strebo

social trend board

strebo – social trend board ❤

Posted on 29. September 20155. October 2015 by streboschreck

Have you ever wished of a web-based application where you can see ALL the trendy stuff and all your relevant content from different social-media platforms at a glance?

Your journey will end here!

strebo is the project that we will work on, while we are in our third and fourth semester at the Cooperative State University in Karlsruhe. This blog will be used to publish news and other stuff about the project.

The following features are thinkable options for strebo:

Core features:

- Showing trending content (text, images, videos, …) from social media platforms like Twitter, Facebook, Google+, Instagram, YouTube, …
- Connecting your social media accounts with strebo and show your personal relevant content from social media platforms
- Searching for content across several social media platforms

Additional features:

- Diagrams about trends, how they grow/how they fall
- World map that shows live posting activities
- Showing trends in single countries/regions or trends depending on other factors
- Mobile solution
- …

Enjoy!

4 □

Final

Posted on 14. June 201614. June 2016 by streboretkowski

Hey guys,

Have you ever wished of a web-based application where you can see ALL the trendy stuff and all your relevant content from different social-media platforms at a glance?

We are here – the semester is over and here are our results of the project which you will find in a nice overview at the following link:

https://github.com/strebo/strebo/wiki

Best regards,
team strebo

0

Installation

Posted on 9. June 201610. June 2016 by streboretkowski

Hi there,

today we want to show you how you can easily set up our project on your own machine.
We wanted to make it as easy as possible for you, so we use **Docker** now and created a **dockerfile** .

You can find the dockerfile in our repository on GitHub (named as dockerfile in our root directory):
https://github.com/strebo/strebo

Clone it or download it to your own machine. Please make sure you have Docker installed.

Build the docker image with (make sure you are in the same repository as our dockerfile) :

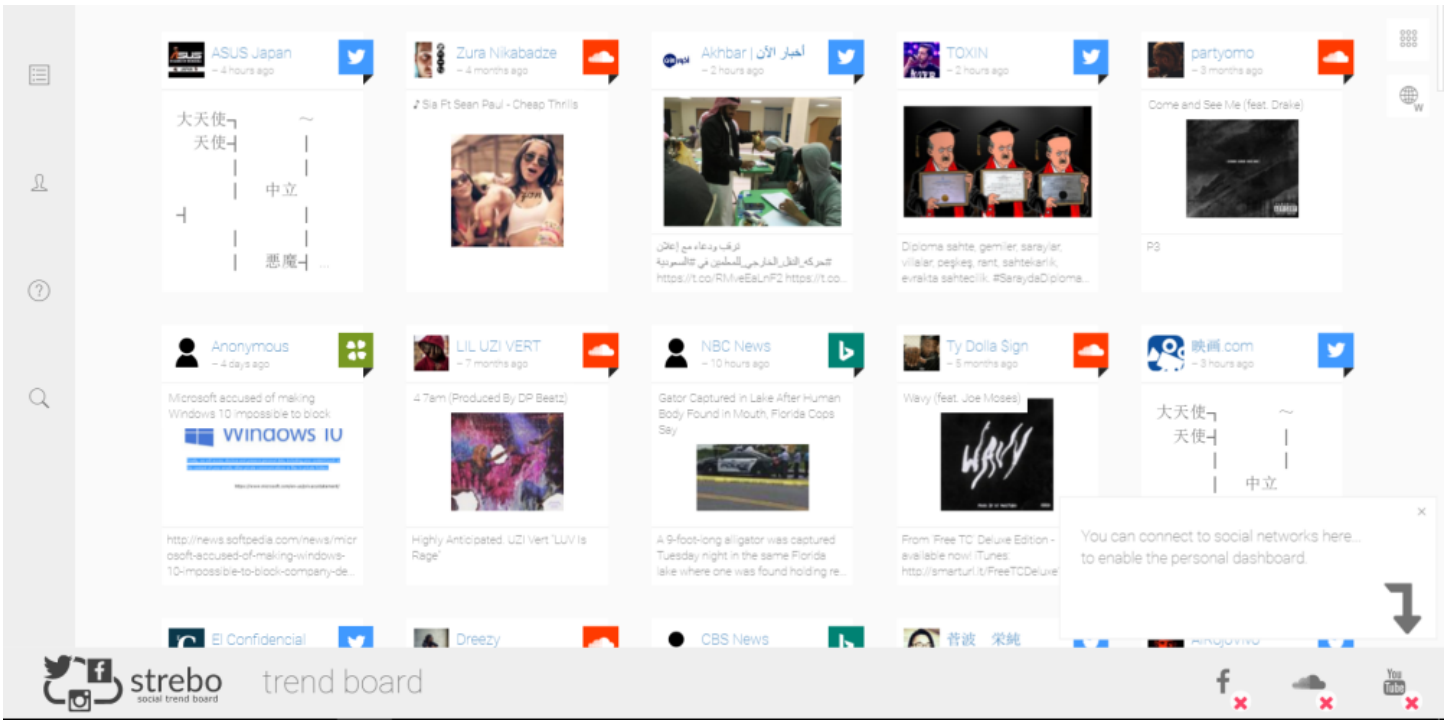
docker build -t strebo .

Then you have just to run it:

docker run -it -p 80:80 -p 8080:8080 -p 443:443

You will get visual feedback if it does work – and the only thing you have to do now is to open your browser and type in: **http://localhost**

Et voilà:



Hint: The personal board for what logins are required *WON'T* work. For this you would have to need to create your own accounts and applications on several social networks. Since in our applications is set that only strebo.net is a valid request and redirect URI.

