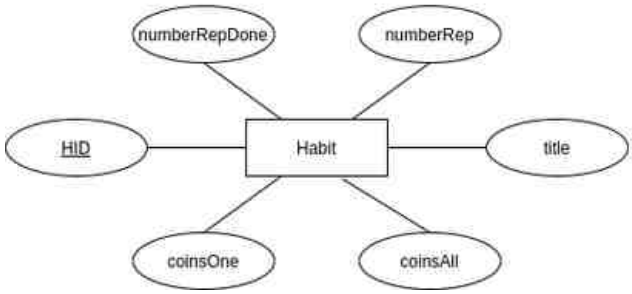


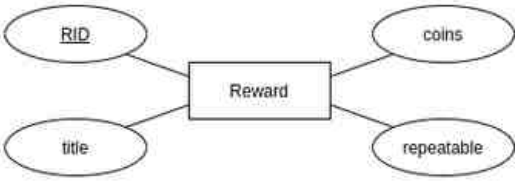
In the future you will find our new class diagramms in our GitHub:

https://github.com/david2628/Work2Play/tree/main/ClassDiagramms

Our App uses a local SQLite Database. So here is our ER-modell:







Your can find our latest diagramms always in our [2]Software Architecture Document.

As always, everything can be changed in the future and we appreciate your feedback.

Thanks for reading, see you next week.

Your Work2Play - Team

1. https://app.diagrams.net/
2. https://github.com/david2628/Work2Play/blob/main/SAD/SAD.md

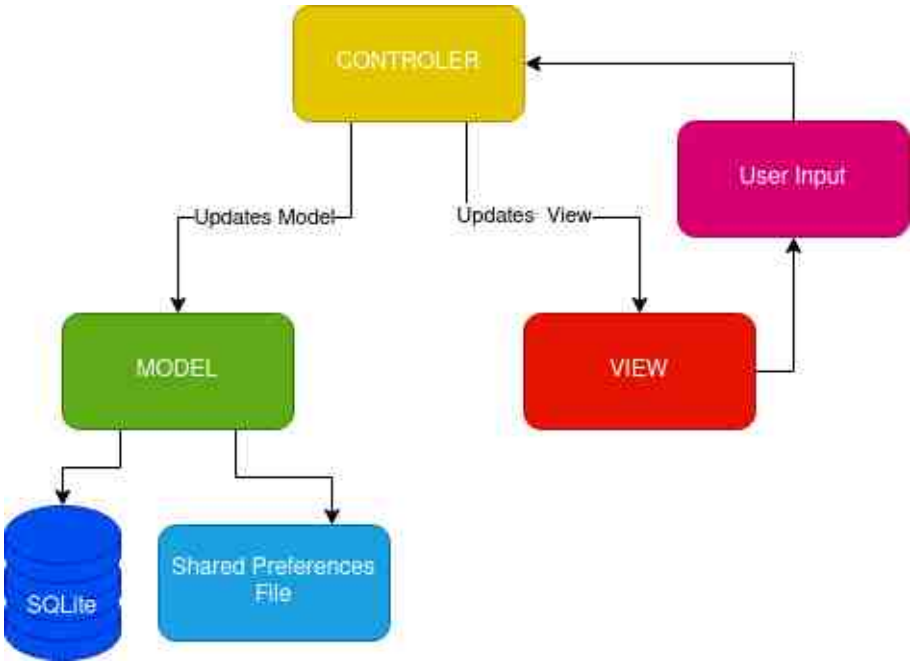
Week 8 - Software Architecture Document (2020-11-30 00:56) - davideb3

Hey guys,

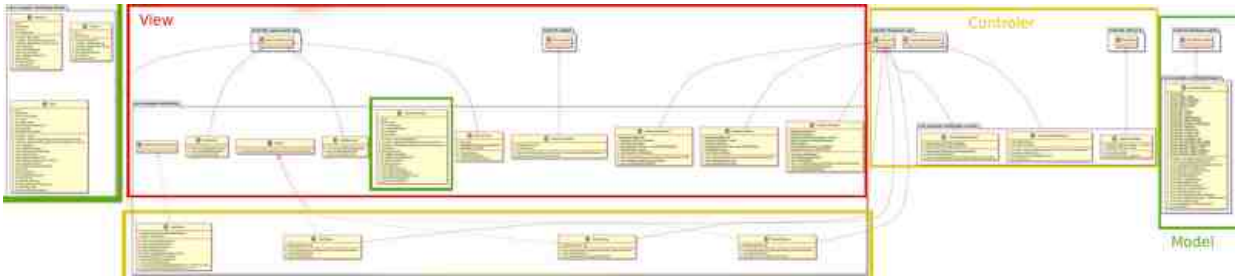
in this week we had to make an [1]SAD to showcase our architecture. As you can see in it we are using MVC as our architectural pattern.

As framework we choose Android Studio. It allows to create Native Android Apps with Java or Kotlin.

Android Studio do not support to generate CRUDs.



Here is our Class Diagramm with the Overlay of the MVC:



A Video with our you can find at our [2]Feature Files Post.

Unfortunately, we have not yet managed to generate ER diagrams and thus still have to draw them manually. Does anyone know how to link DataGrip with Android Studio and a SQLite database? Thanks a lot

Your Work2Play

1. https://github.com/david2628/Work2Play/blob/main/SAD/SAD.md
2. https://work2playtogether.wordpress.com/2020/11/02/week-5-feature-files/

1.3 December

Week 9 - Retrospective (2020-12-01 22:42) - tomwb3

Hello guys and welcome to another blog post.

Today, we want to share our experiences of the first Retrospective of this project.

In a Retrospective, the team reflects on the process of the project in the past and agrees on what went well, and where are possibilities to improve. This knowledge should then be used to increase productivity and quality of the project and improve the atmosphere in the Team.

We agreed on the following points:

What went well:

- Team Communication
- Team Delegation
- Finishing homework/ tasks on time
- Working out requirements
- Planning future Use Cases

What can be improved:

- Time management
- Team motivation/ work ethic
- Fairer distribution of tasks
- Getting every Member to the same ratio of Programming - Management

As you can see, some things already work out pretty well, but there is still a lot to work on. Nevertheless, we won’t stop trying to improve, to get Work2Play as successful as possible.

Thank you all for coming with us on this way and we’ll see you next week.

Your Work2Play- Team

2. 2021

2.1 March

Week 10 - Midterm-Rundown (2021-03-18 20:30) - rbnsch00

Hello and welcome back to our Blog.

As the first half of our project is finished, here is your chance to get up to date with the current state of Work2Play.

Requirements

First of all, down below you can find all of our Use-Case Specifications to give you an overview on what we have planned and also what is about to come.

Feel free to check regulary, these documents will always get updated and improved.
Feel also free to leave feedback and comment any ideas or questions that might occur.

https://github.com/david2628/Work2Play/tree/main/UCs

Also take a look at our SRS which will also be extended and updated as the development moves on.

https://github.com/david2628/Work2Play/blob/main/SRS.md

Our .feature files can be found in our Use Cases.

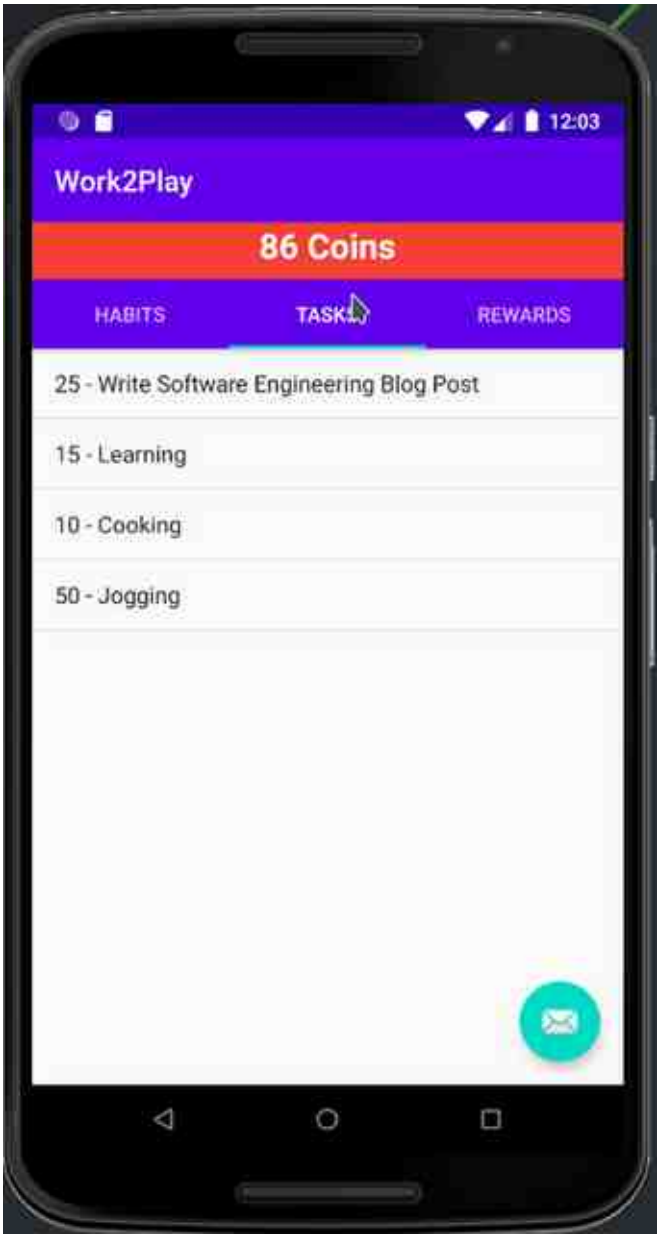
Project management

Next up we have improved our Youtrack-Dashboard and made use of the time tracking for every team member.
There you can follow our every steps and watch the development of Work2Play first hand. On the dashboard you can also find our Burndown and our Cumulative Flow Chart.

