strebo
MENU

strebo – social trend board ❤

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!

Featured Post / 4 Comments /

Final

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

Best regards,
team strebo

14. June 201614. June 2016 / Leave a comment /

Installation

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 (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) :

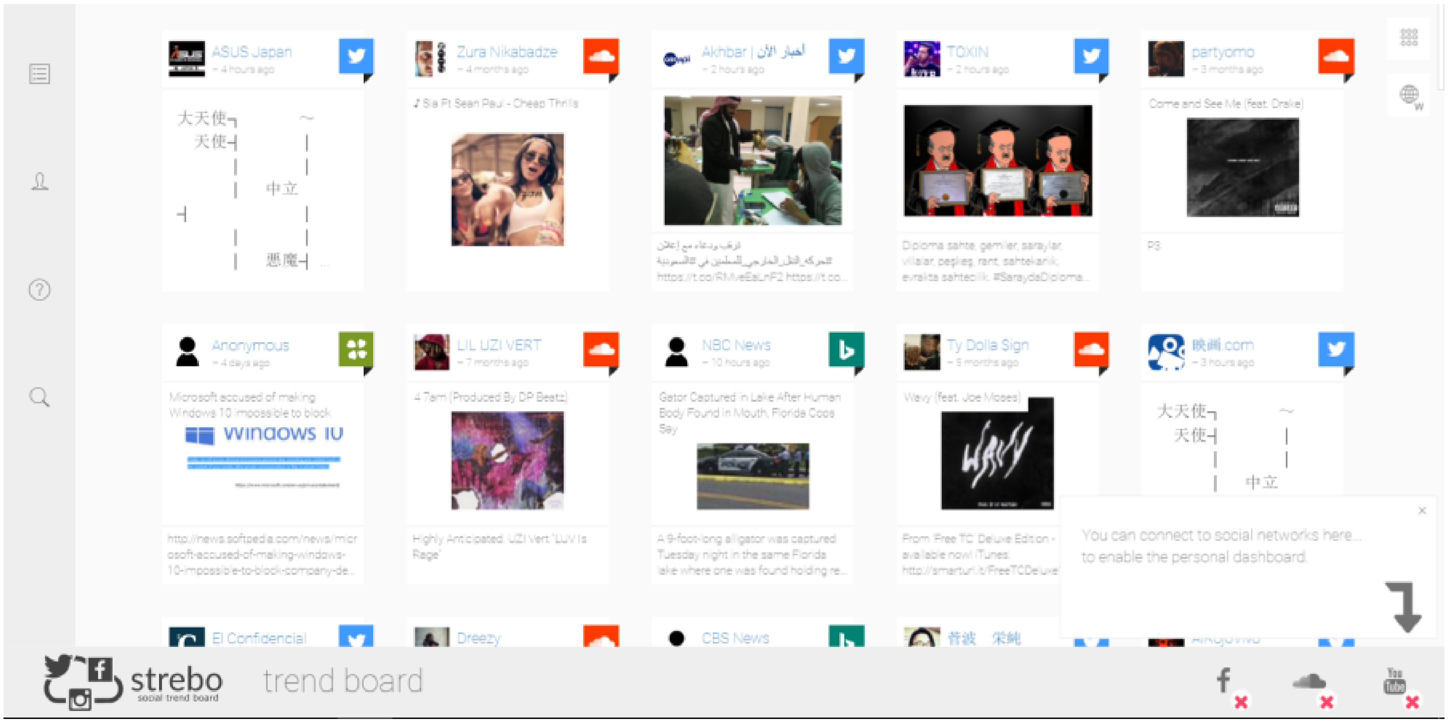
```
docker build -t strebo .
```

Then you have just to run it:

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

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 (http://localhost)**

Et voilà:



You can connect to social networks here to enable the personal dashboard.

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

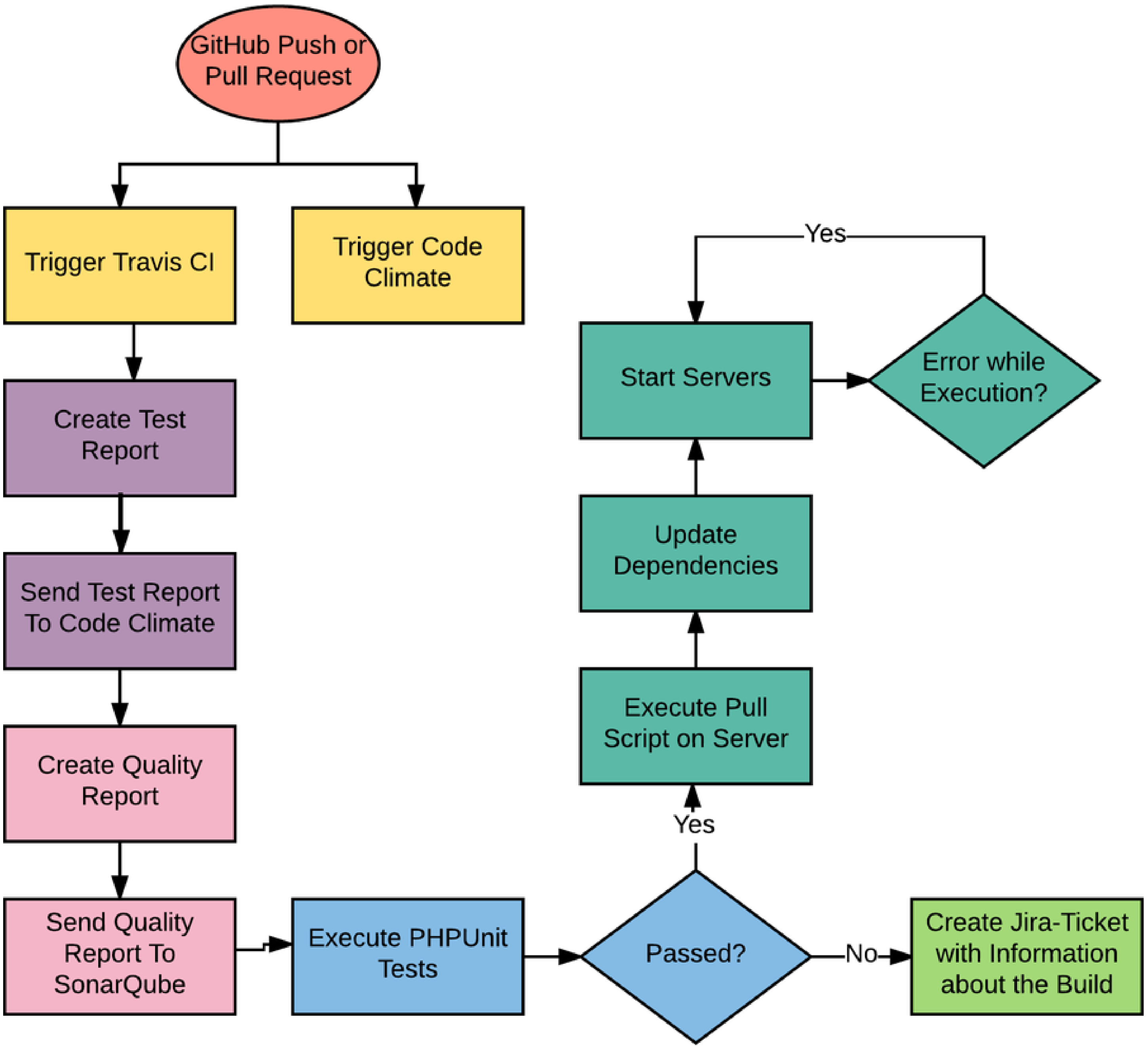
9. June 201615. June 2016 / 1 Comment /

Auto Depolyment

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:

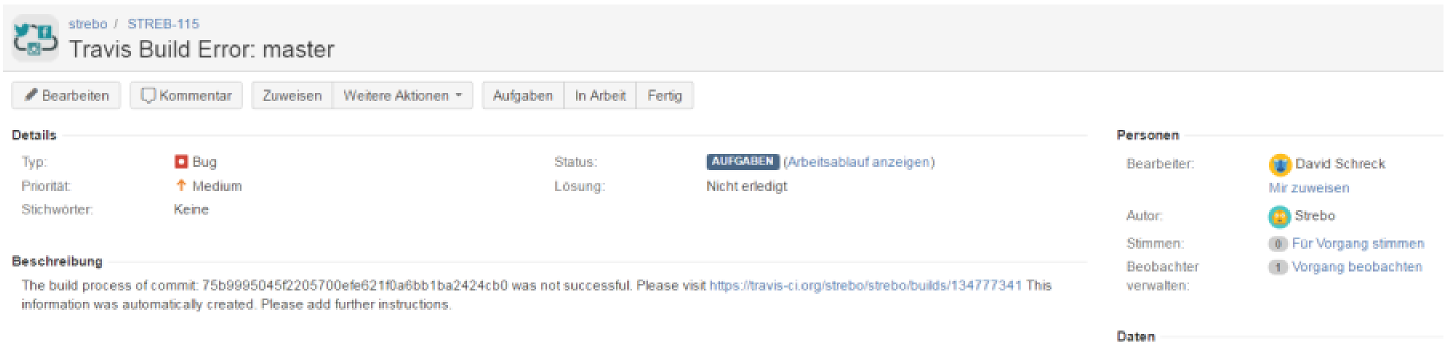

GitHub Push or Pull Request
Trigger Travis CI
Trigger Code Climate
Create Test Report
Send Test Report To Code Climate
Create Quality Report
Send Quality Report To SonarQube
Execute PHPUnit Tests
Passed?
No
Create Jira-Ticket with Information about the Build
Yes
Execute Pull Script on Server
Update Dependencies
Start Servers
Error while Execution?
Yes

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 (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/start.sh)https://github.com/strebo/strebo/blob/master/serverstart.sh (https://github.com/strebo/strebo/blob/master/serverstart.sh)https://github.com/strebo/strebo/blob/master/nserverstart.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:



strebo / STREB-115

Travis Build Error: master

Bearbeiten | Kommentar | Zuweisen | Weitere Aktionen | Aufgaben | In Arbeit | Fertig

Details

Typ:	Bug	Status:	AUFGABEN (Arbeitsablauf anzeigen)
Priorität:	Medium	Lösung:	Nicht erledigt
Stichwörter:	Keine		

Beschreibung

The build process of commit: 75b9995045f2205700efe621f0a6bb1ba2424cb0 was not successful. Please visit https://travis-ci.org/strebo/strebo/builds/134777341 This information was automatically created. Please add further instructions.