Best regards,
team strebo

0

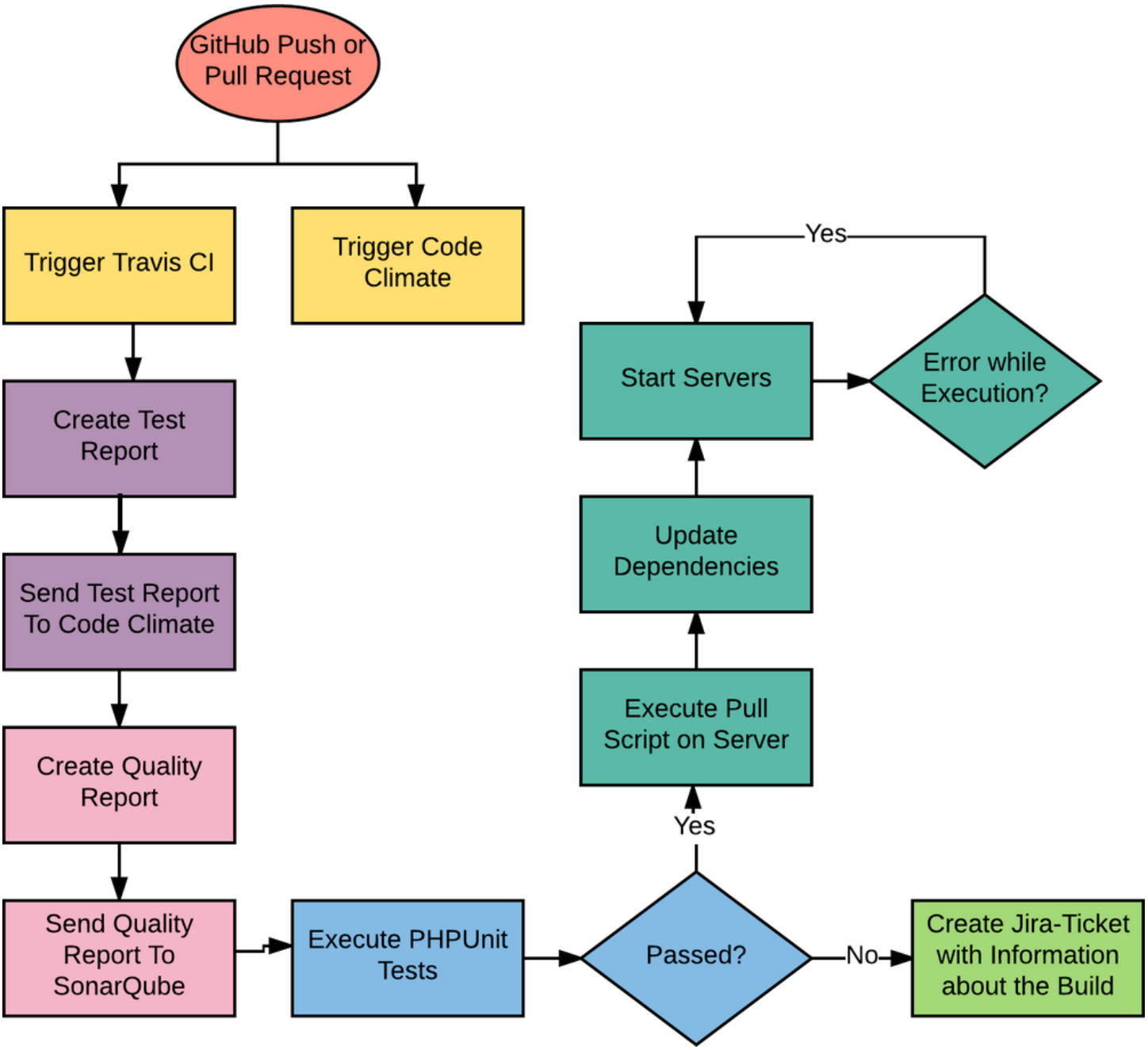
Auto Depolyment

Posted on 3. June 20163. June 2016 by streboretkowski

Hi guys,

we want to show you how we deal with auto deployment.

You will get a rough overview about our build and deployment process (it's the same since we use PHP) in the following flow chart:

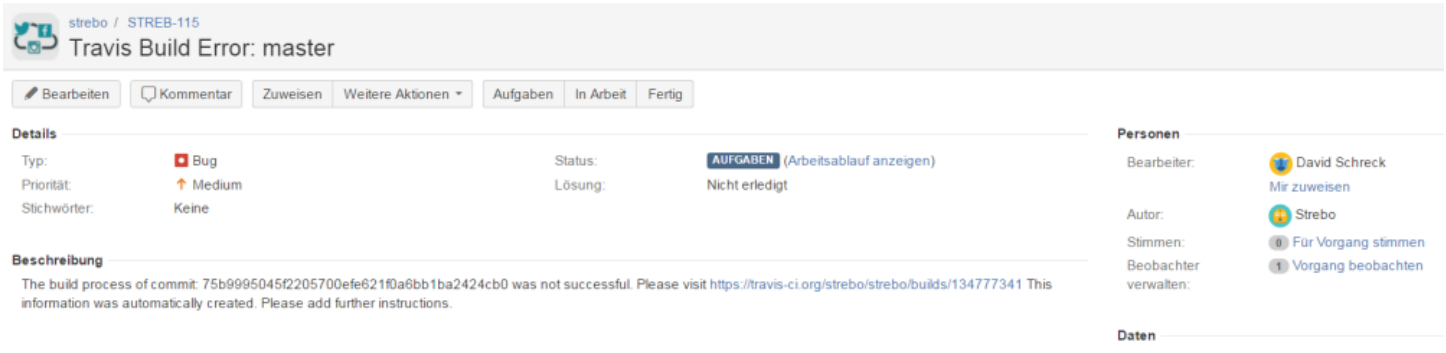


It all begins with a GitHub Push / Pull Request which triggers Travis CI to execute the build and deployment process and Code Climate to create a Quality Report. This happens via Webhooks. Then Travis CI creates a Test Report and send it to Code Climate, so it will be informed about our Code Coverage. For SonarQube there will be created a Quality Report. After that our PHPUnit tests will be executed. Depending on the success there is a jira connection which creates a ticket (in case of a fail) or the deployment process on the server will be started. The server pulls from GitHub, updates the dependencies and then restart the servers. If an error occures while execution, the server restarts automatically after 10 seconds. So strebo.net is nearly always available.

travis.yml: https://github.com/strebo/strebo/blob/master/.travis.yml

Start Scripts to make auto deployment possible:
https://github.com/strebo/strebo/blob/master/start.sh
https://github.com/strebo/strebo/blob/master/serverstart.sh
https://github.com/strebo/strebo/blob/master/nserverstart.sh

Here you see what happens if a build process fails. There will be automatically created a jira ticket:



Bash Script which informs Jira:
https://github.com/strebo/strebo/blob/master/.send_ticket.sh

We integrated several badges in GitHub:

Link: https://github.com/strebo/strebo

In addition we updated our Test Document:
https://github.com/strebo/strebo/wiki/Test-Document

Best regards,
strebo team

4

Metrics

Posted on 22. May 20162. June 2016 by streboretkowski

Hi all,

we use Code Climate (it's also a Code Coverage Tool) and SonarQube as Metric Tools. Both are integrated in our deployment process by using Travis CI.

See our Travis Configuration File:
https://github.com/strebo/strebo/blob/master/.travis.yml

SonarQube: http://sonarqube.it.dh-karlsruhe.de/overview?id=5357

Code Climate: https://codeclimate.com/github/strebo/strebo

We analyzed our application regarding complexity. Complexity means the cylocmatic complexity which is defined as:

It is a quantitative measure of the number of linearly independent paths through a program's source code.

So we refactored **/Strebo/Twitter.php** because of Code Climate suggestion:

The method formatTime() has a Cyclomatic Complexity of 13. The configured cyclomatic complexity threshold is 10.

```
153   public function formatTime($time)
154   {
155
156       $month = 0;
157
158       //Timestamp oder Array
159       switch (substr($time, 4, 3)) {
160           case 'Jan':
161               $month = 1;
162               break;
163           case 'Feb':
164               $month = 2;
165               break;
166           case 'Mar':
167               $month = 3;
168               break;
169           case 'Apr':
170               $month = 4;
171               break;
172           case 'May':
173               $month = 5;
174               break;
175           case 'Jun':
176               $month = 6;
177               break;
178           case 'Jul':
179               $month = 7;
180               break;
181           case 'Aug':
182               $month = 8;
183               break;
184           case 'Sep':
185               $month = 9;
186               break;
187           case 'Oct':
188               $month = 10;
189               break;
190           case 'Nov':
191               $month = 11;
192               break;
193           case 'Dec':
194               $month = 12;
195               break;
196       }
197
198       $timeJSON = array('day' => substr($time, 8, 2),
199           'month' => $month,
200           'year' => substr($time, 26),
201           'hour' => substr($time, 11, 2),
202           'minute' => substr($time, 14, 2),
203           'second' => substr($time, 17, 2)
204       );
205
206       return json_encode($timeJSON);
```

This switch can be replaced with a simple associative array:

```
172     public function formatTime($time)
173     {
174         $month = ["Jan" => 1,
175             "Feb" => 2,
176             "Mar" => 3,
177             "Apr" => 4,
178             "May" => 5,
179             "Jun" => 6,
180             "Jul" => 7,
181             "Aug" => 8,
182             "Sep" => 9,
183             "Oct" => 10,
184             "Nov" => 11,
185             "Dec" => 12];
186
    Similar code found in 1 other location (mass = 29)
188         $timeJSON = array('day' => substr($time, 8, 2),
189             'month' => $month[substr($time, 4, 3)],
190             'year' => substr($time, 26),
191             'hour' => substr($time, 11, 2),
192             'minute' => substr($time, 14, 2),
193             'second' => substr($time, 17, 2)
194         );
195
196         return json_encode($timeJSON);
197
198     }
199 }
```

So the Cyclomatic Complexity issue disappear. Instead of going through all alternatives with O(n), it's now O(1), thanks to array access. Also – the source code has much less lines of codes what improves the clarity of our code.