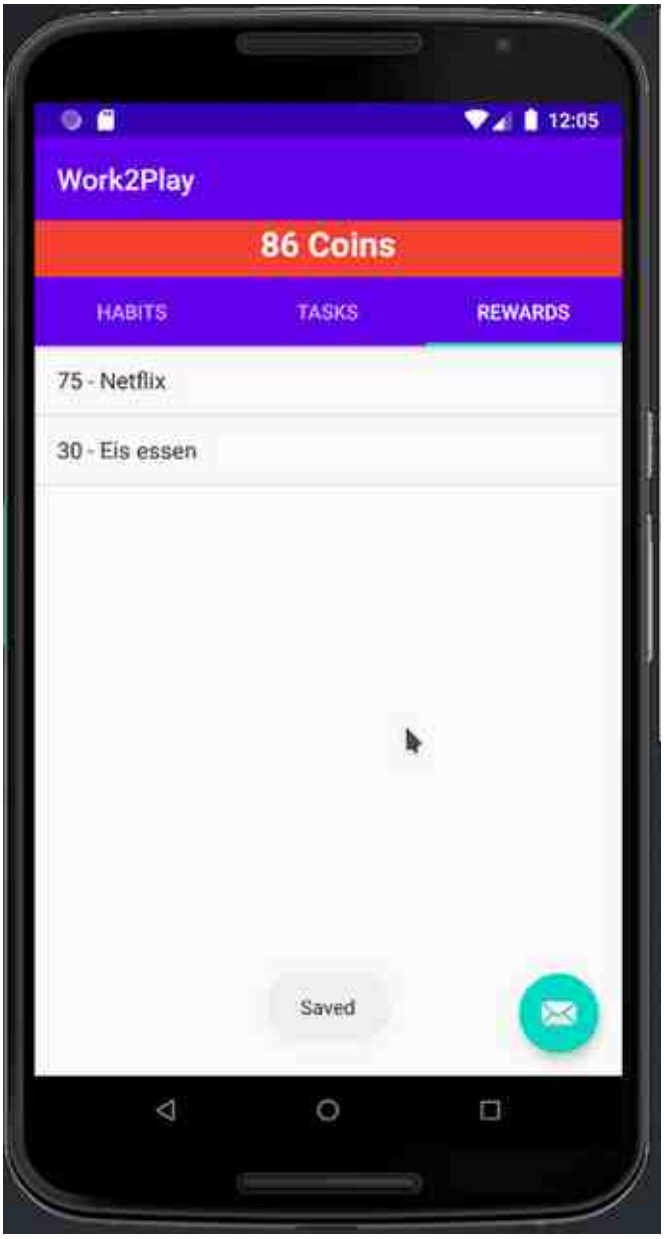
https://dhbw-karlsruhe.myjetbrains.com/youtrack/dashboard?id=186e6d48-444e-48 30-af90-6e1e9a86bfd8

Ability to execute

Here are some Screenshots of our latest version of the app:



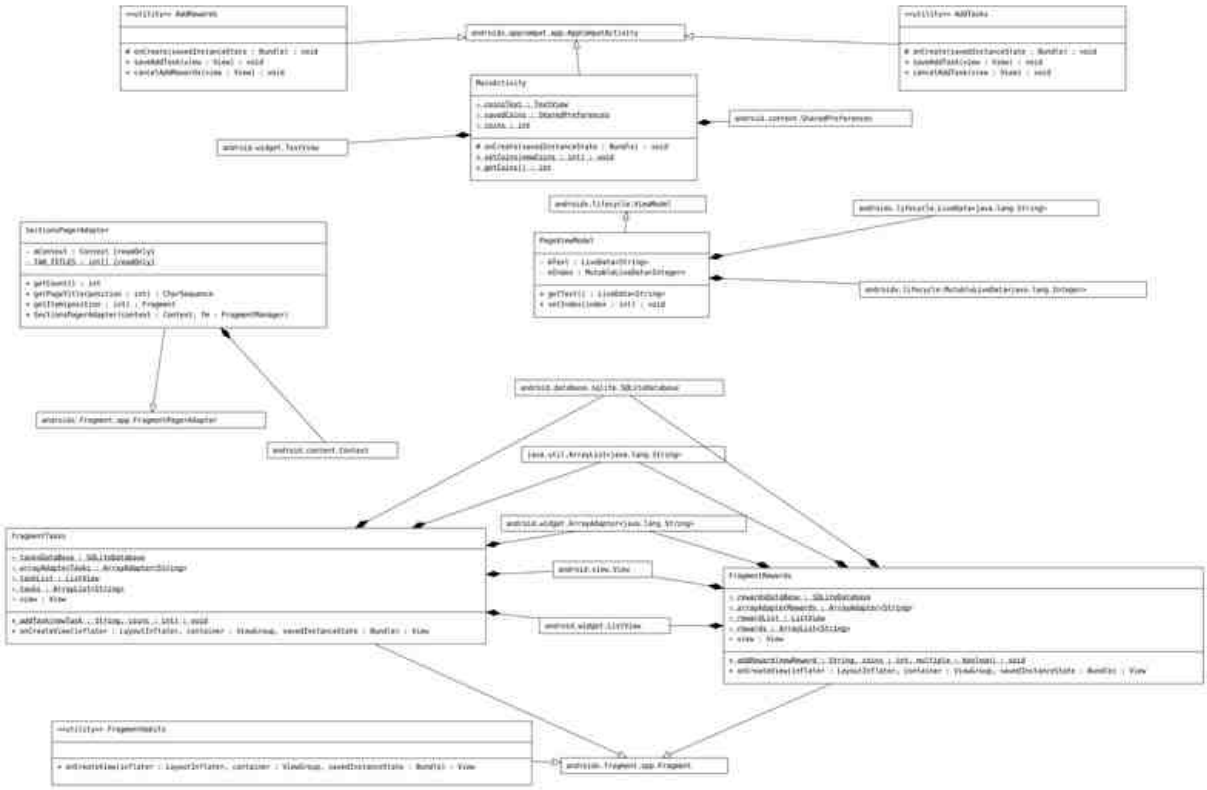

12:05
Work2Play
Welche Belohnung möchtest du hinzufügen?
Netflix
Wie viele Coins kostet deine Belohnung?
75
Mehrfach einlösbar?
CANCEL
SAVE



The code can be found on our code-github:

https://github.com/rbnsch/Work2Play

At last here is our latest Class Diagramm:



For some more infortmation, here you can find our Midterm presentation and the additional Handout for it.

https://github.com/david2628/Work2Play/tree/main/Midterm

That’s it for this post and this semester actually. Thank you for going this path with us and we hope to see you again in the next half.

Your Work2Play-Team

2.2 April

WEEK 11 - WELCOME BACK (2021-04-10 17:05) - tomwb3

Hello guys and welcome back to our blog about Work2Play - Together.

A new semester has started and we are motivated to continue our work of the previous months.

You can also expect weekly updates in form of new posts in order to keep you up to date with the latest state of this project.

Of course we did not do nothing in the past weeks, so here are the links to our Github and our Youtrack dashboard for you to get back on track.

Github:

https://github.com/david2628/Work2Play

Youtrack:

https://dhbw-karlsruhe.myjetbrains.com/youtrack/dashboard?id=186e6d48-444e-48 30-af90-6e1e9a86bfd8

And thats it for this quick announcement this week. Thank you for walking this path with us and we'll see us next week!

Your Work2play - Team

Week 12 - Risk Management (2021-04-11 16:48) - rbnsch00

Welcome back to our Blog.

In this week we dealed with our Risk Management. This is a very important part of Software Engineering, because if you don’t do it, you can get a lot of trouble with unexpected problems.

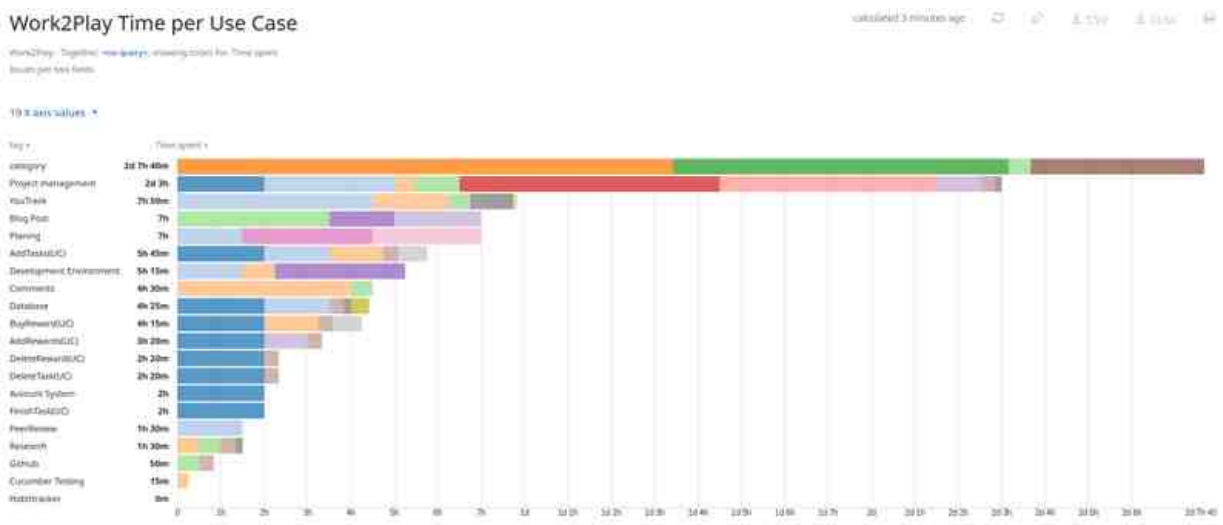
As a consequence we examined our risks, that could possible occur. To interpret our risks we estimated for every item a probability of occurrence and a damage value. With these there is a Risk Factor calculated to show which risks we have to care the most. Here is a Screenshot of our results:

	A	B	C	D	E	F	G
1	Risk	Description	Prob. Of Occurrence	Damage	Mitigation Strategy	Person in Charge	Risk Factor
2	Project involves the use of technology that has not been used prior project		10	6	Learn the technology	everyone	60
3	Team members lack of specialized skill required by the project		9	5	aquire specialized skills	everyone	45
4	Inexperience project management		9	4	collect experience	David	36
5	Inexperience team members		9	4	we are students	everyone	36
6	Inadequate estimation of required resources		7	5	collect experience	David	35
7	High level of technical complexity		5	6	break down complexity	Robin	30
8	Project mile stone not clearly defined		9	3	planning mile stones	David	27
9	Project involves the use of new technology		9	2	using things we know, watching tutorials and asking other groups for help	Tom	18
10	Continually changing requirements		2	7	working agile	Tom	14
11	Ineffective communications		2	7	planed meetings	Tom	14
12	Unclear system requirements		4	3	write requirements down and communicate	Robin	12
13	Project progress not monitored closely enough		3	4	more excessive use of YouTrack	Robin	12
14	Poor project planning		3	4	planning together better	Tom	12

The excel sheet you can also download on our GitHub:

https://github.com/david2628/Work2Play/blob/main/risk _management/risk _management.xlsx

In addition we created a Time per Use Case Diagramm on YouTrack:



It helps us to keep track of how many time we spend per Use Case. You can find it on our Dashboard. Feel free to visit!

https://dhbw-karlsruhe.myjetbrains.com/youtrack/dashboard?id=186e6d48-444e-48 30-af90-6e1e9a86bfd8

Thats it for this week. Thanks for reading and we hope to see you in near future.

Your Work2Play-Team

Week 13 - Function Points (2021-04-18 17:51) - rbnsch00

Hey guys,

in this week we have been working on Function Points. These points we calculate for every Use Case to get a prediction, how long a Use Case will take to implement.

The first step is to fill out the Complexity Adjustment Table. It helps to get more accurate Function Points and is the same on every Use Case in a project.

Complexity Adjustment Table

ITEM	COMPLEXITY ADJUSTMENT QUESTIONS	SCALE (No Influence 0 1 2 3 4 5 Essential)
1	Does the system require reliable backup and recovery?	4
2	Are data communications required?	3
3	Are there distributed processing functions?	3
4	Is performance critical?	2
5	Will the system run in an existing, heavily utilized operational environment?	0
6	Does the system require on-line data entry?	3
7	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	1
8	Are the master files updated on-line?	2
9	Are the inputs, outputs, files or inquiries complex?	1
10	Is the internal processing complex?	1
11	Is the code to be designed reusable?	2
12	Are conversion and installation included in the design?	0
13	Is the system designed for multiple installations in different organizations?	0
14	Is the application designed to facilitate change and ease of use by the user?	3

Domain Characteristic Table | FP Calculation

The next step is to fill out the Domain Characteristic Table for each Use Case. Here is one example:

Domain Characteristic Table

MEASUREMENT PARAMETER	COUNT (value >= 0)	WEIGHTING FACTOR Simple	Average	Complex
Number of User Input	1	◉	○	○
Number of User Outputs	3	○	◉	○
Number of User Inquiries	1	◉	○	○
Number of Files	3	◉	○	○
Number of External Interfaces	0	◉	○	○

Complexity Adjustment Table | FP Calculation

And as a result the Function Points for this Use Case are calculated:

RESULT	
PROJECT FUNCTION POINTS	37.800000000000004

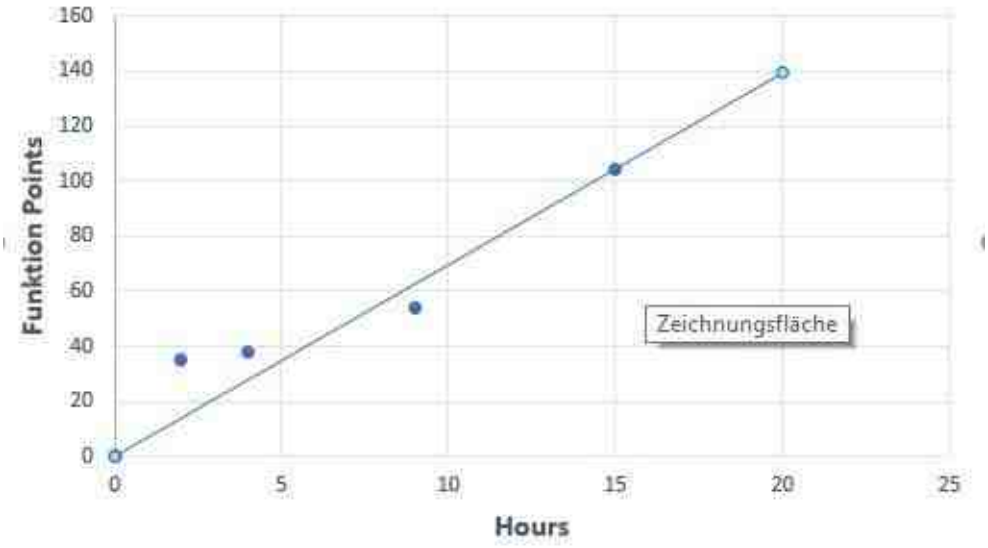
Top of Page | Domain Characteristic Table | Complexity Adjustment Table

In addition for every Measurement Parameter the FTR’s, RET’s and DET’s have to be examined. Than the Use Cases are written in an Excel Sheet:

	A	B	C	D	E	F
1	UC1 CRUD Tasks	RET	DET	FTR	Resulting Complexity	Count
2	External Inputs		9	0	Complex	9
3	External Outputs	2	2	0	Avarage	4
4	External Queries		0	3	Avarage	3
5	Internal Logical Files	3	0		Avarage	3
6	External Interface Files	0	0		low	0
7	Function Points	104.4				
8						
9	UC2 CRUD Rewards	RET	DET	FTR	Resulting Complexity	Count
10	External Inputs		5	0	average	5
11	External Outputs	1	1	0	average	2
12	External Queries		0	3	low	3
13	Internal Logical Files	3	0		low	3
14	External Interface Files	0	0		low	0
15	Function Points	54.0				
16						
17	UC3 Finish Task	RET	DET	FTR	Resulting Complexity	Count
18	External Inputs	0	1	0	low	1
19	External Outputs	2	1	0	average	3
20	External Queries	0	0	1	low	1
21	Internal Logical Files	3	0	0	low	3
22	External Interface Files	0	0	0	low	0
23	Function Points	37.8				
24						
25	UC4 Finish Reward	RET	DET	FTR	Resulting Complexity	Count
26	External Inputs		1	0	low	1
27	External Outputs	2	1	0	low	3
28	External Queries		0	1	low	1
29	Internal Logical Files	3	0		low	3
30	External Interface Files	0	0		low	0
31	Function Points	35.1				

You can find it also on our GitHub: [1]https://github.com/david2628/Work2Play/blob/main/Function _Points %20(1).xlsx

Now with the Function Points per Use Case and the time we have spent on a Use Case you can create a graph to estimate the times for future Use Cases:



For the final presentation more Use Cases will be added and so the graph is getting more accurate in the future.

Thanks for reading and feel free to leave a comment.

Your Work2Play-Team

1. https://github.com/david2628/Work2Play/blob/main/Function_Points%20(1).xlsx

2.3 May

Week 14 - Testing (2021-05-02 02:34) - davideb3

Welcome back to our blog this week,

An important part in Software is testing which we tackled this week. For our Project we have to choose three different types of testing for our project.

Last Semester we wrote feature files that will be our first type of testing.
This week we started making Unit-Tests with Mockito. The inclusion of all the classes and library's we need to write a Test is done via Gradle.
Mockito can be used to mock all the surroundings for the methode you are testing.
In our case we use it to verify that specific Methods are called and executed.

```
package com.example.work2play;

import ...

public class FragmentTasksTest {
    @Test
    void testMethodTask() {
        final FragmentTasks fragmentTasks = mock(FragmentTasks.class);
        fragmentTasks.addTask(anyString(),anyInt());
        verify(fragmentTasks).addTask(anyString(),anyInt());
    }

}
```