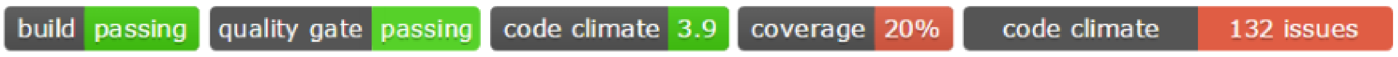
Personen

Bearbeiter:	David Schreck Mir zuweisen
Autor:	Strebo
Stimmen:	0 Für Vorgang stimmen
Beobachter verwalten:	1 Vorgang beobachten

Daten

Bash Script which informs Jira:
https://github.com/strebo/strebo/blob/master/.send_ticket.sh
(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 (https://github.com/strebo/strebo/wiki/Test-Document)

Best regards,
strebo team

3. June 20163. June 2016 / 4 Comments /

Metrics

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

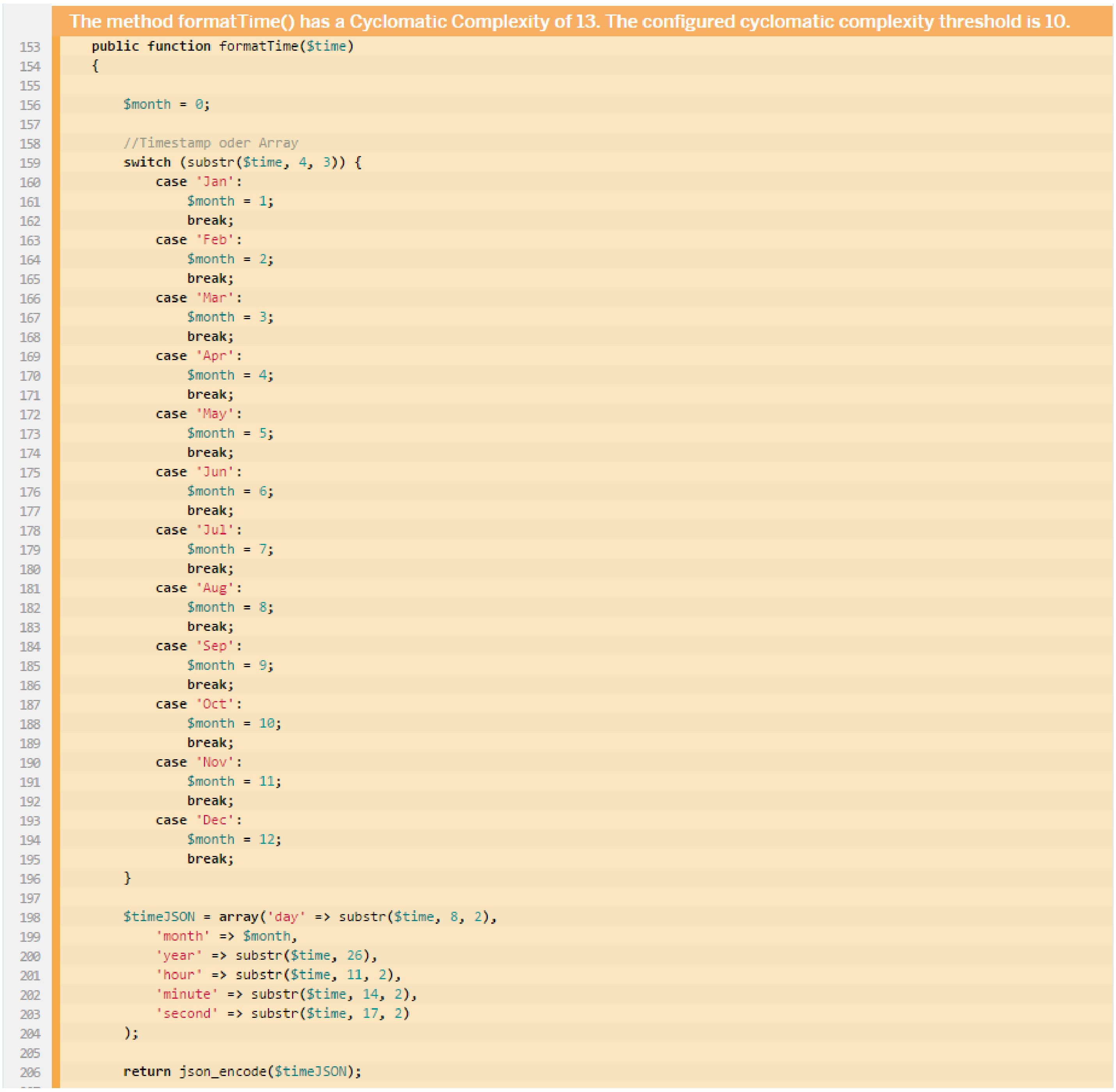
SonarQube: http://sonarqube.it.dh-karlsruhe.de/overview?id=5357 (http://sonarqube.it.dh-karlsruhe.de/overview?id=5357)

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

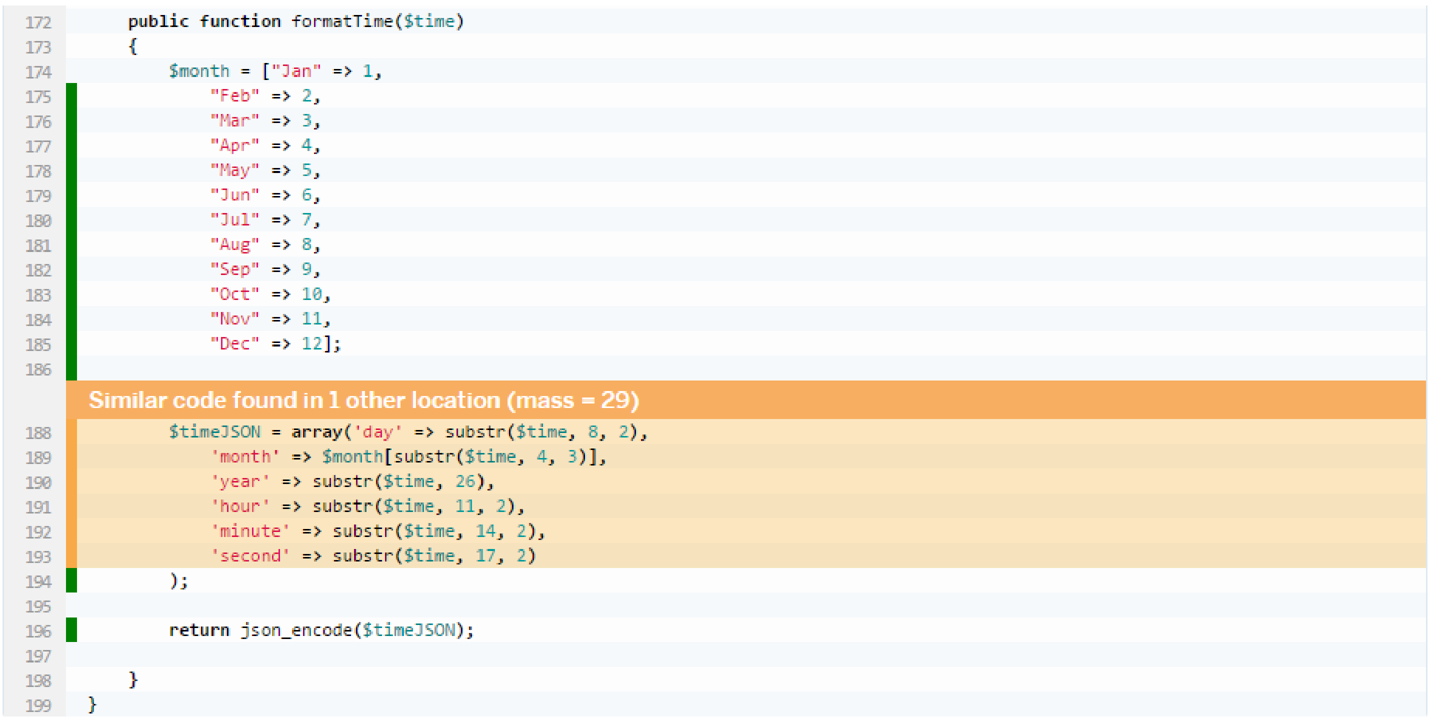


The method formatTime() has a Cyclomatic Complexity of 13. The configured cyclomatic complexity threshold is 10.

```php
    public function formatTime($time)
    {

        $month = 0;

        //Timestamp oder Array
        switch (substr($time, 4, 3)) {
            case 'Jan':
                $month = 1;
                break;
            case 'Feb':
                $month = 2;
                break;
            case 'Mar':
                $month = 3;
                break;
            case 'Apr':
                $month = 4;
                break;
            case 'May':
                $month = 5;
                break;
            case 'Jun':
                $month = 6;
                break;
            case 'Jul':
                $month = 7;
                break;
            case 'Aug':
                $month = 8;
                break;
            case 'Sep':
                $month = 9;
                break;
            case 'Oct':
                $month = 10;
                break;
            case 'Nov':
                $month = 11;
                break;
            case 'Dec':
                $month = 12;
                break;
        }

        $timeJSON = array('day' => substr($time, 8, 2),
            'month' => $month,
            'year' => substr($time, 26),
            'hour' => substr($time, 11, 2),
            'minute' => substr($time, 14, 2),
            'second' => substr($time, 17, 2)
        );

        return json_encode($timeJSON);
```

(https://raw.githubusercontent.com/strebo/strebo/master/documentation/Metrics_complexity_twitter_1.png)

This switch can be replaced with a simple associative array:



```
    public function formatTime($time)
    {
        $month = ["Jan" => 1,
            "Feb" => 2,
            "Mar" => 3,
            "Apr" => 4,
            "May" => 5,
            "Jun" => 6,
            "Jul" => 7,
            "Aug" => 8,
            "Sep" => 9,
            "Oct" => 10,
            "Nov" => 11,
            "Dec" => 12];

Similar code found in 1 other location (mass = 29)
        $timeJSON = array('day' => substr($time, 8, 2),
            'month' => $month[substr($time, 4, 3)],
            'year' => substr($time, 26),
            'hour' => substr($time, 11, 2),
            'minute' => substr($time, 14, 2),
            'second' => substr($time, 17, 2)
        );

        return json_encode($timeJSON);

    }
}
```

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.