We also analyzed our application regarding duplicated code. Dublicated code is defined by the following subsections:

Identical code

When 2 or more blocks of code contain the exact same variable names and structure.

Similar code

When 2 or more blocks of code contain the same structure, but have different contents (such as variable names or literal values). This can help catch cases where a developer has copy and pasted a section of code, leaving the structure the same, but adjusting some variable names for a different context.

Mass

"Mass" refers to the size of the duplicated code. Specifically, mass is determined by the size of a code block's s-expression, after it has been parsed into a node of an Abstract Syntax Tree (AST).

Here you can see Code Climates suggestion:

```
184     public function formatTime($time)
185     {
186         $month = ["Jan" => 1,
187             "Feb" => 2,
188             "Mar" => 3,
189             "Apr" => 4,
190             "May" => 5,
191             "Jun" => 6,
192             "Jul" => 7,
193             "Aug" => 8,
194             "Sep" => 9,
195             "Oct" => 10,
196             "Nov" => 11,
197             "Dec" => 12];
198
        Similar code found in 1 other location (mass = 29)
199         $timeJSON = array('day' => substr($time, 8, 2),
200             'month' => $month[substr($time, 4, 3)],
201             'year' => substr($time, 26),
202             'hour' => substr($time, 11, 2),
203             'minute' => substr($time, 14, 2),
204             'second' => substr($time, 17, 2)
205         );
206
207         return json_encode($timeJSON);
208
209     }
210 }
```

We solved this problem by integrating the function formatTime($time) into the abstract class AbstractSocialNetwork. Each other social setwork class extends this class and is so able to use this function without the need to reimplement it:

```
$data['createdTime'] = parent::formatTime(strtotime($tweet->created_at));
$data['text'] = $tweet->text;
$data['title'] = null;
```

The resulting code of class Twitter and the other social network classes is much shorter now. It's also much easier now to change the logic of how the time is formatted because just a single class has to be modified.

The source code of this file can be found on GitHub: https://github.com/strebo/strebo/blob/master/Strebo/SocialNetworks/Twitter.php

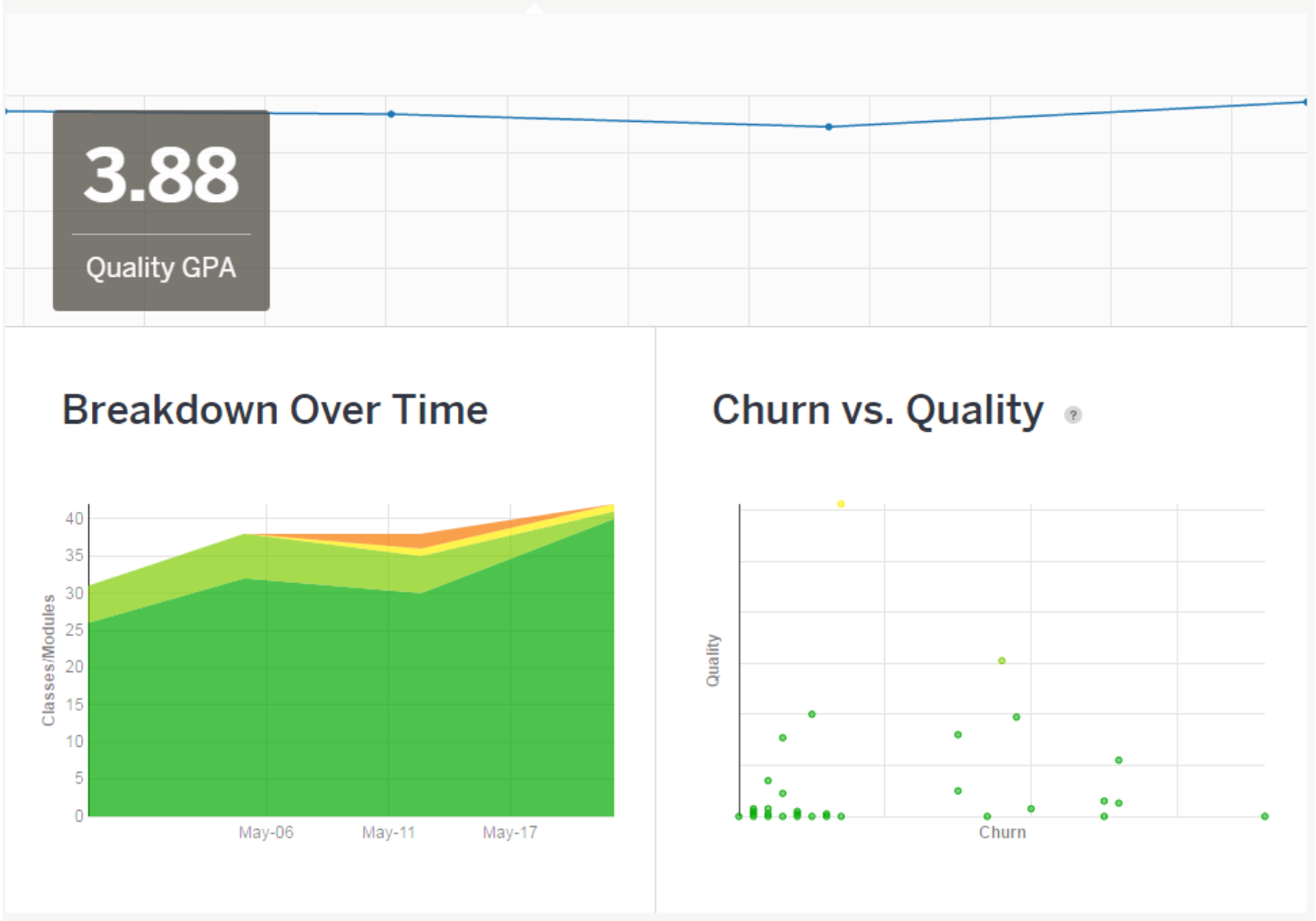
We decided to let the following issue unresolved. The reason is that it's not that obvious to refactor this code without making a mistake and we think it's okay since this code can be seen as completed. We don't expect it to change or to get more complex. It's an isolated piece of code.