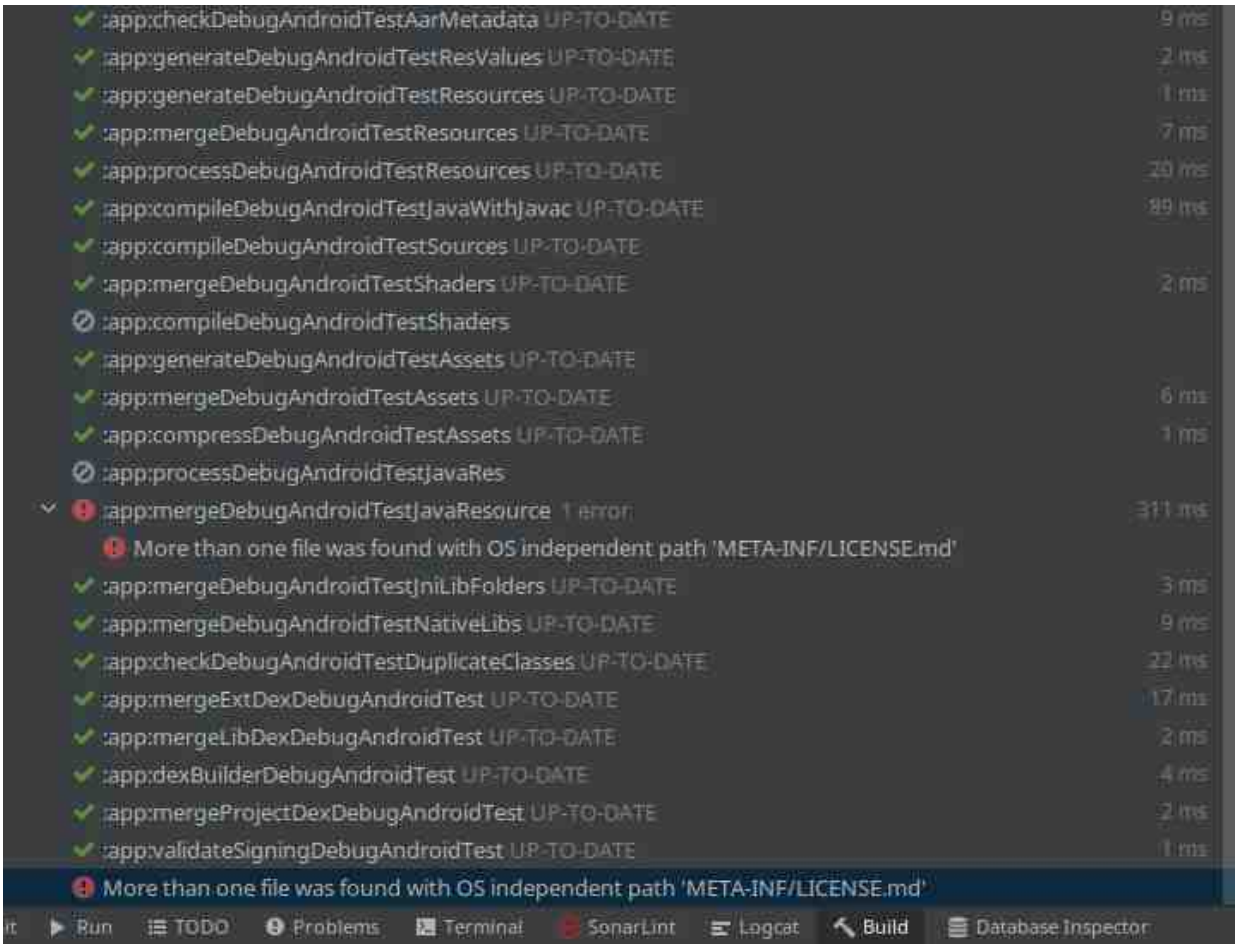
The Test now can be run by the "run test" option.

Later we will be using Installation-Tests.
In combination the three types of testing should give us a solid testing base.

The [1]Testfile is on our [2]Doku-Git.
You can find the test on our [3]GitHub.

Best regards
Team Work2Play

P.S.: for now all the tests i ran exit with a strange error



1. `https://github.com/david2628/Work2Play/blob/main/TestPlan.md`
2. `https://github.com/david2628/Work2Play/`
3. `https://github.com/rbnsch/Work2Play/tree/unittesting/app/src/androidTest/java/com/example/work2play`

Week 15 - Refactoring (2021-05-09 18:09) - rbnsch00

Hello guys and welcome back to our blog,

this week we learned how to refactor code. Every team member had to do this on their own with the help of Martin Fowler's book "[1]Refactoring":

We had to start with [2]this code.

You can view our results in the following repositories:

- [3]David
- [4]Robin
- [5]Tom

All of us used IntelliJ as IDE. IntelliJ often helps with highlighting unused variables/imports/semicolons or messy code parts. Often you can fix this by pressing alt + enter, clicking on the yellow/red light bulb or with a right click on the item. It is also possible to rename all variables at once with pressing Shift + F6. With a right click in the code you can find under "Refactor" many other options.

And that's it for this quick announcement this week. Thanks for reading and we hope to see you in near future.

Your Work2Play-Team

1. http://silab.fon.bg.ac.rs/wp-content/uploads/2016/10/Refactoring-Improving-the-Design-of-Existing-Code-Addison-Wesley-Professional-1999.pdf
2. https://github.com/gnilkreb/Fowler/tree/c0e1c7a21a5335d7e475c2c795ed77deec37b776
3. https://github.com/david2628/refactoring
4. https://github.com/rbnsch/Refactor_Task/commits/master
5. https://github.com/tomwgnr/Refactoring_DHBW

Week 16 - Design Patterns (2021-05-16 22:48) - tomwb3

Hello guys and welcome back to another blog post.

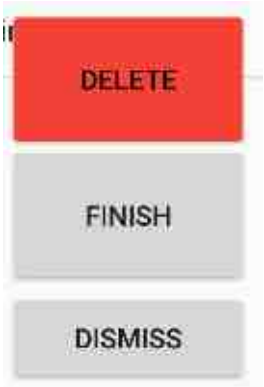
In this week we held on to the idea of last week and continued refactoring our code. This time however, we went a step further an looked into design patterns.

Our task was to implement at least one design pattern that may or may not improve the quality of our code.

In our case, we decided for a Strategy Design Pattern.
A Strategy Design is usually used when many subclasses are overwriting the same methods of the superclass. In this case, those methods can be capsulated into seperate classes in a new interface. This way the overwritten methods are all in the same file, which makes the code much clearer and the methods can be changed easier. In addition to that, the code can be reused if there are, for example, two subclasses with the same overwritten method.

A more detailed and a far more comprehensible explanation can be found [1]here.

In our case, we used this pattern to capsulate the differend popup windows shown when hold-clicked on differend items:



As always, this is still work in progress, the design is not quite ready yet!

In our code, the pattern looks like this: Instead of creating it again in every Fragment of the App, the differend popup windows get outsourced as a class in the Popup Interface:

```
interface Popup{

    public void showPopup(final int position, FragmentActivity currActivity);
}

class TaskPopup extends Fragment implements Popup{...}

class RewardPopup extends Fragment implements Popup {...}
```

In the differend Fragments, there only has to be created an Object of the popup class you want to use, for example in the "Task" fragment:

```
static Popup popup = new TaskPopup();
```

and to show the popup window:

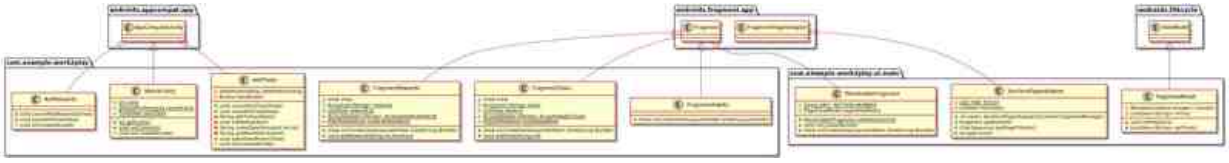
```
taskList.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {

        FragmentActivity currActivity = getActivity();
        popup.showPopup(position, currActivity);


        return true;
    }
});
```

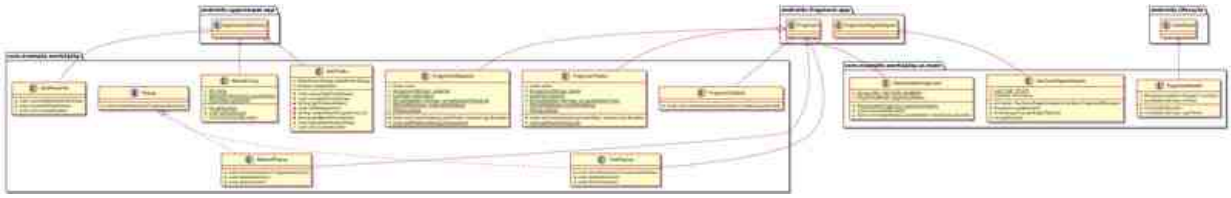
To look at our entire code, you can just follow the link [2]here:

Down below you can see our class diagram before the design pattern:



You can find the entire picture in our Git [3]here

...and after:



...and [4]here

Sorry by the way for the inconvenient format, but it is a automatically generated diagram and we haven’t found a solution yet. As long as that, you have to look in our git to have a closer look.

As you can see that there are 2 new classes for the popup windows and a popup interface.

Even though Design Patterns can help you clean up your code, we will most propably not continue to work with the introduced classic Design Patterns. The main reason for this is that the android framework we are using for this project already provides a pretty significant amout of patterns itself and has its own predefined structure. Messing with this given structure just to implement our own Design Patterns by brute force is very time consuming and prone to failure and is overall not rewarding in the long term.

But that was it for this blog post. We hope you could get some insight in our project and we are always open for new input and tipps.