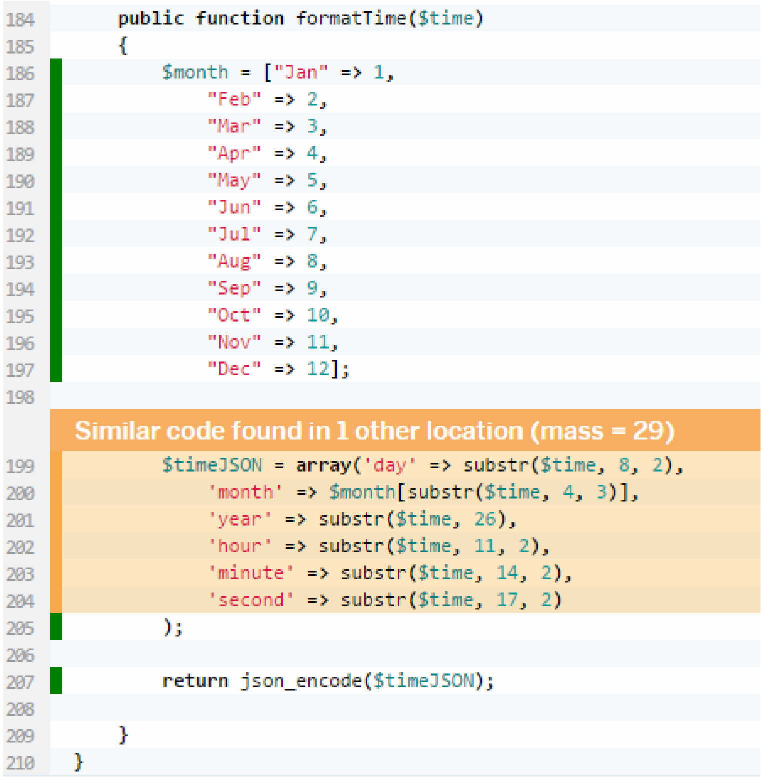
Similar code

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:

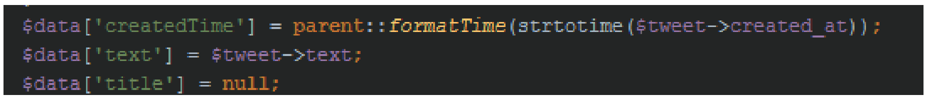


```php
    public function formatTime($time)
    {
        $month = ["Jan" => 1,
            "Feb" => 2,
            "Mar" => 3,
            "Apr" => 4,
            "May" => 5,
            "Jun" => 6,
            "Jul" => 7,
            "Aug" => 8,
            "Sep" => 9,
            "Oct" => 10,
            "Nov" => 11,
            "Dec" => 12];

```

Similar code found in 1 other location (mass = 29)

```php
        $timeJSON = array('day' => substr($time, 8, 2),
            'month' => $month[substr($time, 4, 3)],
            'year' => substr($time, 26),
            'hour' => substr($time, 11, 2),
            'minute' => substr($time, 14, 2),
            'second' => substr($time, 17, 2)
        );

        return json_encode($timeJSON);

    }
}
```

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:

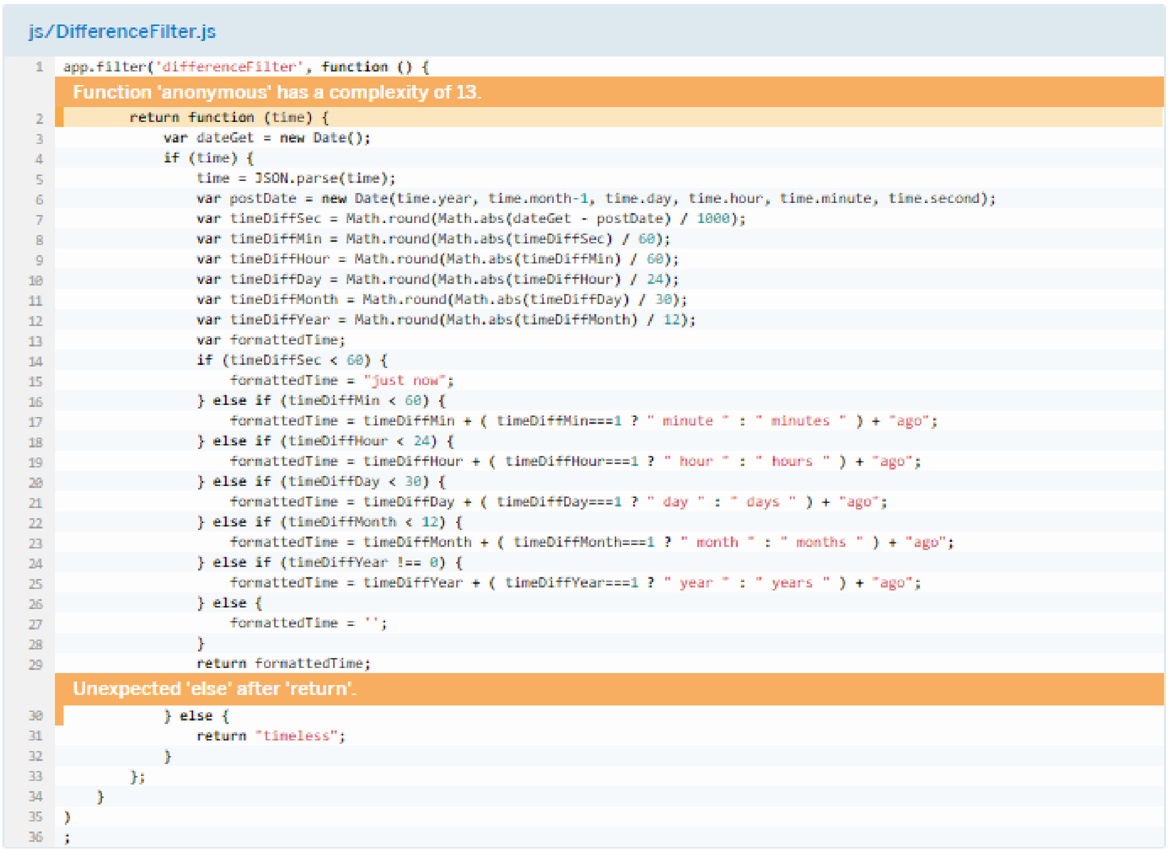


```php
$data['createdTime'] = parent::formatTime(strtotime($tweet->created_at));
$data['text'] = $tweet->text;
$data['title'] = null;
```

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 (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.

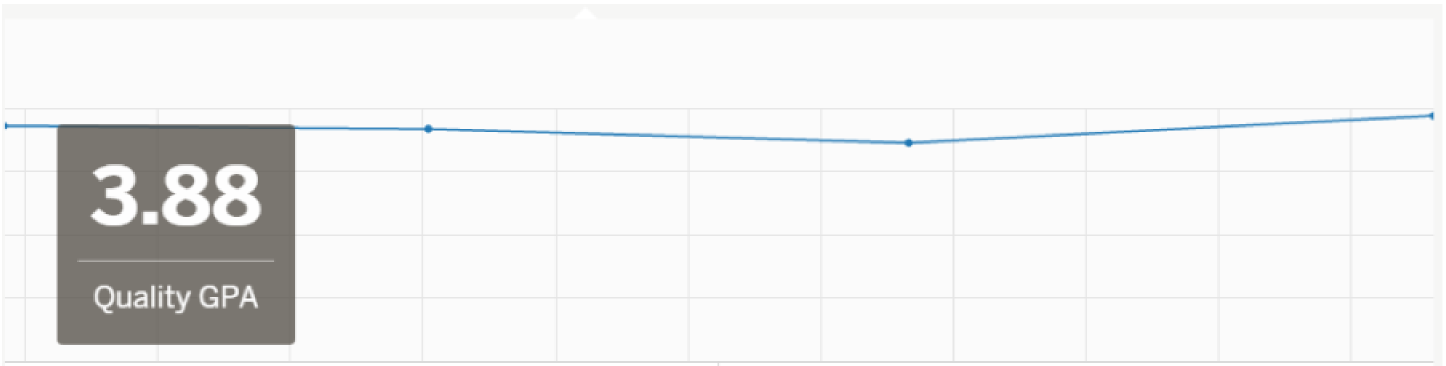
```
        return function (time) {
            var dateGet = new Date();
            if (time) {
                time = JSON.parse(time);
                var postDate = new Date(time.year, time.month-1, time.day, time.hour, time.minute, time.second);
                var timeDiffSec = Math.round(Math.abs(dateGet - 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;
```

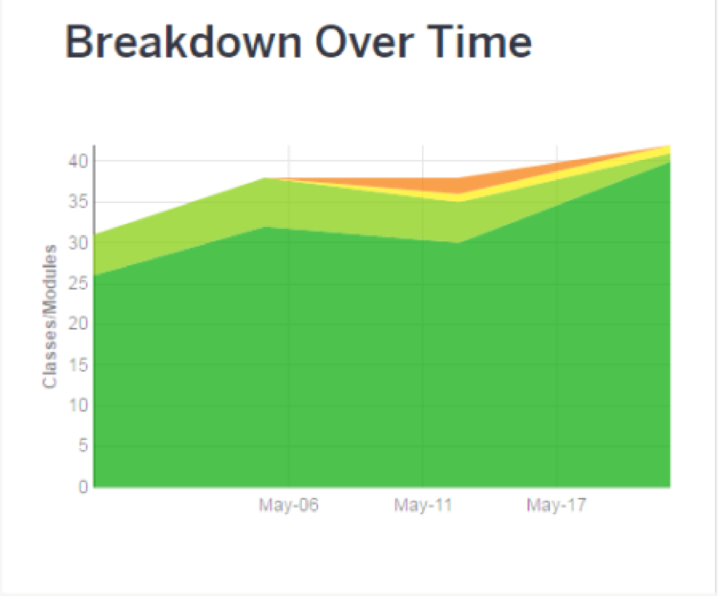
Unexpected 'else' after 'return'.