js/DifferenceFilter.js

```
app.filter('differenceFilter', function () {
```
Function 'anonymous' has a complexity of 13.
```
        return function (time) {
            var dateGet = new Date();
            if (time) {
                time = JSON.parse(time);
                var postDate = new Date(time.year, time.month-1, time.day, time.hour, time.minute, time.second);
                var timeDiffSec = Math.round(Math.abs(dateGet - postDate) / 1000);
                var timeDiffMin = Math.round(Math.abs(timeDiffSec) / 60);
                var timeDiffHour = Math.round(Math.abs(timeDiffMin) / 60);
                var timeDiffDay = Math.round(Math.abs(timeDiffHour) / 24);
                var timeDiffMonth = Math.round(Math.abs(timeDiffDay) / 30);
                var timeDiffYear = Math.round(Math.abs(timeDiffMonth) / 12);
                var formattedTime;
                if (timeDiffSec < 60) {
                    formattedTime = "just now";
                } else if (timeDiffMin < 60) {
                    formattedTime = timeDiffMin + ( timeDiffMin===1 ? " minute " : " minutes " ) + "ago";
                } else if (timeDiffHour < 24) {
                    formattedTime = timeDiffHour + ( timeDiffHour===1 ? " hour " : " hours " ) + "ago";
                } else if (timeDiffDay < 30) {
                    formattedTime = timeDiffDay + ( timeDiffDay===1 ? " day " : " days " ) + "ago";
                } else if (timeDiffMonth < 12) {
                    formattedTime = timeDiffMonth + ( timeDiffMonth===1 ? " month " : " months " ) + "ago";
                } else if (timeDiffYear !== 0) {
                    formattedTime = timeDiffYear + ( timeDiffYear===1 ? " year " : " years " ) + "ago";
                } else {
                    formattedTime = '';
                }
                return formattedTime;
```
Unexpected 'else' after 'return'.
```
            } else {
                return "timeless";
            }
        };
    }
)
;
```

Source code can be found here: https://github.com/strebo/strebo/blob/master/js/DifferenceFilter.js

The overall improvment of our code is shown in the following charts:



Best regards,
team strebo

4

Test Coverage

Posted on 22. May 201623. May 2016 by streboretkowski

Hi all,

we use Code Climate as our test coverage tool. We have written some tests and we have now a test coverage in the desired area of 20 – 40 percent.

Detail information about our test coverage you can find here: https://codeclimate.com/github/strebo/strebo/coverage

It also offers a badge: coverage 32%

As an additional tool we use SonarQube to improve the Code Quality: http://sonarqube.it.dh-karlsruhe.de/overview?id=5357

Our test document can be found here: https://github.com/strebo/strebo/wiki/Test-Document