Until next week,

your Work2Play-Team

1. https://www.philipphauer.de/study/se/design-pattern/strategy.php
2. https://github.com/rbnsch/Work2Play/tree/master/app/src/main
3. https://github.com/david2628/Work2Play/blob/main/ClassDiagramms/CD_Pattern_before.png
4. https://github.com/david2628/Work2Play/blob/main/ClassDiagramms/CD_Pattern_after.png

Week 17 - Metrics (2021-05-30 15:36) - tomwb3

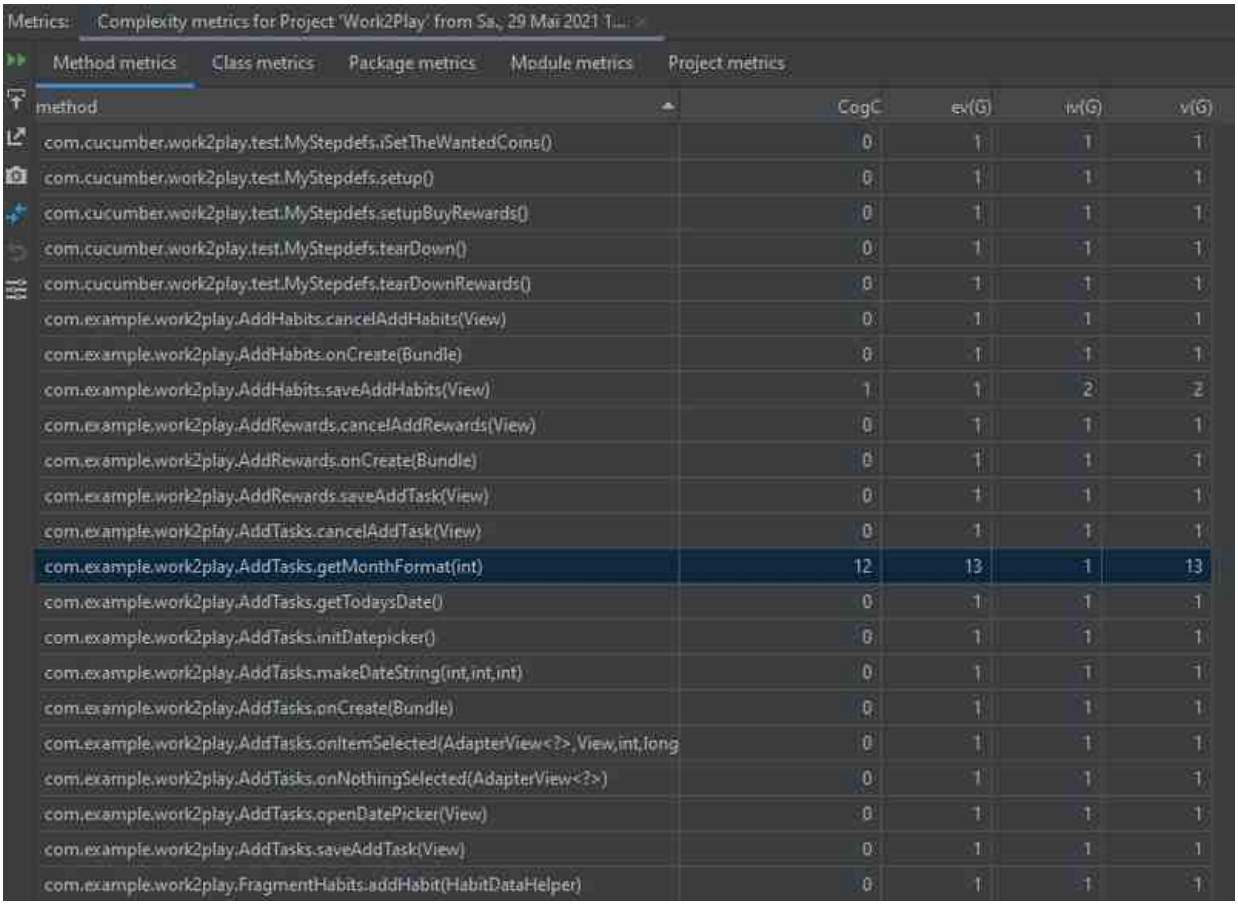
Hello and welcome to another post on this blog.

We are still on our mission of refactoring and improving our code. This week, we introduced the use of metrics in our development process.

In this post, we want to give you 2 examples of metrics we used and 2 differend ways we calculated them. We use 2 tools depending on the situation because we think both have some sort of right to exist.

The first one is pretty easy and obvious, but we decided to share it anyways because it really shows how we work with the information, our metric calculation tool provides.

In this case, we used the IntelliJ plugin [1]MetricsReloaded. This plugin is super easy to install and to use. Once you use the "Calculate Metrics" feature, a comparable window opens iside your IDE:

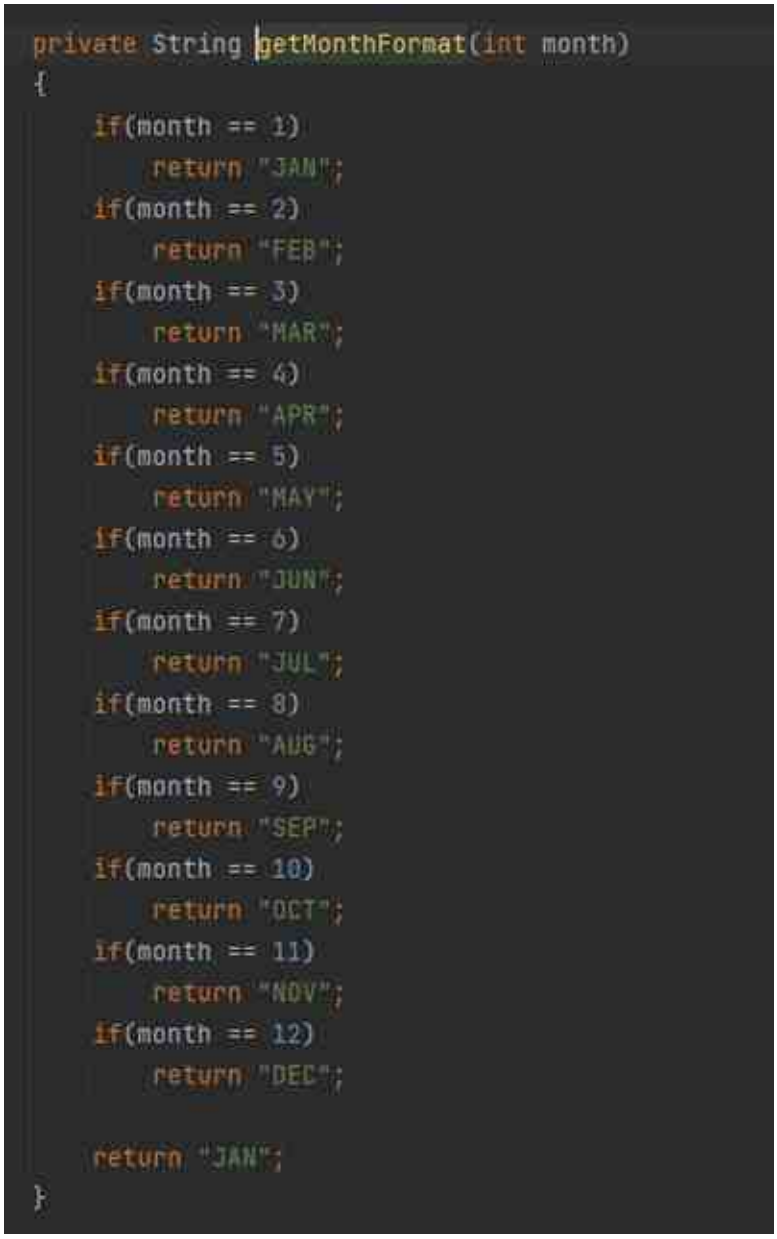


method	CogC	ev(G)	iv(G)	v(G)
com.cucumber.work2play.test.MyStepdefs.iSetTheWantedCoins()	0	1	1	1
com.cucumber.work2play.test.MyStepdefs.setup()	0	1	1	1
com.cucumber.work2play.test.MyStepdefs.setupBuyRewards()	0	1	1	1
com.cucumber.work2play.test.MyStepdefs.tearDown()	0	1	1	1
com.cucumber.work2play.test.MyStepdefs.tearDownRewards()	0	1	1	1
com.example.work2play.AddHabits.cancelAddHabits(View)	0	1	1	1
com.example.work2play.AddHabits.onCreate(Bundle)	0	1	1	1
com.example.work2play.AddHabits.saveAddHabits(View)	1	1	2	2
com.example.work2play.AddRewards.cancelAddRewards(View)	0	1	1	1
com.example.work2play.AddRewards.onCreate(Bundle)	0	1	1	1
com.example.work2play.AddRewards.saveAddTask(View)	0	1	1	1
com.example.work2play.AddTasks.cancelAddTask(View)	0	1	1	1
com.example.work2play.AddTasks.getMonthFormat(int)	12	13	1	13
com.example.work2play.AddTasks.getTodaysDate()	0	1	1	1
com.example.work2play.AddTasks.initDatepicker()	0	1	1	1
com.example.work2play.AddTasks.makeDateString(int,int,int)	0	1	1	1
com.example.work2play.AddTasks.onCreate(Bundle)	0	1	1	1
com.example.work2play.AddTasks.onItemSelected(AdapterView<?>,View,int,long	0	1	1	1
com.example.work2play.AddTasks.onNothingSelected(AdapterView<?>)	0	1	1	1
com.example.work2play.AddTasks.openDatePicker(View)	0	1	1	1
com.example.work2play.AddTasks.saveAddTask(View)	0	1	1	1
com.example.work2play.FragmentHabits.addHabit(HabitDataHelper)	0	1	1	1

Here you can switch between th differend Metrics and see which method or class needs improvement. in that case: The highlighted getMonthFormat() method. You can see that the Cognitive Complexity(CogC)- , the Cyclomatic Complexity(v(G))- and the Essentical Cyclomatic Complexity(ev(G))- Metrics are by far the highest.