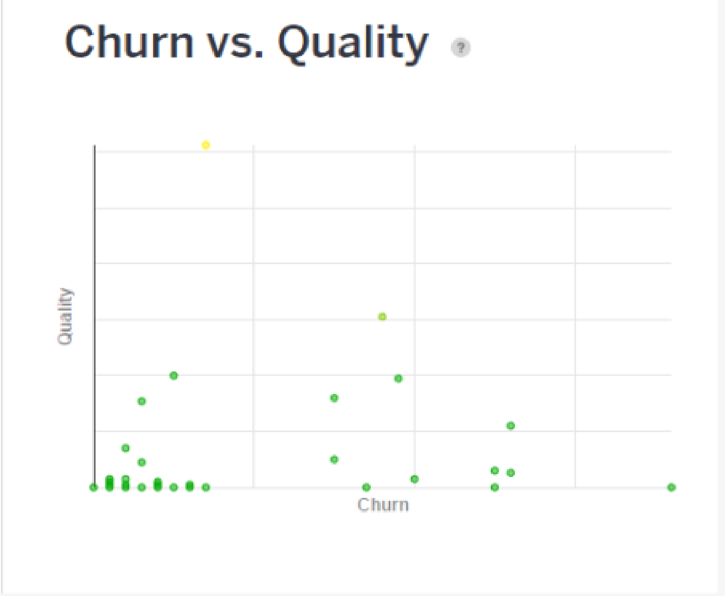
```
            } else {
                return "timeless";
            }
        };
    }
)
;
```

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

22. May 20162. June 2016 / 4 Comments /

Test Coverage

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 (https://codeclimate.com/github/strebo/strebo/coverage)

It also offers a badge: coverage 31%

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

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

Best regards,
team strebo

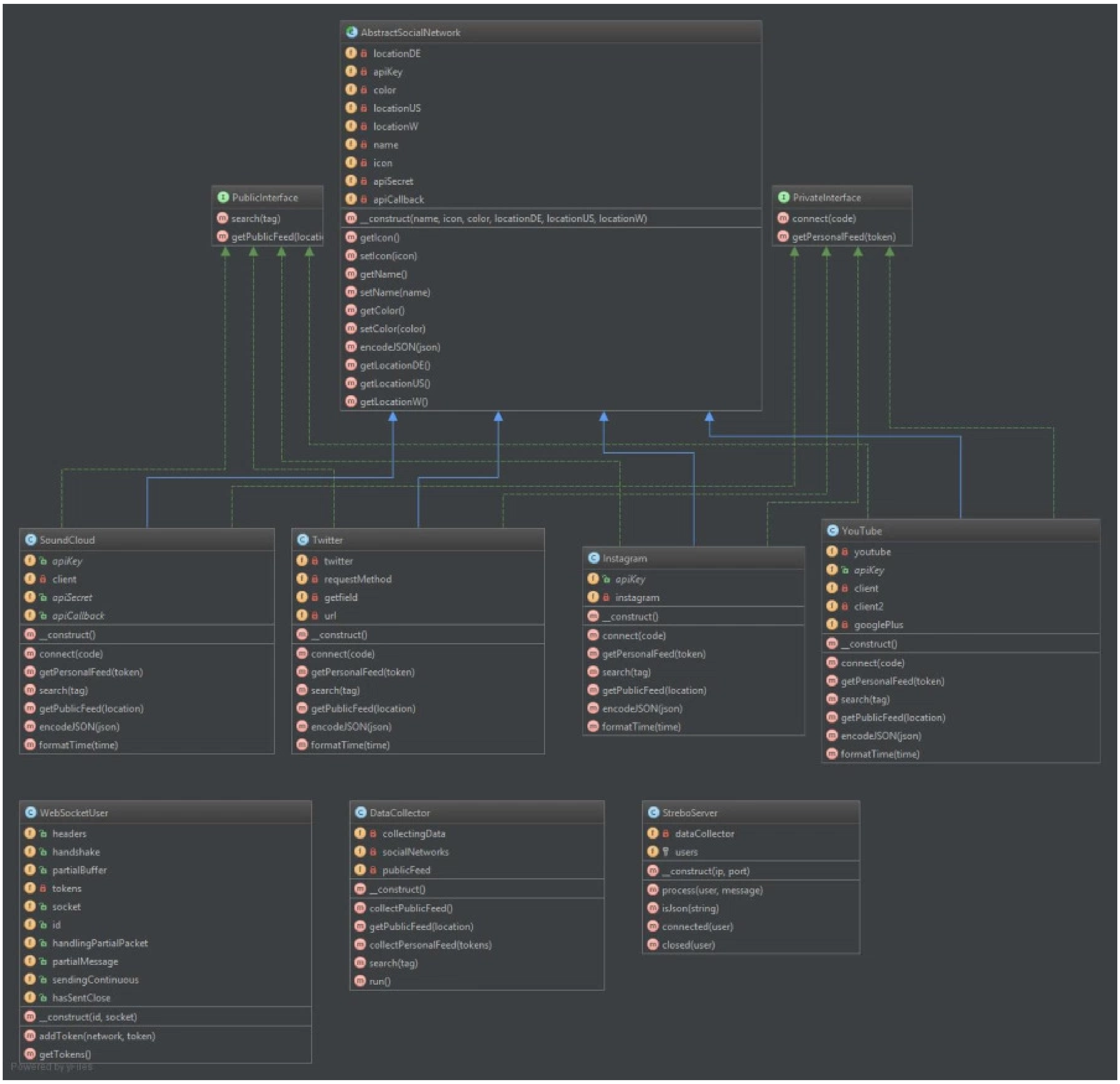
22. May 201623. May 2016 / 4 Comments /

Design Pattern

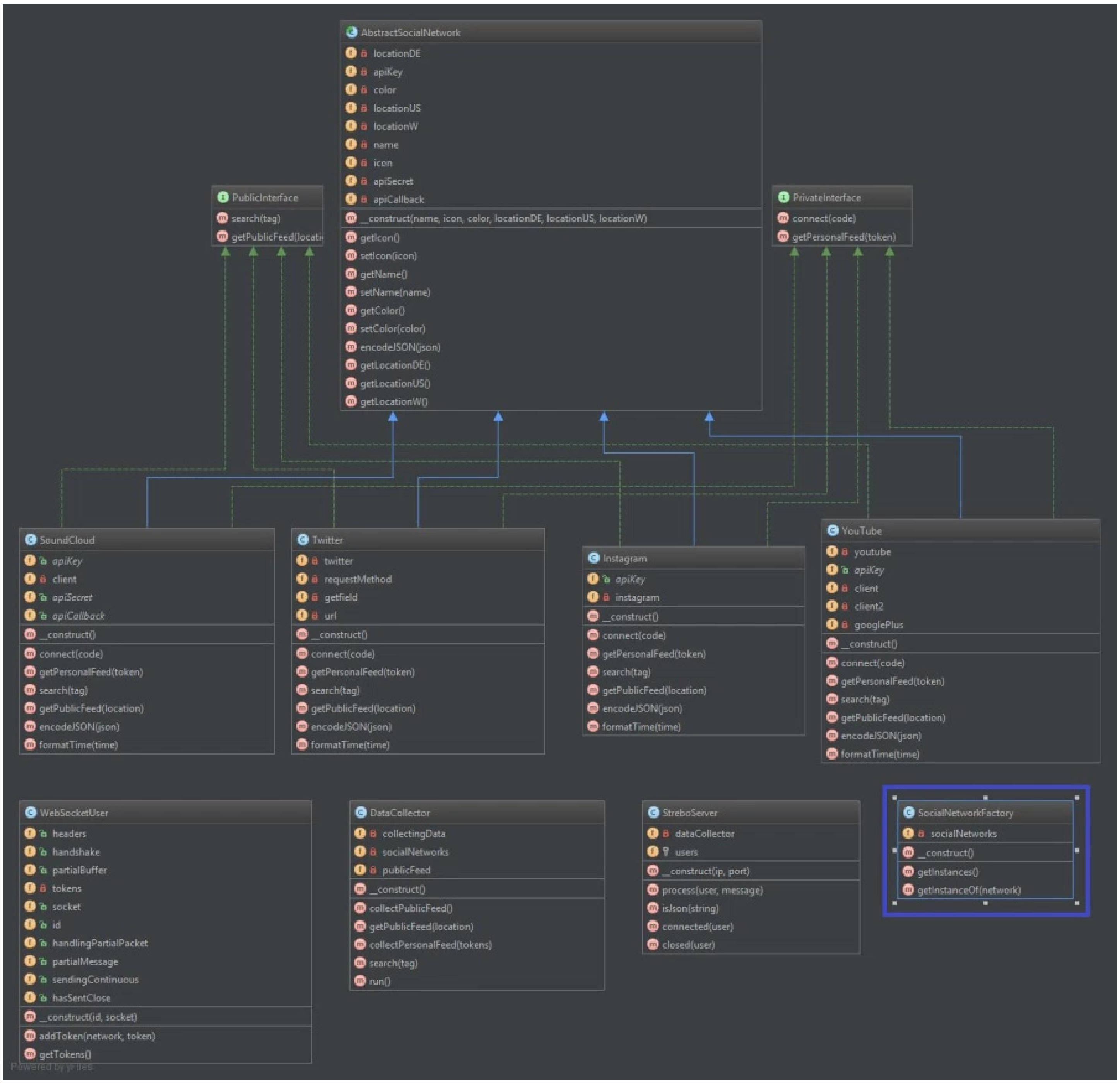
Hi fellows,

we implemented a Factory Pattern.