Best regards,
team strebo

4

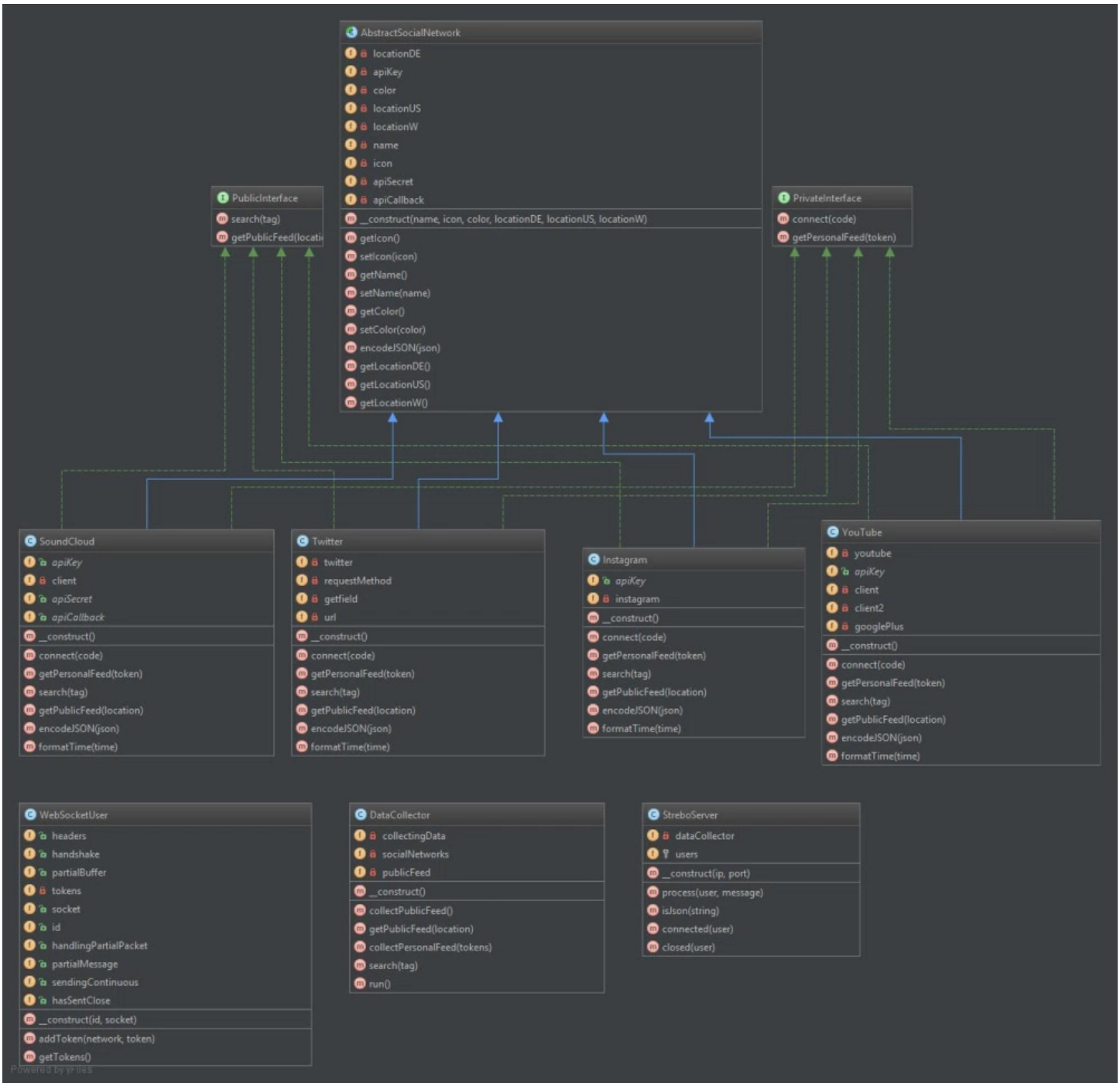
Design Pattern

Posted on 8. May 20169. May 2016 by streboschreck

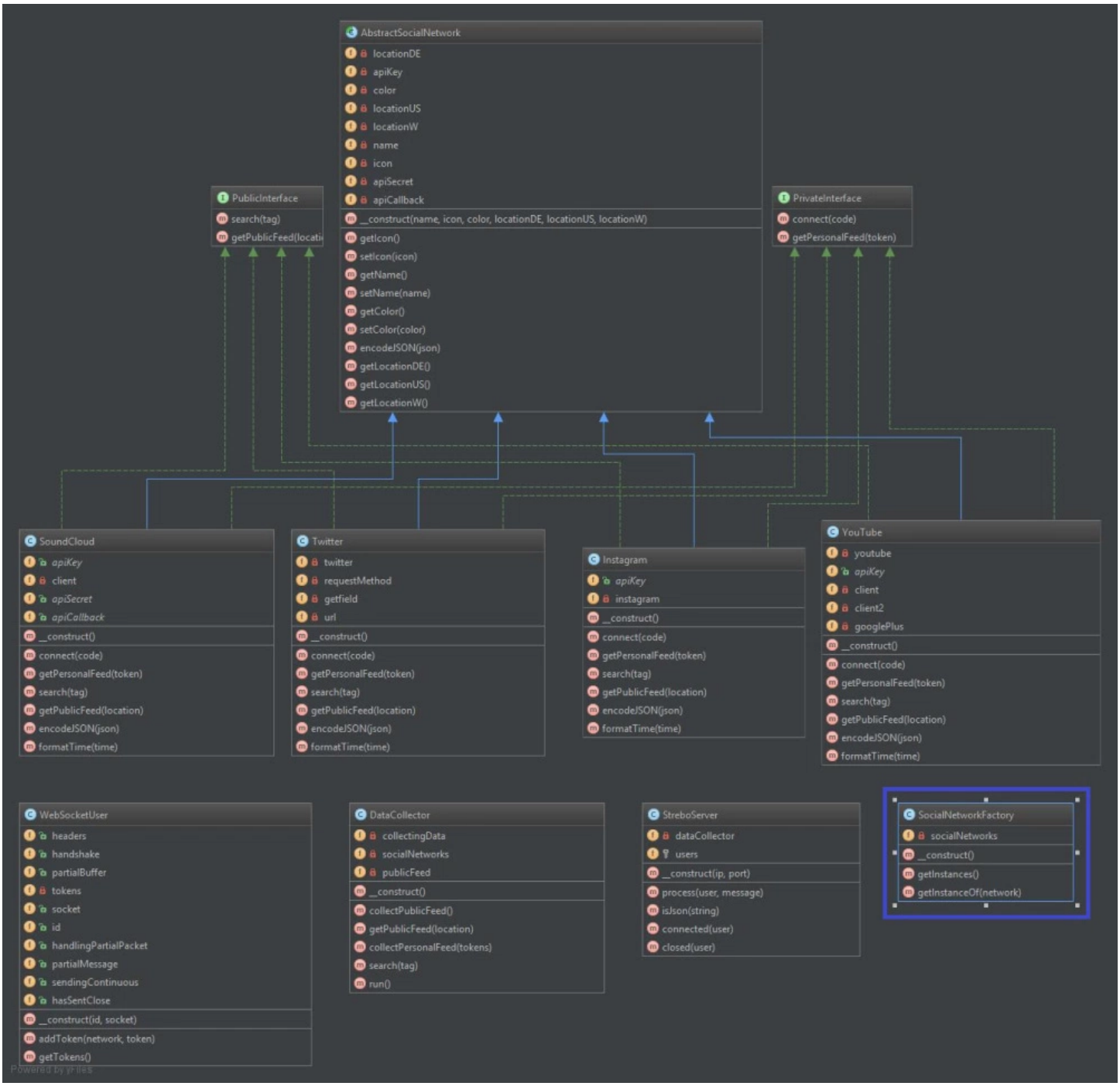
Hi fellows,

we implemented a Factory Pattern.

You can see our old Class-Diagram here:
(link for zooming in)



The new Diagram is shown here, the pattern is marked with a blue box:
(link for zooming in)



So you can see that a class ‘SocialNetworkFactory’ is added. This one creates instances of all available social networks and return these instances. Before this it was done by the ‘DataCollector’ class. For modularization reasons, maintaining the overview and maybe for future use cases – it makes sense to outsource this task.

Code before of DataCollector class: Data Collector Old
Code after of DataCollector:Data Collector New

Code of new SocialNetworkFactory: Social Network Factory

Best Greetings,
strebo.

Refactoring

Posted on 1. May 20161. May 2016 by streboparsegyan

Hey there,

Today each of us shows you how to refactor code. For our example we use an simple project. The used refactoring steps are based on Martin Fowlers book “Refactoring: Improving the Design of Existing Code”.

Here are the links to the single refactored example projects of our team members:

Aram: https://github.com/Aaper/Fowler
David: https://github.com/schreckda/Fowler
Fabian: https://github.com/ScientiaEtVeritas/Fowler

Best regards,
team strebo.

4

Time estimation for Semester II

Posted on 24. April 2016 by streboschreck

Our methods to estimate the time for our Use Cases in Semester II are the following:

- Function Points in relation with Time
- Weekly Sprints planned in Jira with planned Backlog
- Weekly Team coordination