What they mean is, in short: the Cyclomatic Complexity measures how difficult the code is to test. The reason for this can be for example whether the method has many branches or scenarios that need to be tested.
The Cognitive Comlexity however measures how understandable your code is for the reader.

so lets see the code:



```
private String getMonthFormat(int month)
{
    if(month == 1)
        return "JAN";
    if(month == 2)
        return "FEB";
    if(month == 3)
        return "MAR";
    if(month == 4)
        return "APR";
    if(month == 5)
        return "MAY";
    if(month == 6)
        return "JUN";
    if(month == 7)
        return "JUL";
    if(month == 8)
        return "AUG";
    if(month == 9)
        return "SEP";
    if(month == 10)
        return "OCT";
    if(month == 11)
        return "NOV";
    if(month == 12)
        return "DEC";

    return "JAN";
}
```

Terrible piece of code isn't it? Propably just copy pasted from any forum without thinking. So to reduce the Cognitive complexity, we changed it into this:

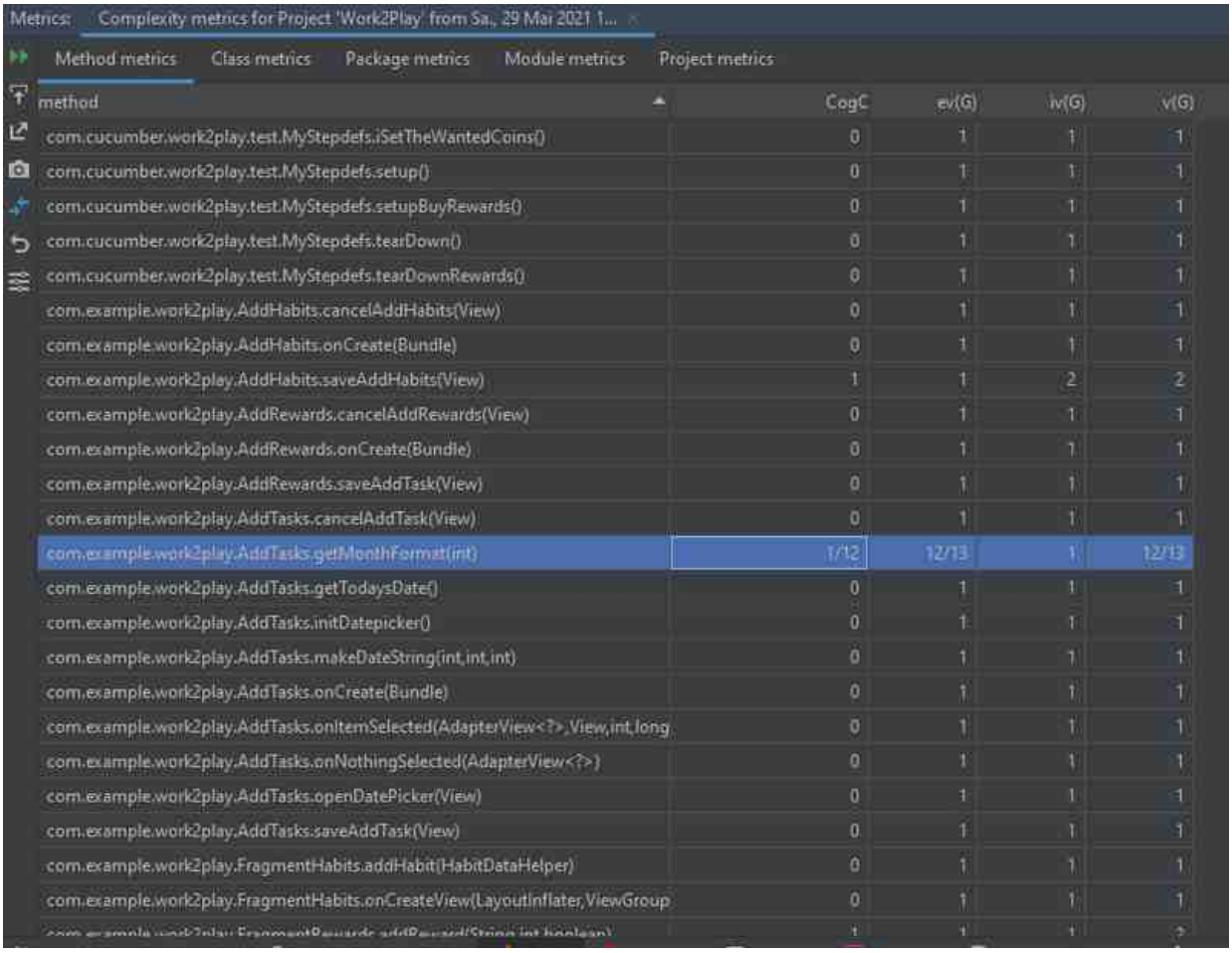
```
private String getMonthFormat(int month)
{
    switch (month){

        case 2: return "FEB";
        case 3: return "MAR";
        case 4: return "APR";
        case 5: return "MAY";
        case 6: return "JUN";
        case 7: return "JUL";
        case 8: return "AUG";
        case 9: return "SEP";
        case 10: return "OCT";
        case 11: return "NOV";
        case 12: return "DEC";

        default:return "JAN";

    }
```

A little bit better. So we looked at the newly calculated metrics and see this:

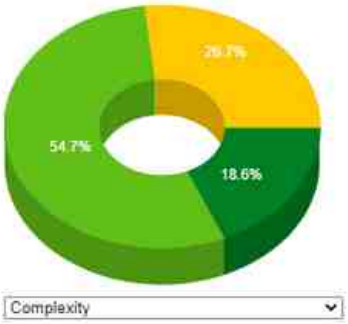


Instead of a score of 12, we only have a Cognitive Complexity of 1 left. They Cyclomatic Complexities however are still high, but we don't care, because there are 12 months in a year, which means 12 differend possible outcomes. So we just leave the code as it is. To see this change in the code directly, the link to this commit is right [2]here.

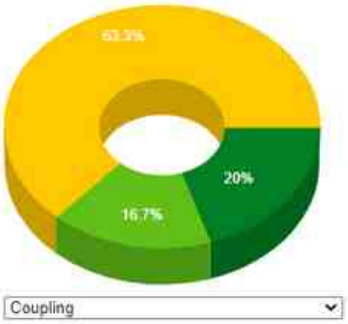
For the second Metric we looked at, we used the [3]CodeMR Plugin. When started, it opens a Dashboard where it displays the metrics of your selected code in colourful diagrams:

Distribution of Quality Attributes

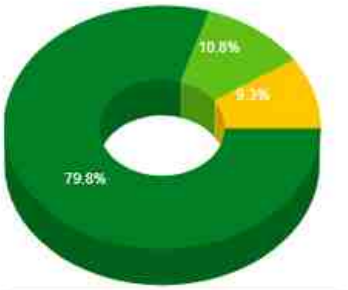
Complexity, Coupling, Cohesion, and Size