You can see our old Class-Diagram here:
(link for zooming in (https://github.com/strebo/strebo/blob/master/documentation/strebo_uml.jpg))



The new Diagram is shown here, the pattern is marked with a blue box:
(link for zooming in (https://github.com/strebo/strebo/blob/master/documentation/strebo_uml_new.jpg))



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 (https://github.com/strebo/strebo/blob/b76fa858a0077cac43b1a8c88d16c5be671798c0/Strebo/DataCollector.php)
Code after of DataCollector:Data Collector New (https://github.com/strebo/strebo/blob/935b2cd9c8712f5656c741f276ed2812ee1ab823/Strebo/DataCollector.php)

Code of new SocialNetworkFactory: Social Network Factory (https://github.com/strebo/strebo/blob/935b2cd9c8712f5656c741f276ed2812ee1ab823/Strebo/SocialNetworkFactory.php)

Best Greetings,
strebo.

8. May 20169. May 2016 / 4 Comments /

Refactoring

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

Best regards,
team strebo.

1. May 20161. May 2016 / 4 Comments /

Time estimation for Semester II

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 */

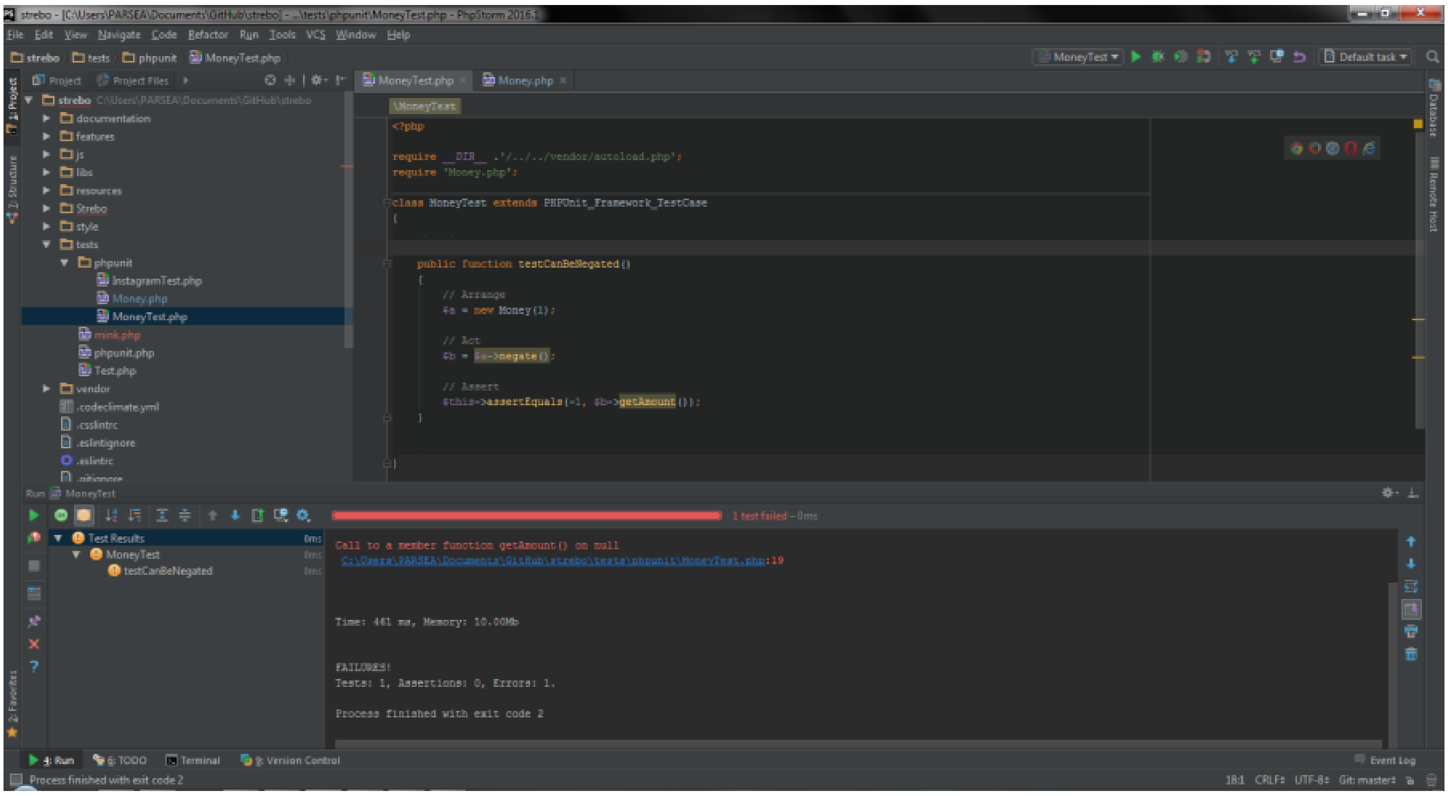
24. April 2016 / Leave a comment /

Unit Testing

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.

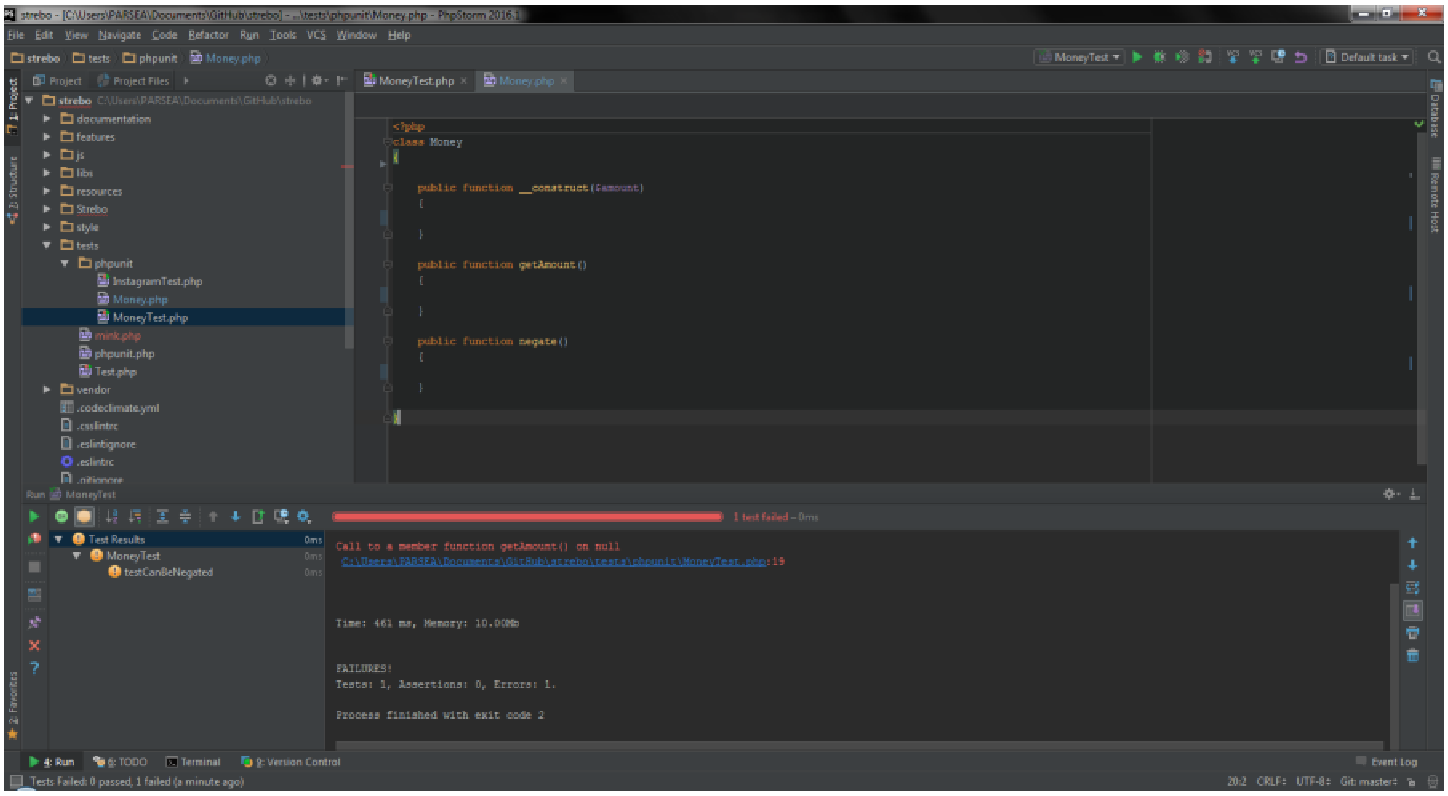
As you can see in this picture the testcode with the expected results is createtd at first.



```
<?php

require __DIR__ .'/../../vendor/autoload.php';
require 'Money.php';

class MoneyTest extends PHPUnit_Framework_TestCase
{

    public function testCanBeNegated()
    {
        // Arrange
        $a = new Money(1);

        // Act
        $b = $a->negate();

        // Assert
        $this->assertEquals(-1, $b->getAmount());
    }

}
```

```
Call to a member function getAmount() on null
 C:\Users\PARSEA\Documents\GitHub\strebo\tests\phpunit\MoneyTest.php:19



Time: 461 ms, Memory: 10.00Mb


FAILURES!
Tests: 1, Assertions: 0, Errors: 1.

Process finished with exit code 2
```