/* TODO Insert Chart with Time estimation */

0

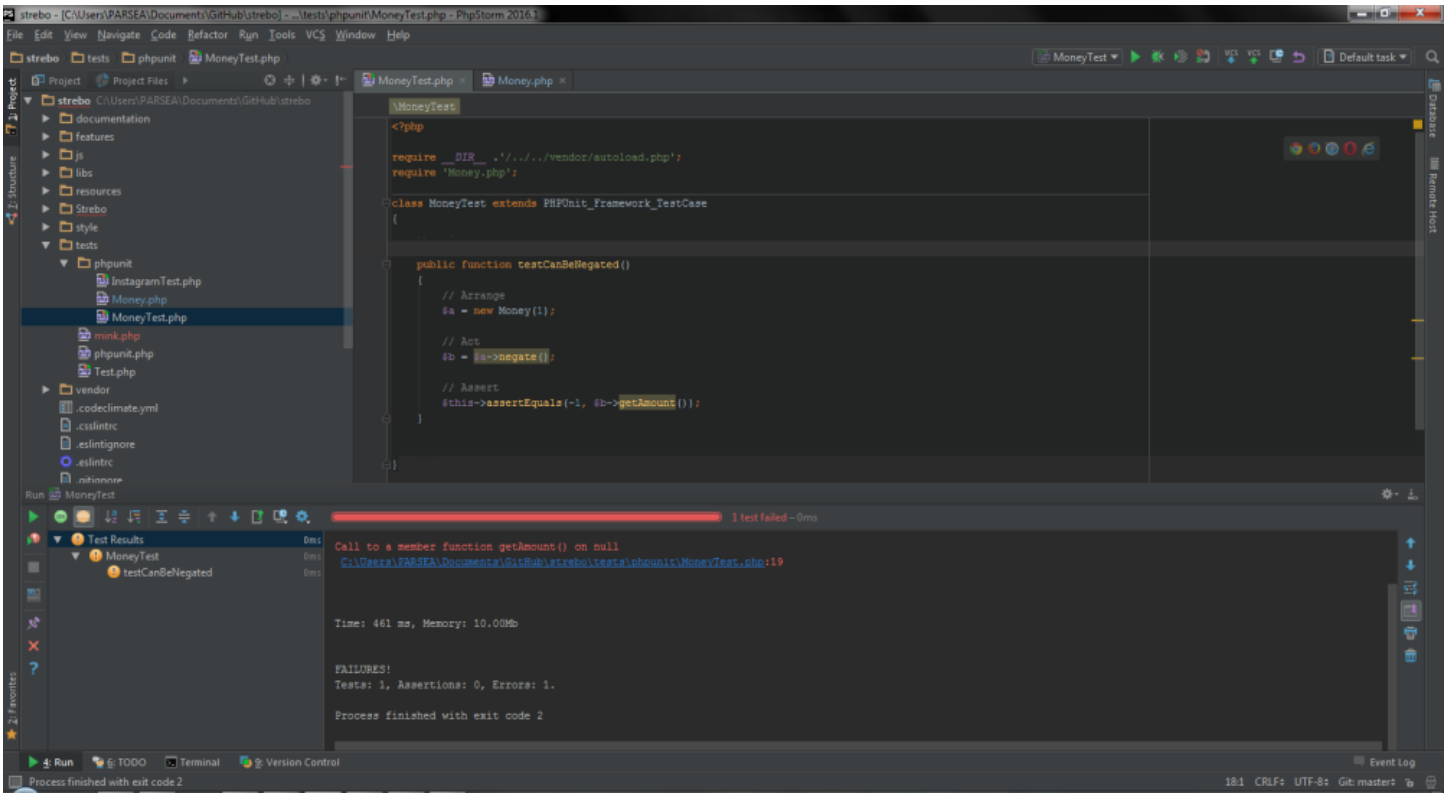
Unit Testing

Posted on 24. April 201627. April 2016 by streboretkowski

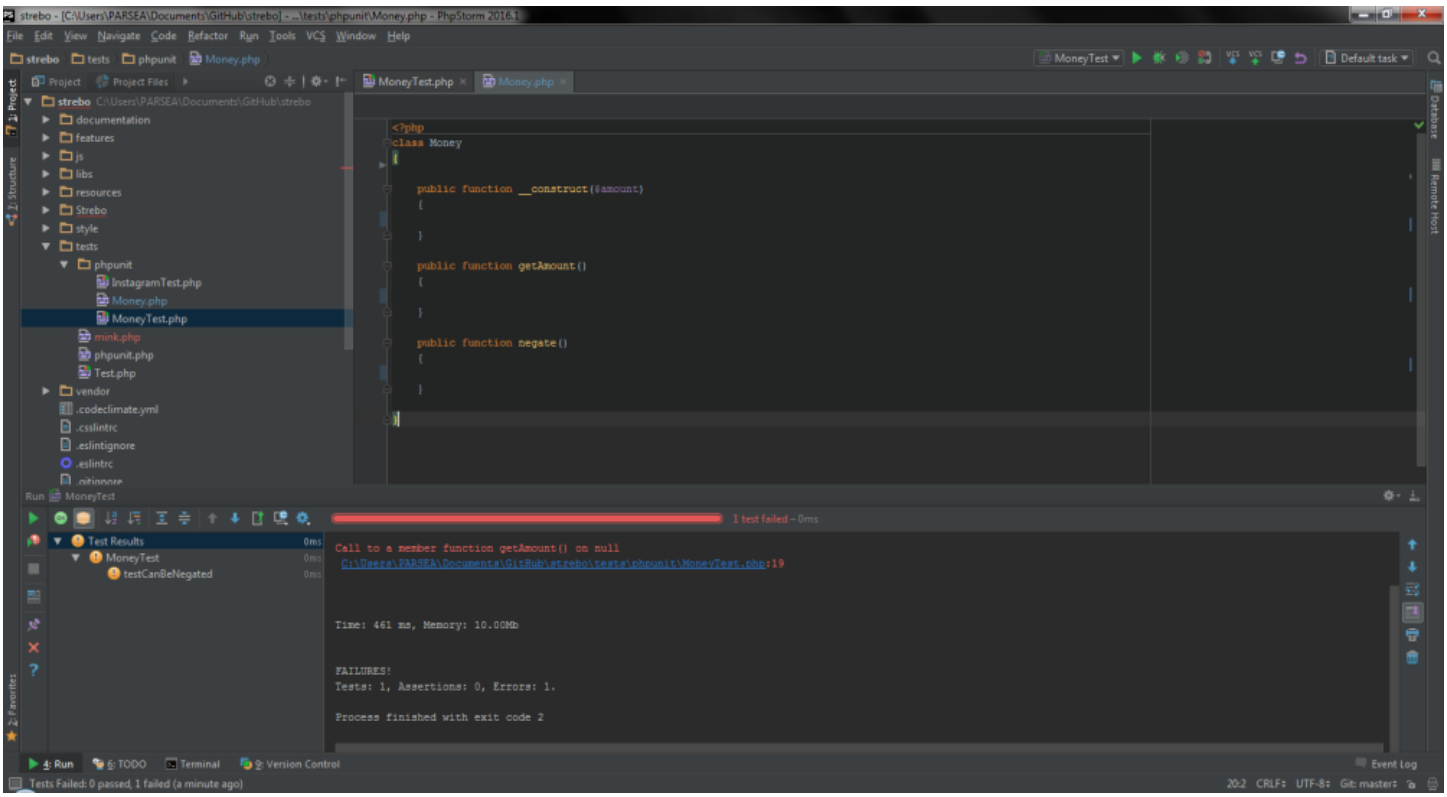
Hey there 🙂

Here is a short report about the TDD in our project. We chose an easy toycode example for demonstration. Furthermore we're using Phpstorm for our development. Phpstorm provides the posibility to easily create a test class for a newly created class.

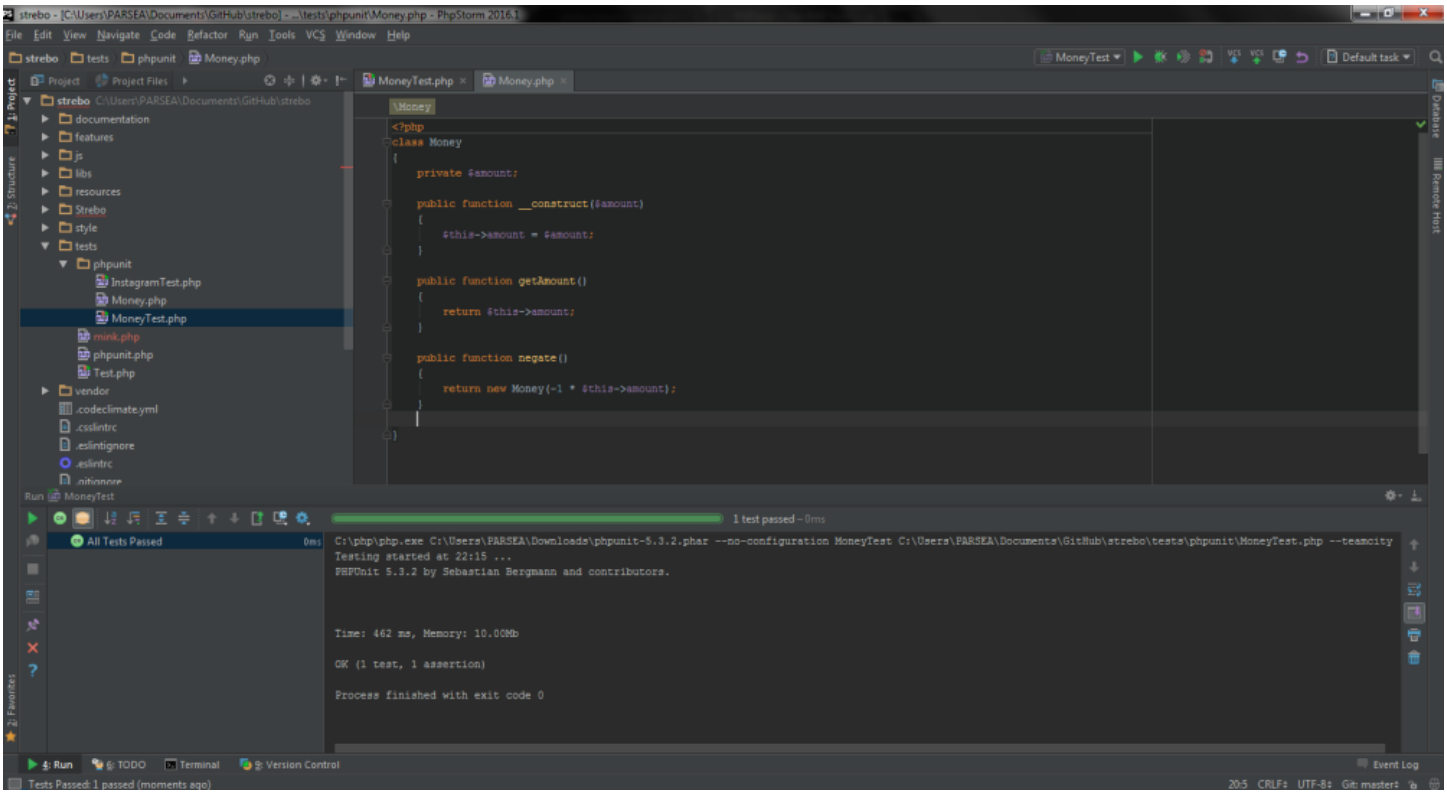
As you can see in this picture the testcode with the expected results is createtd at first.



If you try to run this testcode it is going to fail because the tested class itself is not implemented yet, as you can see in the next picture.