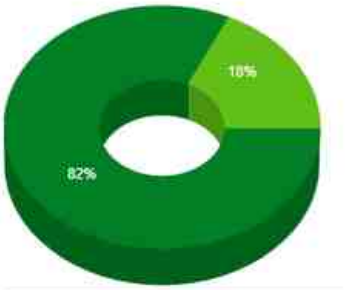
Complexity



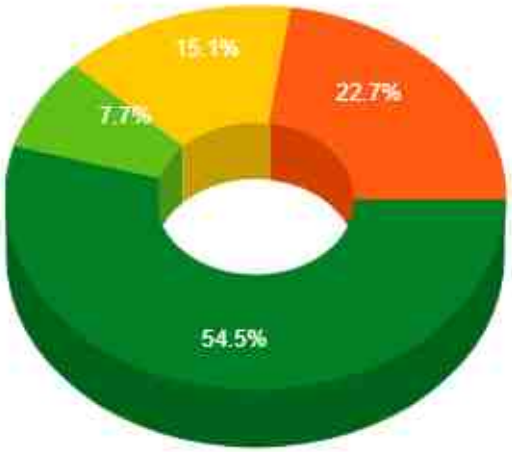
Coupling



Lack of Cohesion

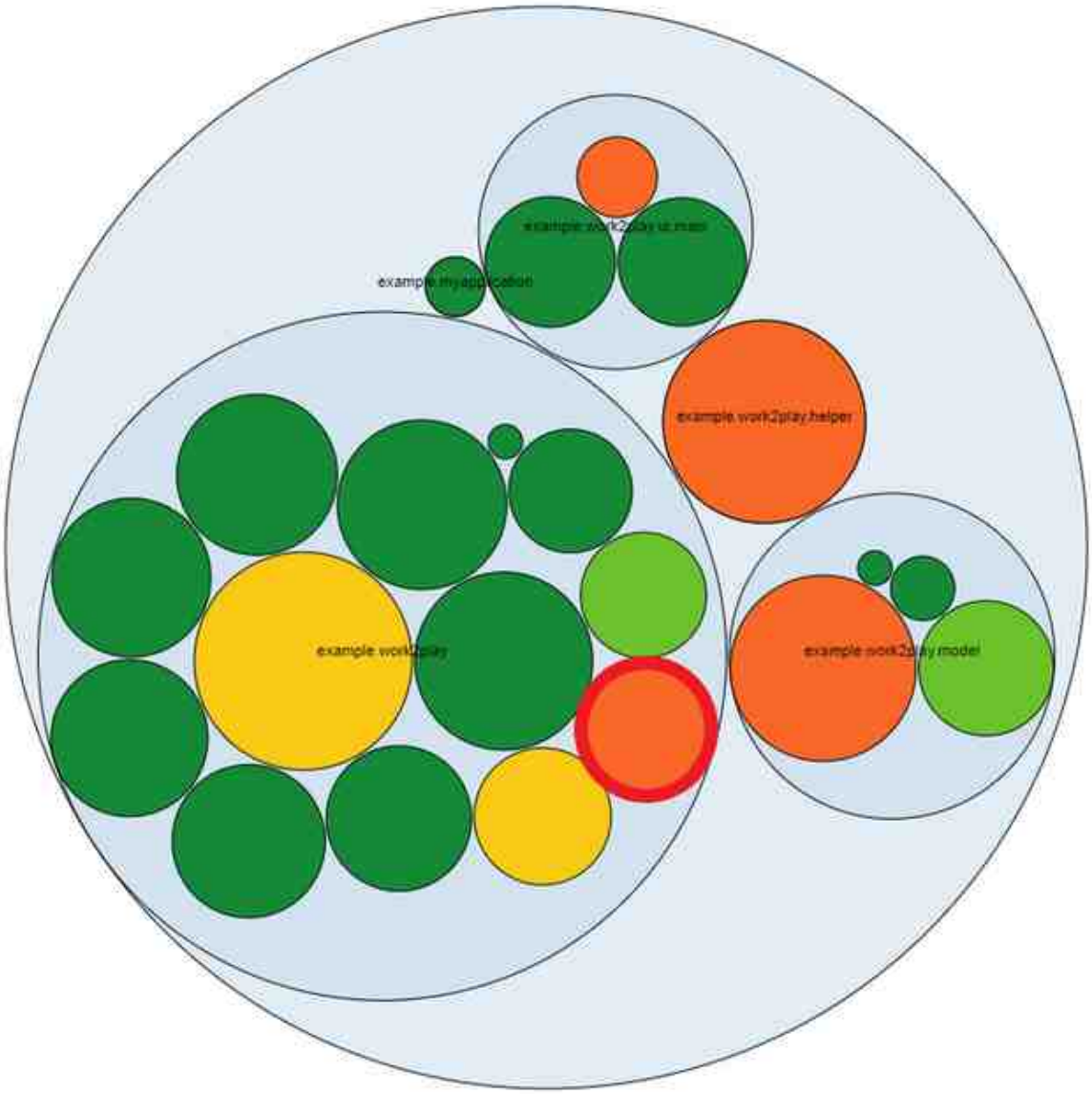


Class-Methods Lines of Code



Lack of Cohesion of Methods

As you can see, you can get a quick overview on what Metrics need improvement and which ones are fine with the classical red means high, green means low colourscheme. We decided for the "Lack of Cohesion of Methods"- Metric (LCOM) and looked a bit deeper into it:



Here you can see all the Classes in our Project and their LCOM value. The one we marked in red is our FragmentHabit class, which has to be improved the most.

LCOM measures a class which should or should not be split apart, because there are too many dissimilar methods in it, so methods, who don’t share the attributes the touch with any other method in this class. In our example:

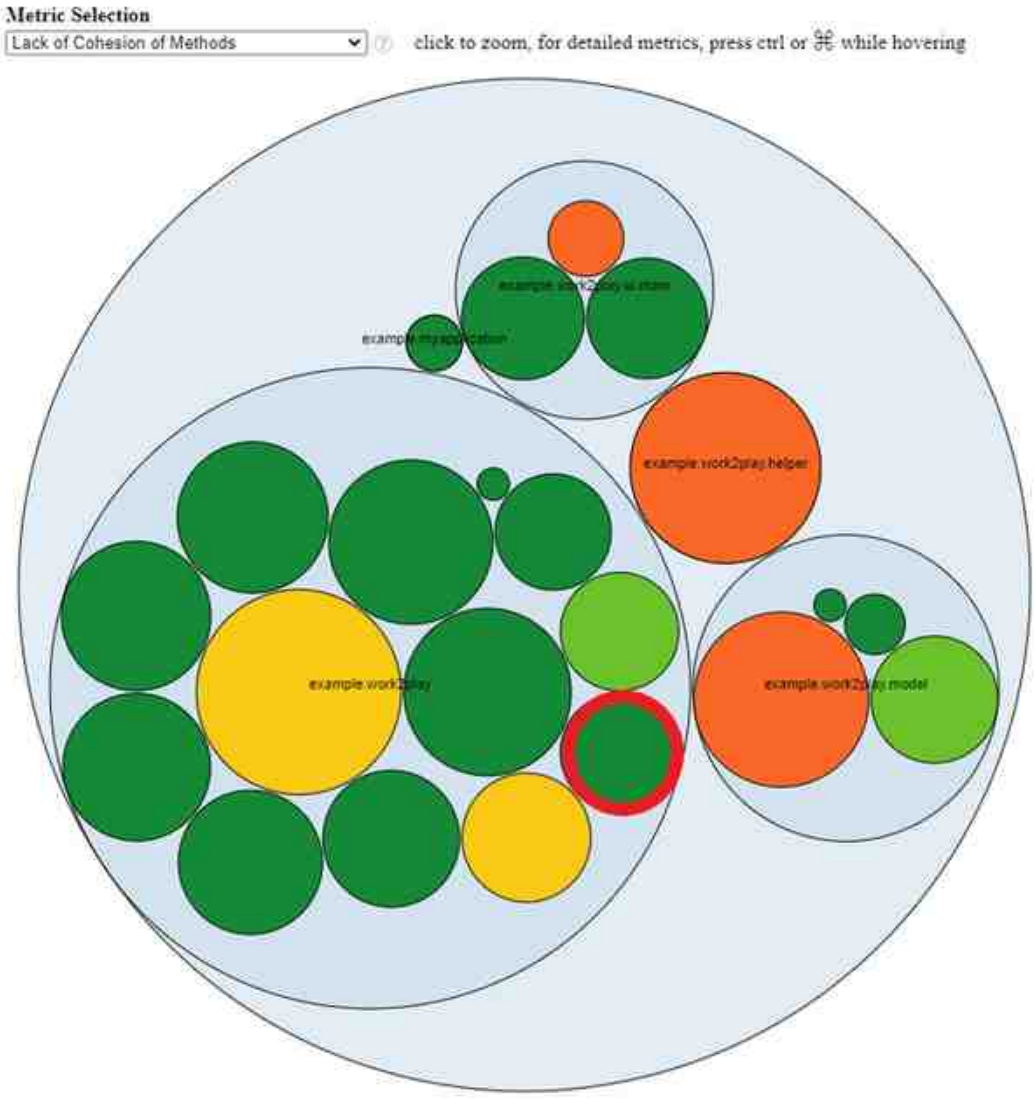
```
public static void addHabit(HabitDataHelper newHabit){
    habits.add(newHabit);
    habitList.setAdapter(habitListAdapter);
}
```

This is a method of the FragmentHabit class, but no other method touches this HabitData-Helper Attribute. So we moved it into anouther class and referenced it from there.

```
private void addHabit(HabitDataHelper newHabit) {
    FragmentHabits.habits.add(newHabit);
    FragmentHabits.habitList.setAdapter(FragmentHabits.habitListAdapter);
}
```

I think to comprehend this change a bit more, you have to see the entire code, so you can find this commit [4]here.

But anyway, after another calculation, you can see that the LCOM Metric got better for this class:



Unfortunately, both these tools can’t be integrated into our Pipeline, so we need to use Codacity in addition, just for this purpose. [5]Here is the link to it. If you have any Ideas how we could integrate them, feel free to leave them in the comments.

But thats it for this weeks blog post! Sorry it got that long, we promise the next one will be shorter again.

Thats why we to thank you extra this time for reading and we'll see us next week.

Sincerely,
the Work2Play - Team

1. https://plugins.jetbrains.com/plugin/93-metricsreloaded
2. https://github.com/rbnsch/Work2Play/commit/8af00f65d2e7678317d96a43a319cf63528cf229#diff-03fa6ccd30f47cfcce8caa66ec10f1fcea1eefaf76bf8568b10f2b0dcf7fcb72
3. https://plugins.jetbrains.com/plugin/10811-codemr
4. https://github.com/rbnsch/Work2Play/commit/3595277b42df9e20f69852ee4821afd721d6d94d#diff-03fa6ccd30f47cfcce8caa66ec10f1fcea1eefaf76bf8568b10f2b0dcf7fcb72
5. https://app.codacy.com/gh/rbnsch/Work2Play/dashboard?branch=master

2.4 June

Week 18 - Deployment (2021-06-27 17:46) - tomwb3

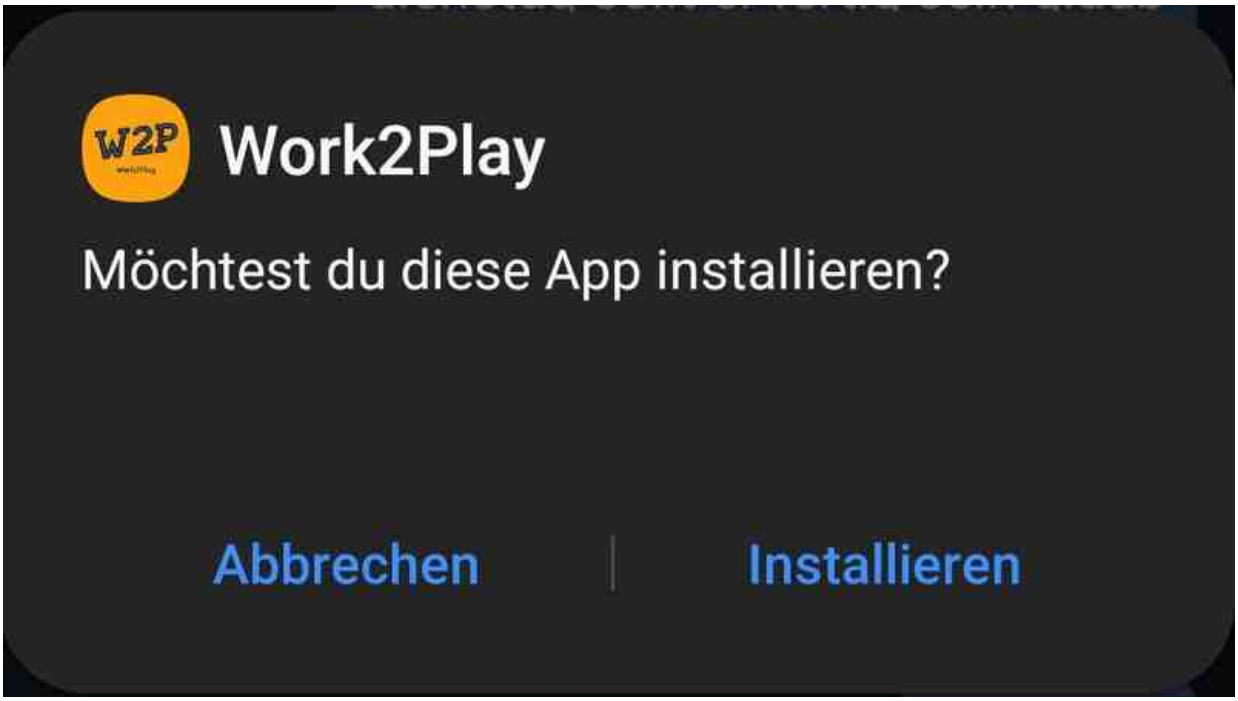
Hello guys and welcome to our (propably) second to last post on this blog.