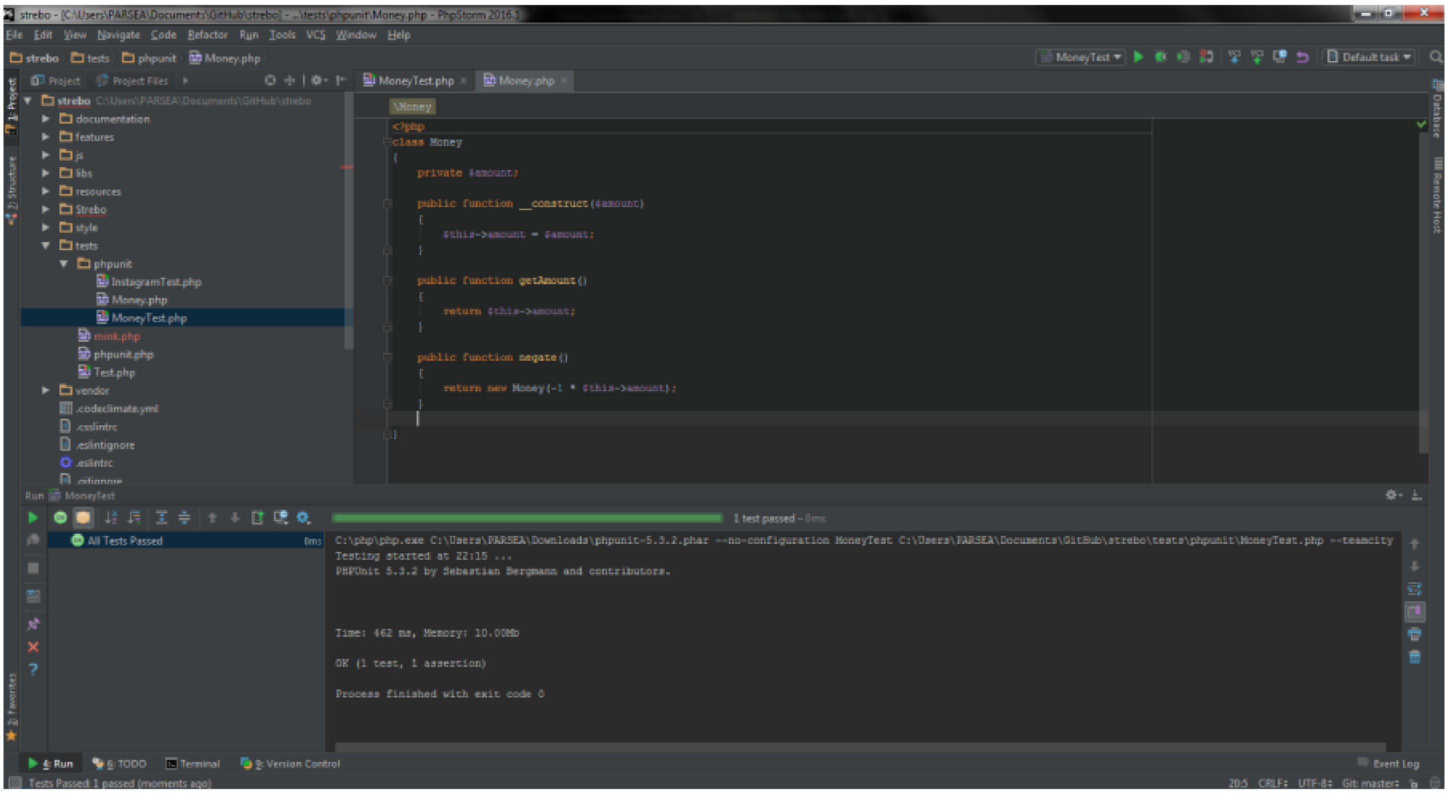
If you try to run this testcode it is going to fail because the tested class itself is not implemented yet, as you can see in the next picture.



```
<?php
class Money
{

    public function __construct($amount)
    {

    }

    public function getAmount()
    {

    }

    public function negate()
    {

    }

}
```

```
Call to a member function getAmount() on null
 C:\Users\PARSEA\Documents\GitHub\strebo\tests\phpunit\MoneyTest.php:19



Time: 461 ms, Memory: 10.00Mb


FAILURES!
Tests: 1, Assertions: 0, Errors: 1.

Process finished with exit code 2
```

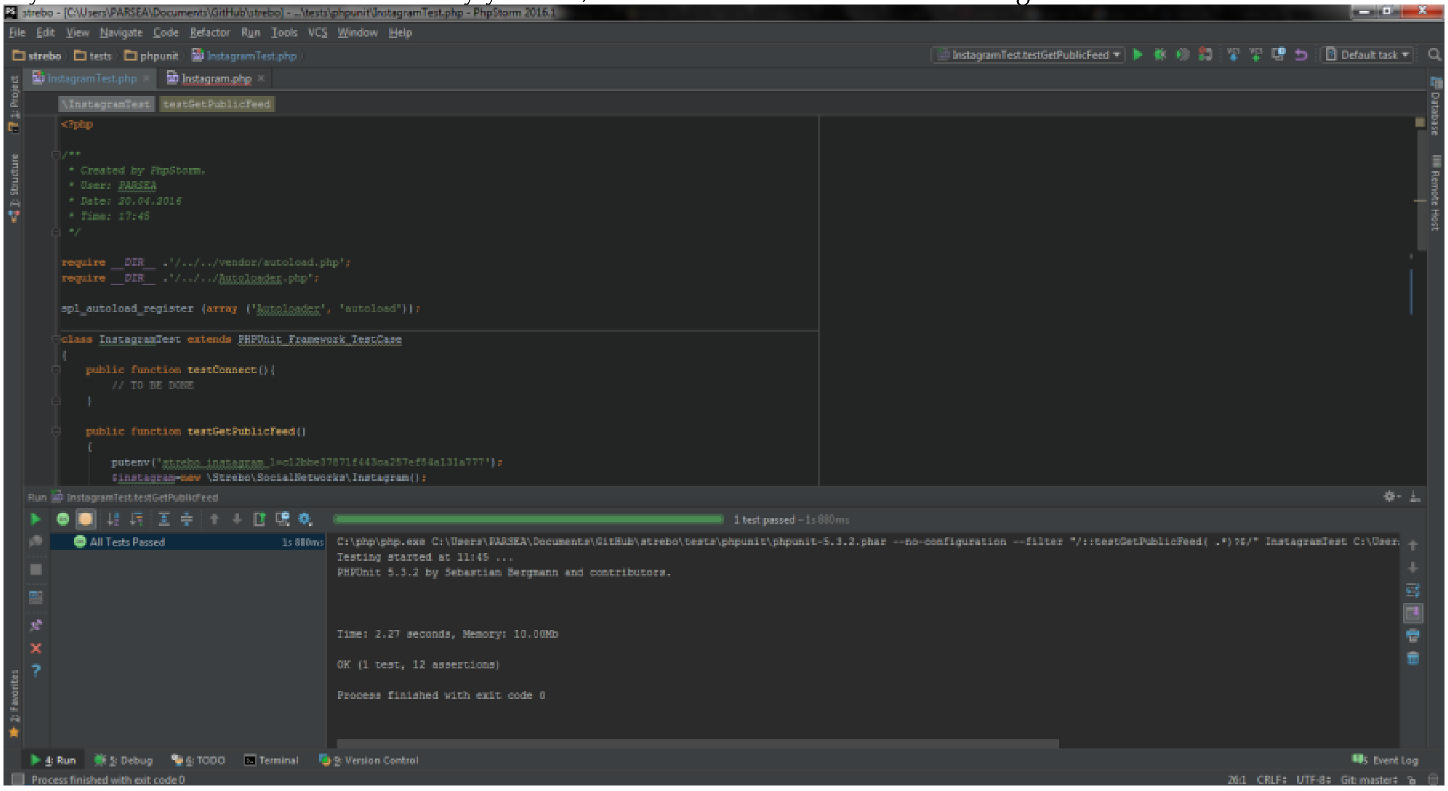
After the functionality of the specific class has been implemented, you can rerun the test and see the tests passing.


```php
<?php
class Money
{
    private $amount;

    public function __construct($amount)
    {
        $this->amount = $amount;
    }

    public function getAmount()
    {
        return $this->amount;
    }

    public function negate()
    {
        return new Money(-1 * $this->amount);
    }

}
```

```
C:\php\php.exe C:\Users\PARSEA\Downloads\phpunit-5.3.2.phar --no-configuration MoneyTest C:\Users\PARSEA\Documents\GitHub\strebo\tests\phpunit\MoneyTest.php --teamcity
Testing started at 22:15 ...
PHPUnit 5.3.2 by Sebastian Bergmann and contributors.



Time: 462 ms, Memory: 10.00Mb

OK (1 test, 1 assertion)

Process finished with exit code 0
```

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 (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:


```php
<?php

/**
 * Created by PhpStorm.
 * User: PARSEA
 * Date: 20.04.2016
 * Time: 17:45
 */

require __DIR__ .'/../../vendor/autoload.php';
require __DIR__ .'/../../Autoloader.php';

spl_autoload_register (array ('Autoloader', 'autoload'));

class InstagramTest extends PHPUnit_Framework_TestCase
{
    public function testConnect(){
        // TO BE DONE
    }

    public function testGetPublicFeed()
    {
        putenv('strebo_instagram_1=c12bbe37871f443ca257ef54a131a777');
        $instagram=new \Strebo\SocialNetworks\Instagram();
```