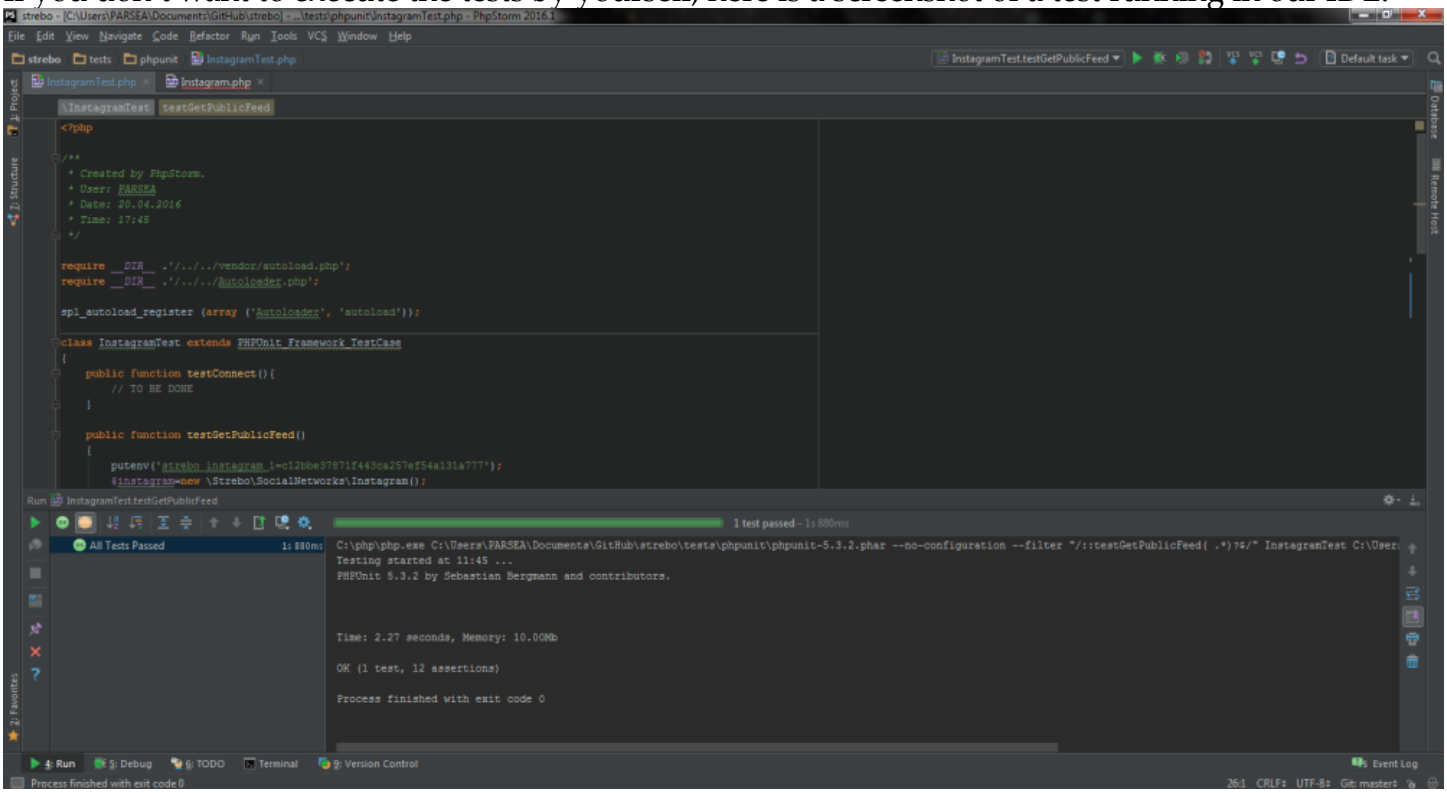


After the functionality of the specific class has been implemented, you can rerun the test and see the tests passing.



We have chosen PHPUnit as our testing framework to test our backend, PHP code and our PHP classes. Our test codings (including the toycode) can be found here: https://github.com/strebo/strebo/tree/master/tests

If you don't want to execute the tests by yourself, here is a screenshot of a test running in our IDE:

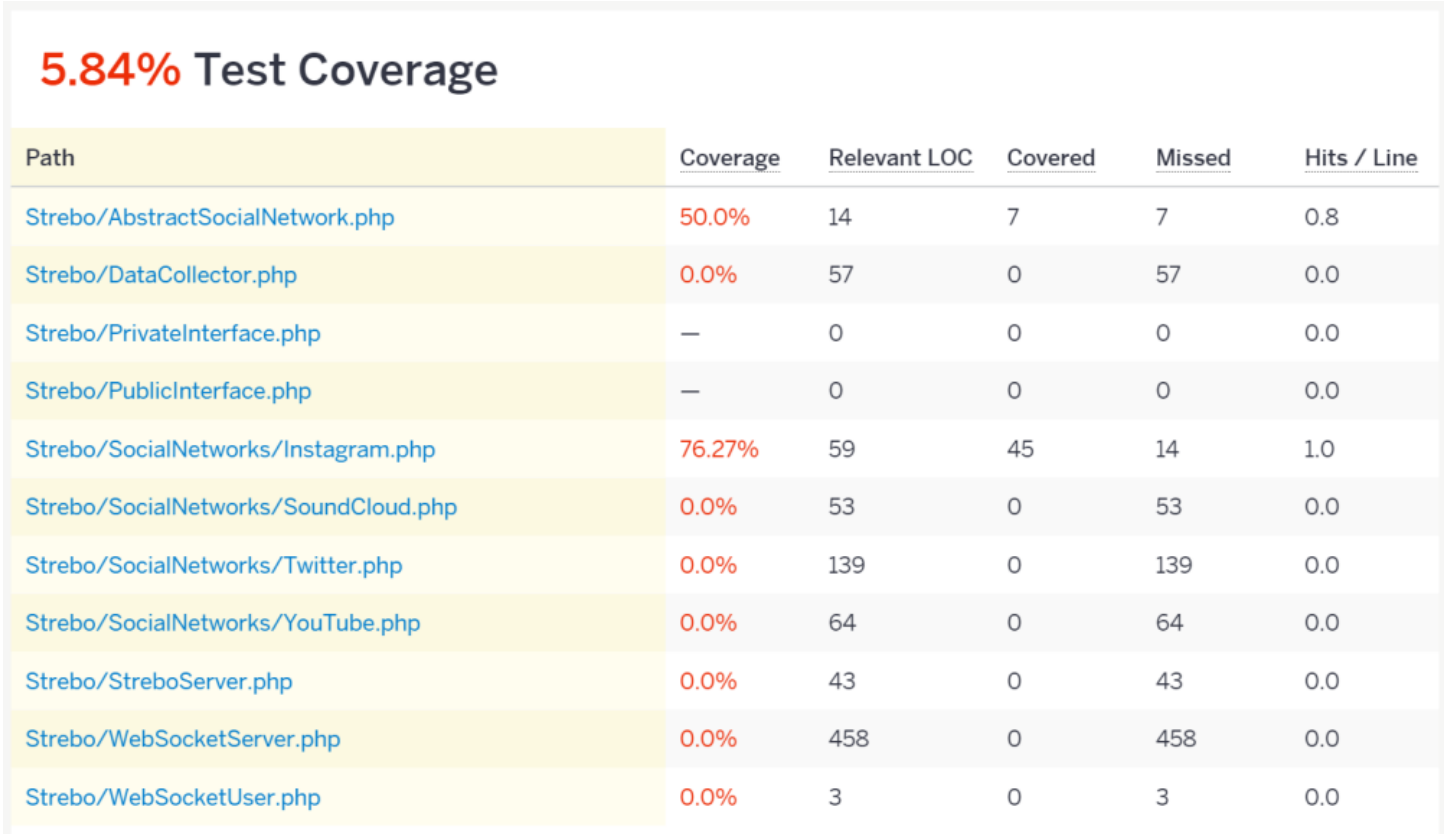


In the last week we also developed a sophisticated build and testing process:
We use Travis CI as platform for automated testing and our build process as a whole which will be triggered by GitHub pushes. See https://travis-ci.org/strebo/strebo.