This time, we want to talk about the deployment of Work2Play. Due to the fact that it is a plain and simple Android Application, this will be a short one.

Step 1: Download the .apk file of the app on your Android device, you can find it on our Git [1]here.

Step 2: Open it with your Android package manager

Step 3: Install the app by clicking on "Installieren" (sorry my smartphone is in german) or rather "Install":



Step 4: Click on this exact Icon:

And thats it for this post already, as said before: quick deployment - short post. The app isn’t available in the PlayStore yet, but maybe in the future!

Thank you for reading and we’ll hopefully see you on our next and final blog post.

Sincerely,
your Work2Play - Team

1. https://github.com/david2628/Work2Play/blob/main/W2P.apk

Final Post (2021-06-28 22:14) - rbnsch00

Hello and welcome to our last post on this blog.

The Semester is over and here are our results:

Github Repository

- [1]Code GitHub
- [2]Documentation GitHub
- [3]Code GitHub Zip
- [4]Documentation GitHub Zip

Youtrack

- [5]Dashboard
- [6]Agile Board
- [7]Burndown #1
- [8]Burndown #2
- [9]Cumulative Chart
- [10]Time per UseCase
- [11]Time per Workflow
- [12]Time per User
- [13]Issues list
- [14]Gant Chart

Use cases

- [15]OUCD
- [16]CRUD Habit
- [17]CRUD Reward
- [18]CRUD Task
- [19]Buy Rewards
- [20]Finish Habit
- [21]Finish Task

Software Specifications

- [22]SRS
- [23]SAD
- [24]TestPlan

Testing

- [25]Cucumber Test Run Video

- [26]Feature Files
- Test Coverage/Unit Test run (Davids job)

Pipeline

- David dein Job

[27]Risk management

[28]Function points (And in the Use Cases)

Refactoring homework

- [29]David
- [30]Robin
- [31]Tom

[32]Design Patterns (Blog post)

[33]Metrics (Blog Post)

Deployment

- [34]Download APK
- [35]Demo Video

Final

- [36]Handout
- [37]Presentation
- PDF Blog Posts

Blog posts

- [38]WEEK 1 – OUR VISION

- [39]WEEK 2 – ORGANIZING AND SETUP
- [40]WEEK 3 – SOFTWARE REQUIREMENTS SPECIFICATION
- [41]WEEK 4 – FIRST USE-CASES
- [42]WEEK 5 – FEATURE FILES
- [43]WEEK 6 – SCRUM
- [44]WEEK 7 – CLASS DIAGRAMM
- [45]WEEK 8 – SOFTWARE ARCHITECTURE DOCUMENT
- [46]WEEK 9 – RETROSPECTIVE
- [47]WEEK 10 – MIDTERM-RUNDOWN
- [48]WEEK 11 – WELCOME BACK
- [49]WEEK 12 – RISK MANAGEMENT
- [50]WEEK 13 – FUNCTION POINTS
- [51]WEEK 14 – TESTING
- [52]WEEK 15 – REFACTORING
- [53]WEEK 16 – DESIGN PATTERNS
- [54]WEEK 17 – METRICS
- [55]WEEK 18 – DEPLOYMENT

Our semester is over now and so this was the last blog post for now. We had a lot of fun doing this project and learning all the new things.

Thanks for following

Your Work2Play-Team

1. https://github.com/rbnsch/Work2Play
2. https://github.com/david2628/Work2Play
3. https://github.com/david2628/Work2Play/blob/main/Work2Play-Code.zip
4. https://github.com/david2628/Work2Play/blob/main/Work2Play-Doku.zip
5. https://dhbw-karlsruhe.myjetbrains.com/youtrack/dashboard?id=186e6d48-444e-4830-af90-6e1e9a86bfd8
6. https://dhbw-karlsruhe.myjetbrains.com/youtrack/agiles/108-62/109-639
7. https://dhbw-karlsruhe.myjetbrains.com/youtrack/reports/burndown/147-111

8. https://dhbw-karlsruhe.myjetbrains.com/youtrack/reports/burndown/147-112
9. https://dhbw-karlsruhe.myjetbrains.com/youtrack/reports/cumulativeFlow/143-36
10. https://dhbw-karlsruhe.myjetbrains.com/youtrack/reports/issueDistribution/146-35?line=user
11. https://dhbw-karlsruhe.myjetbrains.com/youtrack/reports/time/141-105?line=issue
12. https://dhbw-karlsruhe.myjetbrains.com/youtrack/reports/time/141-86?line=user
13. https://dhbw-karlsruhe.myjetbrains.com/youtrack/issues/W2PT
14. https://dhbw-karlsruhe.myjetbrains.com/youtrack/reports/gantt/148-41?view=actual
15. https://github.com/david2628/Work2Play/blob/main/SAD/PNGs/UCD.png
16. https://github.com/david2628/Work2Play/blob/main/UseCases/CRUD-Habit.md
17. https://github.com/david2628/Work2Play/blob/main/UseCases/CRUD-Reward.md
18. https://github.com/david2628/Work2Play/blob/main/UseCases/CRUD-Task.md
19. https://github.com/david2628/Work2Play/blob/main/UseCases/UC-Buy_Rewards.md
20. https://github.com/david2628/Work2Play/blob/main/UseCases/UC-Finish_Habit.md
21. https://github.com/david2628/Work2Play/blob/main/UseCases/UC-Finish_Task.md
22. https://github.com/david2628/Work2Play/blob/main/SRS.md
23. https://github.com/david2628/Work2Play/blob/main/SAD/SAD.md
24. https://github.com/david2628/Work2Play/blob/main/TestPlan.md
25. https://github.com/david2628/Work2Play/blob/main/cucumber_testing_video.mkv
26. https://github.com/rbnsch/Work2Play/tree/master/app/src/androidTest/assets/features
27. https://github.com/david2628/Work2Play/blob/main/risk_management.xlsx
28. https://github.com/david2628/Work2Play/blob/main/Function_Points.xlsx
29. https://github.com/david2628/refactoring
30. https://github.com/rbnsch/Refactor_Task/commits/master
31. https://github.com/tomwgnr/Refactoring_DHBW
32. https://work2playtogether.wordpress.com/2021/05/16/week-16-design-patterns/
33. https://work2playtogether.wordpress.com/2021/05/30/week-17-metrics/
34. https://github.com/david2628/Work2Play/blob/main/W2P.apk
35. https://github.com/david2628/Work2Play/blob/main/demo_video.m4v
36. https://github.com/david2628/Work2Play/blob/main/Final/W2P_final_handout.pdf
37. https://github.com/david2628/Work2Play/blob/main/Final/W2P_final.pptx
38. https://work2playtogether.wordpress.com/2020/10/01/example-post/
39. https://work2playtogether.wordpress.com/2020/10/08/week-2-organizing-and-setup/
40. https://work2playtogether.wordpress.com/2020/10/19/week-3-software-requirements-specification/
41. https://work2playtogether.wordpress.com/2020/10/26/week-4-first-use-cases/
42. https://work2playtogether.wordpress.com/2020/11/02/week-5-feature-files/
43. https://work2playtogether.wordpress.com/2020/11/09/week-6-scrum/
44. https://work2playtogether.wordpress.com/2020/11/24/week-7-class-diagramm/
45. https://work2playtogether.wordpress.com/2020/11/30/week-8-software-architecture-document/
46. https://work2playtogether.wordpress.com/2020/12/01/week-x-retrospective/
47. https://work2playtogether.wordpress.com/2021/03/18/midterm-rundown/
48. https://work2playtogether.wordpress.com/2021/04/10/week-11-welcome-back/
49. https://work2playtogether.wordpress.com/2021/04/11/week-12-risk-management/
50. https://work2playtogether.wordpress.com/2021/04/18/week-13-function-points/
51. https://work2playtogether.wordpress.com/2021/05/02/week-14-testing/
52. https://work2playtogether.wordpress.com/2021/05/09/week-14-refactoring/
53. https://work2playtogether.wordpress.com/2021/05/16/week-16-design-patterns/
54. https://work2playtogether.wordpress.com/2021/05/30/week-17-metrics/
55. https://work2playtogether.wordpress.com/2021/06/27/week-x-deployment/

BlogBook v1.2,
LaTeX 2$_\varepsilon$ & GNU/Linux.
https://www.blogbooker.com

Edited: June 29, 2021