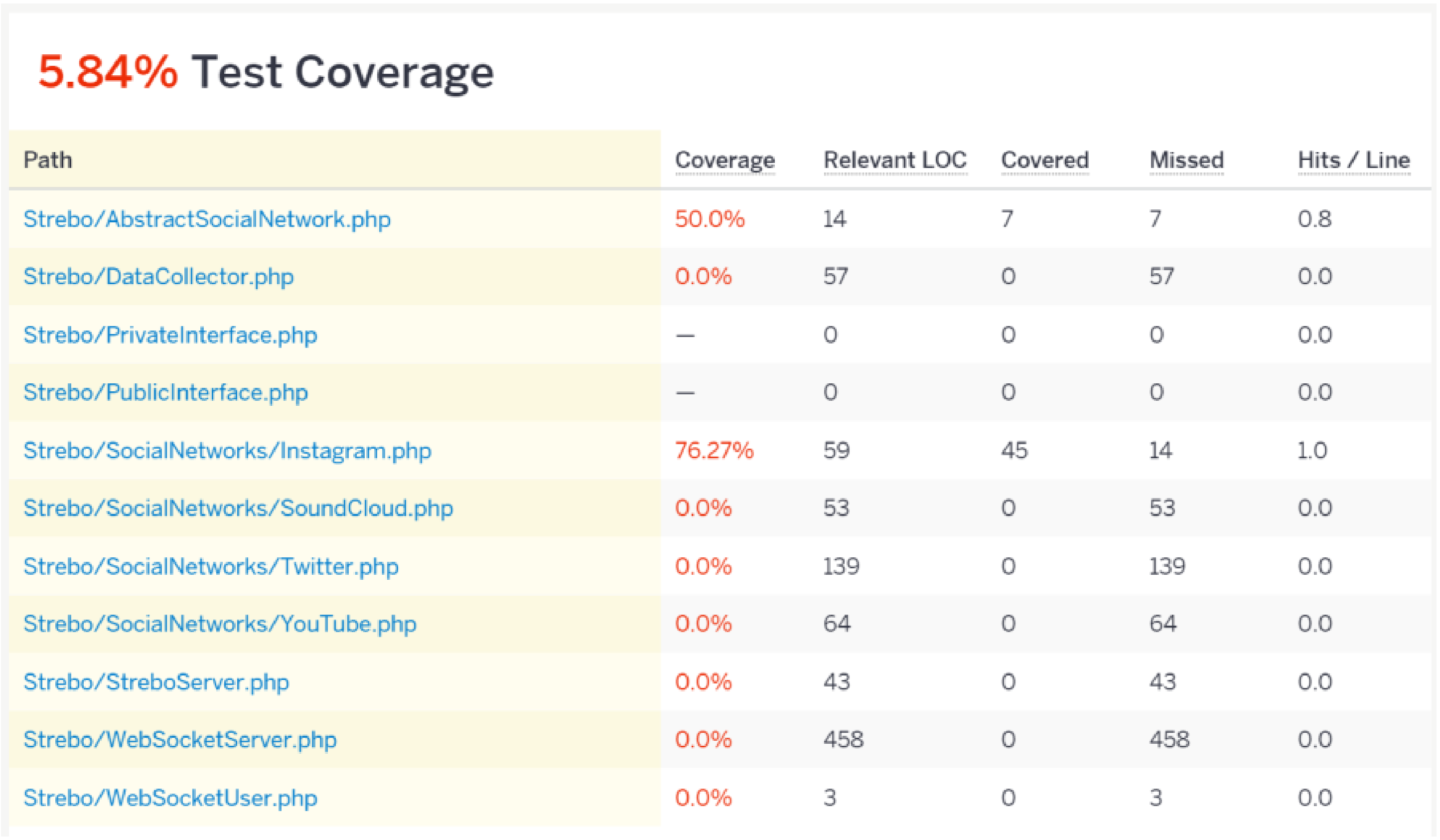
```
C:\php\php.exe C:\Users\PARSEA\Documents\GitHub\strebo\tests\phpunit\phpunit-5.3.2.phar --no-configuration --filter "/::testGetPublicFeed( .*)?$/" InstagramTest C:\User
Testing started at 11:45 ...
PHPUnit 5.3.2 by Sebastian Bergmann and contributors.



Time: 2.27 seconds, Memory: 10.00Mb

OK (1 test, 12 assertions)

Process finished with exit code 0
```

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 (https://travis-ci.org/strebo/strebo).

The configuration file can be found here: https://github.com/strebo/strebo/blob/master/.travis.yml (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 (https://codeclimate.com/github/strebo/strebo)
The configuration file can be found here: https://github.com/strebo/strebo/blob/master/.codeclimate.yml (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 (https://github.com/strebo/strebo/blob/master/.eslintrc) and https://github.com/strebo/strebo/blob/master/.csslintrc (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.

24. April 201627. April 2016 / 4 Comments /

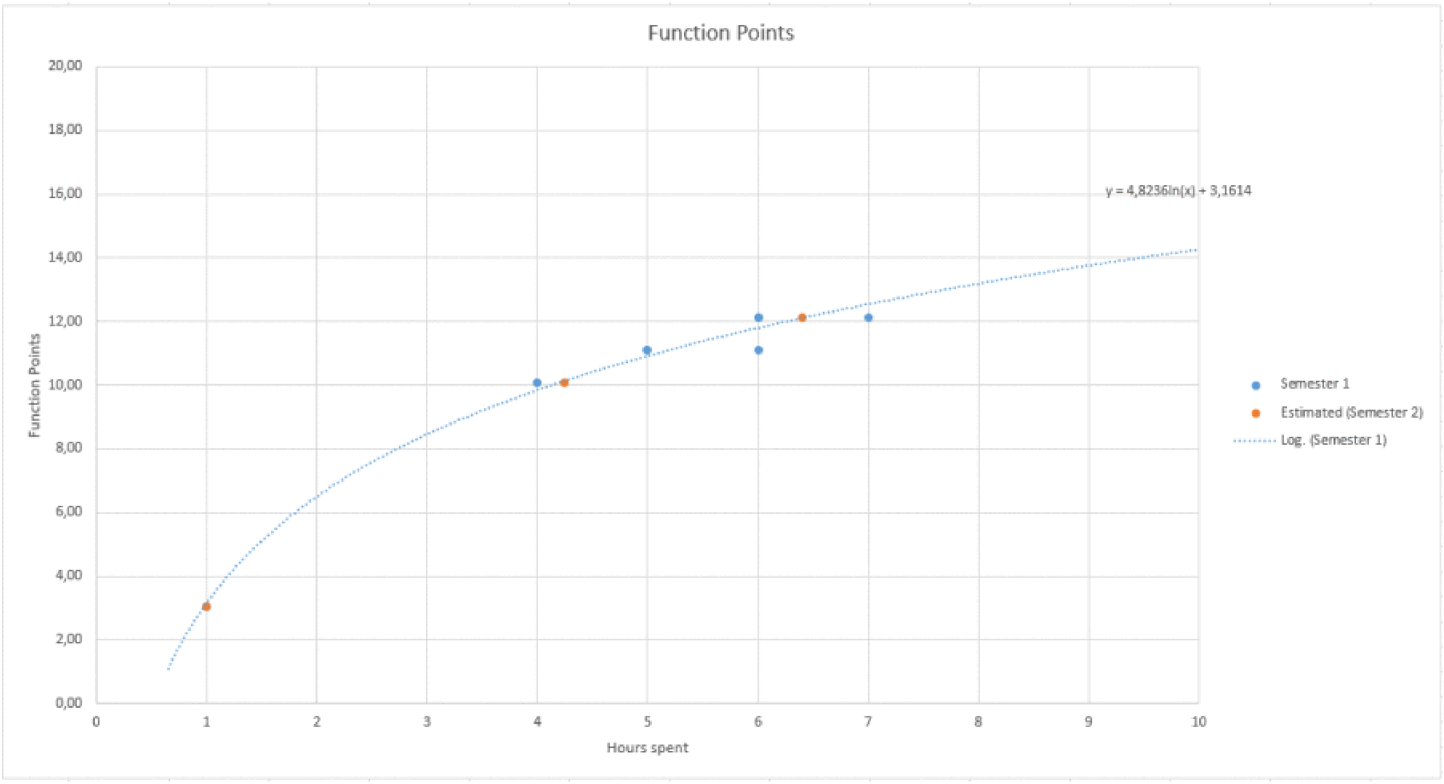
Function Points

Hi Fellows,

we have calculated the Function Points for our UseCases.
You can find the documents here (https://github.com/strebo/strebo/wiki).

~~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:



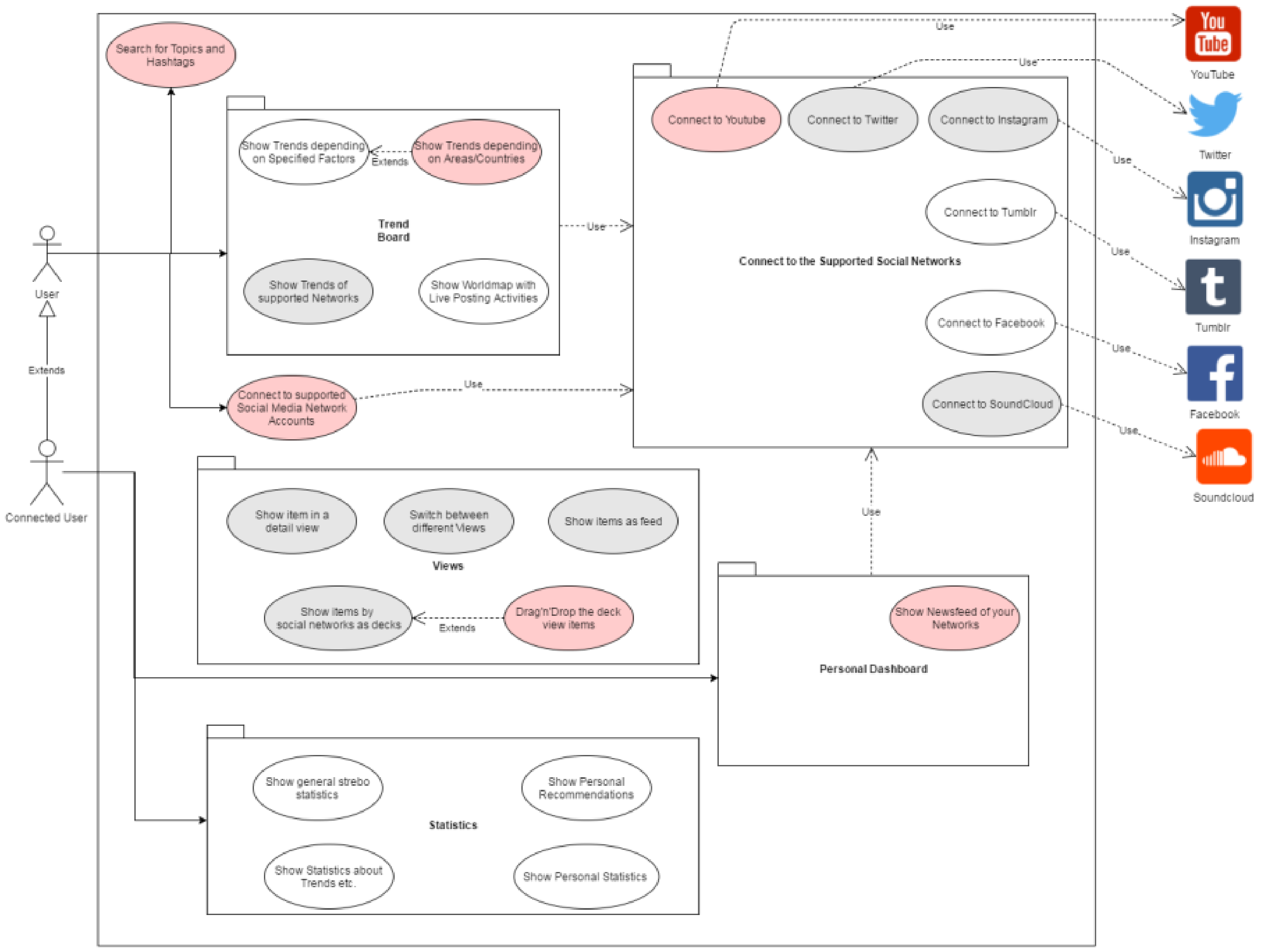
Best Greetings,
the strebo team.