The configuration file can be found here: https://github.com/strebo/strebo/blob/master/.travis.yml

In addition we set up a code quality and test coverage tool called Code Climate: https://codeclimate.com/github/strebo/strebo
The configuration file can be found here: https://github.com/strebo/strebo/blob/master/.codeclimate.yml
We also have configuration files for ESLint and CSS Lint: https://github.com/strebo/strebo/blob/master/.eslintrc and https://github.com/strebo/strebo/blob/master/.csslintrc

Here is screenshot of our working test coverage analysis:



5.84% Test Coverage

Path	Coverage	Relevant LOC	Covered	Missed	Hits / Line
Strebo/AbstractSocialNetwork.php	50.0%	14	7	7	0.8
Strebo/DataCollector.php	0.0%	57	0	57	0.0
Strebo/PrivateInterface.php	—	0	0	0	0.0
Strebo/PublicInterface.php	—	0	0	0	0.0
Strebo/SocialNetworks/Instagram.php	76.27%	59	45	14	1.0
Strebo/SocialNetworks/SoundCloud.php	0.0%	53	0	53	0.0
Strebo/SocialNetworks/Twitter.php	0.0%	139	0	139	0.0
Strebo/SocialNetworks/YouTube.php	0.0%	64	0	64	0.0
Strebo/StreboServer.php	0.0%	43	0	43	0.0
Strebo/WebSocketServer.php	0.0%	458	0	458	0.0
Strebo/WebSocketUser.php	0.0%	3	0	3	0.0

Best regards,
team strebo.

4

Function Points

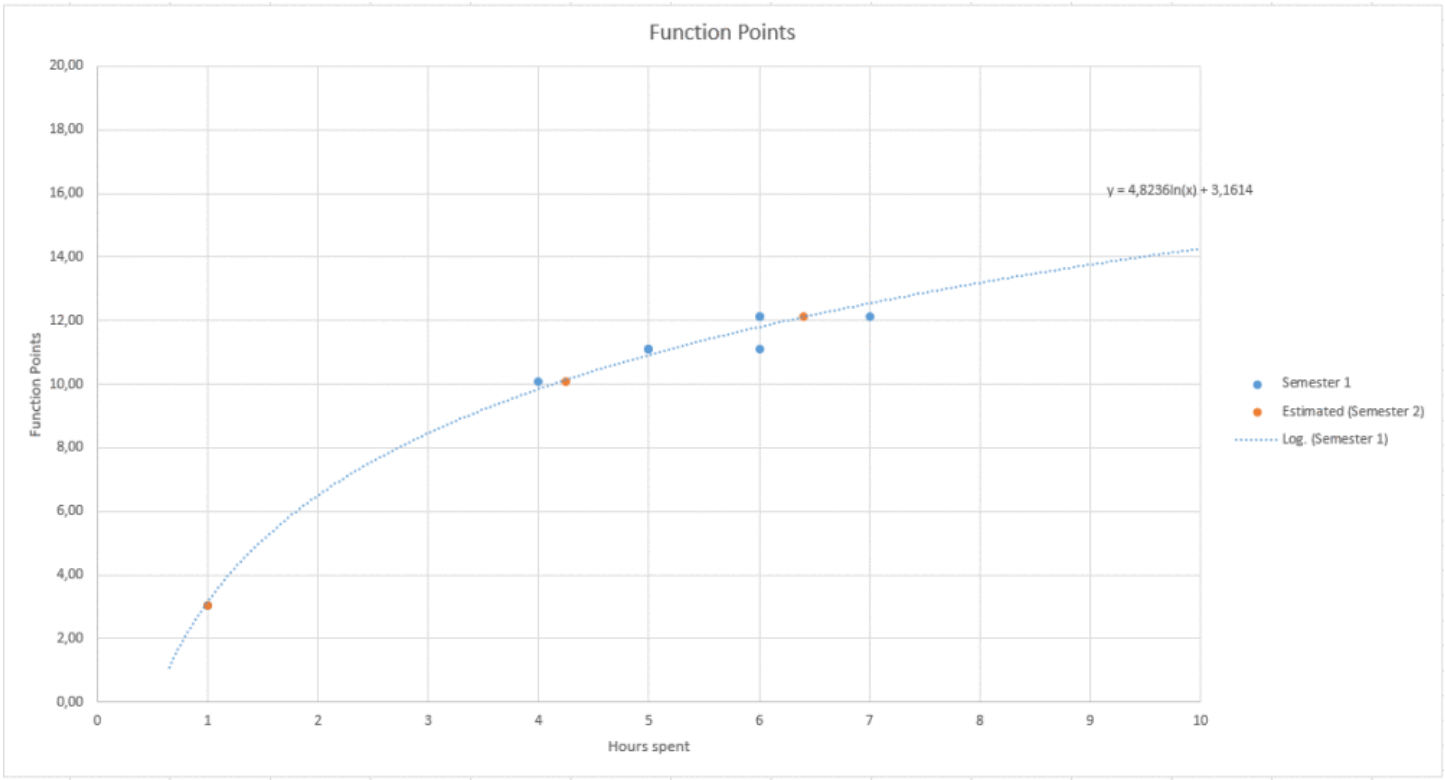
Posted on 18. April 201624. April 2016 by streboschreck

Hi Fellows,

we have calculated the Function Points for our UseCases.
You can find the documents here.

~~Our Function Points / Time graph needs some rework as there are still some discrepancies and we will update this post as soon as our data is correct.~~

Edit: We managed to overcome the discrepancies and are happy that we are now able to present you our Function Points / Time graph:



Best Greetings,
the strebo team.

3

Semester II – Use Cases and Risk Management

Posted on 10. April 2016 by streboschreck

Hello everybody,
the next semester has started and we will continue to work on strebo!

Fist of all we updated our Overall Use Case Diagramm to display our goals for this semester (the red ones):

Here are the links to the new Use-Cases:

Search for Topics and Hashtags
Show Newsfeed of your Networks
Drag'n'Drop the deck view items
Show Trends depending on Areas/Countries
Connect to Supported Social Media Network Accounts

We also created a "Use Case Effort Estimation" document which can be found here.

And last but not least we now have a "Risk Management Plan", found here.

An Overview for all our documents can be found here.

Best greetings,
the strebo team.

Midterm Project Report

Posted on 22. December 2015 by streboparsegyan

Hello everybody,

after a long time full of work we want to present you the results we have so far.

Our deployed project can be found at http://strebo.net/.

The Sourcecode of our project is available on GitHub as well as all of our documentations.

If you want to see how we organised our project have a look at our Gantt Chart and our Scrum Borard.

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

your Team strebo

0

Gantt Chart

Posted on 29. November 201530. November 2015 by streboparsegyan

Today we want to give you an overview of all the work we have done until today. To do so we created an Gant Chart which can be found here as a pdf.

4

Jira

Posted on 23. November 201523. November 2015 by streboschreck

We are now on Jira, which can be found here! 🙂

We already started our first sprint, that will take two weeks, found here.

On strebo.net you can already access the trend board with two different views, and three connected Social Networks.

Best greetings
the strebo team.

6