18. April 201624. April 2016 / 3 Comments /

Semester II – Use Cases and Risk Management

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):


Search for Topics and Hashtags
Show Trends depending on Specified Factors
Extends
Show Trends depending on Areas/Countries
Trend Board
Show Trends of supported Networks
Show Worldmap with Live Posting Activities
Use
Connect to Youtube
Connect to Twitter
Connect to Instagram
Connect to Tumblr
Connect to the Supported Social Networks
Connect to Facebook
Connect to SoundCloud
Use
YouTube
Twitter
Instagram
Tumblr
Facebook
Soundcloud
User
Extends
Connected User
Connect to supported Social Media Network Accounts
Use
Show item in a detail view
Switch between different Views
Show items as feed
Views
Show items by social networks as decks
Extends
Drag'n'Drop the deck view items
Show Newsfeed of your Networks
Personal Dashboard
Show general strebo statistics
Show Personal Recommendations
Statistics
Show Statistics about Trends etc.
Show Personal Statistics

Here are the links to the new Use-Cases:

Search for Topics and Hashtags (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Search-for-Topics-and-Hashtags)
Show Newsfeed of your Networks (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Show-Newsfeed-of-your-Networks)Drag'n'Drop the deck view items (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Drag'n'Drop-the-deck-view-items)Show Trends depending on Areas/Countries (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Show-Trends-depending-on-Areas-Countries)Connect to Supported Social Media Network Accounts (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Connect-to-Supported-Social-Media-Network-Accounts)

We also created a "Use Case Effort Estimation" document which can be found here (https://github.com/strebo/strebo/wiki/Use-Case-Effort-Estimation).

And last but not least we now have a "Risk Management Plan", found here (https://github.com/strebo/strebo/wiki/Risk-Management).

An Overview for all our documents can be found here (https://github.com/strebo/strebo/wiki).

Best greetings,
the strebo team.

10. April 2016 / 4 Comments /

Midterm Project Report

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/ (http://strebo.net/).

The Sourcecode of our project (https://github.com/strebo/strebo) is available on GitHub as well as all of our documentations (https://github.com/strebo/strebo/wiki).

If you want to see how we organised our project have a look at our Gantt Chart (https://github.com/strebo/strebo/blob/master/documentation/Strebo_GanttChart.pdf) and our Scrum Borard (http://jira.it.dh-karlsruhe.de:8080/secure/RapidBoard.jspa?rapidView=9&projectKey=STREB&view=planning.nodetail).

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

your Team strebo

22. December 2015 / Leave a comment /

Gantt Chart

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 (https://github.com/strebo/strebo/blob/master/documentation/Strebo_GanttChart.pdf).

29. November 201530. November 2015 / 4 Comments /

Jira

We are now on Jira, which can be found here (http://jira.it.dh-karlsruhe.de:8080/projects/STREB/summary)! 🙂

We already started our first sprint, that will take two weeks, found here (http://jira.it.dh-karlsruhe.de:8080/secure/RapidBoard.jspa?projectKey=STREB&rapidView=9).

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

Best greetings
the strebo team.

23. November 201523. November 2015 / 6 Comments /

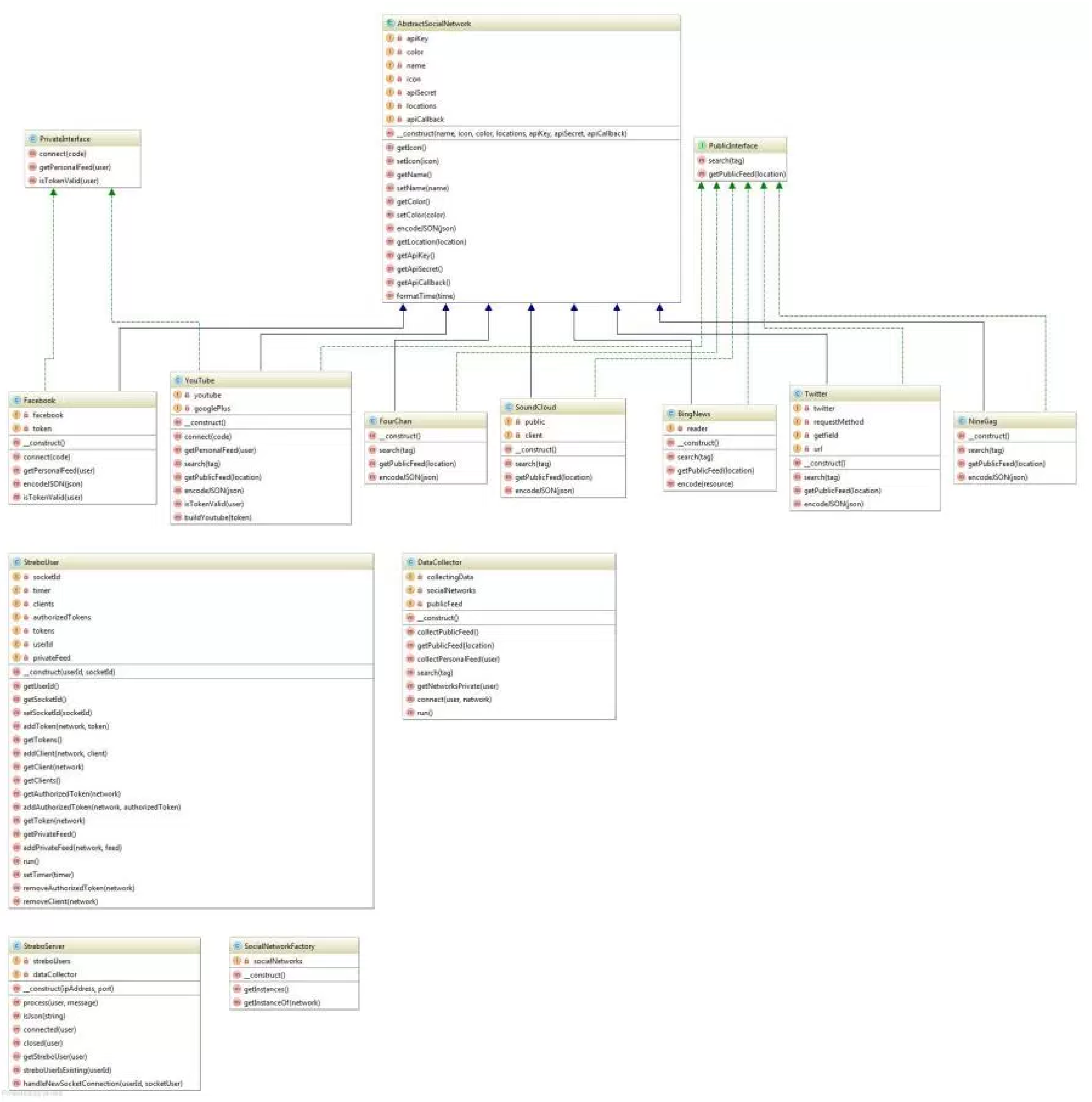
Software Architecture Document

We are happy to tell you that we moved our Documents to the GitHub Wiki and show you our new Software Architecture Document.

You will find it here: Software Architecture Document (https://github.com/strebo/strebo/wiki/Software-Architecture-Document)

16. November 201516. November 2015 / 6 Comments /

UML Class Diagram



Today we want to show you our first idea of an UML Class Diagram of our application. It is automatically generated with JetBrains PhpStorm. We will update this diagram if something changes.

The image above shows you an idea how our application works. There is a StreboServer that handles all user requests and a DataCollector that handles all connections to the social networks. To create instances of the single social networks the class SocialNetworkFactory is used. The different social networks (at the moment there are Twitter, BingNews, Facebook, 4Chan, 9Gag, SoundCloud and YouTube) inherits from AbstractSocialNetwork that have methods and attributes which every social network has. If a social network can provide a public feed it implements PublicInterface. If a social network provide a private feed it implements PrivateInterface. There can be networks that implements both interfaces, most will implement only one. The class StreboUser is used to administrate our Users as well as to save the access tokens of the single social networks if a user is loged in to a social network implementing the PrivateInterface.

8. November 201513. June 2016 / 4 Comments /

Gherkin: .feature files

We have updated out UC-1 and UC-2 documents and they now contain information about our .feature files!

The documents can be found here:

UC-1 (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Show-Trends-of-Supported-Networks)

UC-2 (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Connect-to-Supported-Social-Media-Network-Accounts)

2. November 201512. November 2015 / 4 Comments /

Use Case Specications

We specified two of our Use Cases.

The first one is "Show trends of supported networks" and can be found here:
Show trends of supported networks (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Show-Trends-of-Supported-Networks)

The second one is "Connect to supported Social Media Network Accounts" and can be found here:
Connect to Accounts (https://github.com/strebo/strebo/wiki/Use-Case-%E2%80%93-Connect-to-Supported-Social-Media-Network-Accounts)

We also added the links to our SRS, found here:
SRS (https://github.com/strebo/strebo/wiki/Software-Requirements-Specification)

25. October 201512. November 2015 / 4 Comments /

Software Requirements Specification

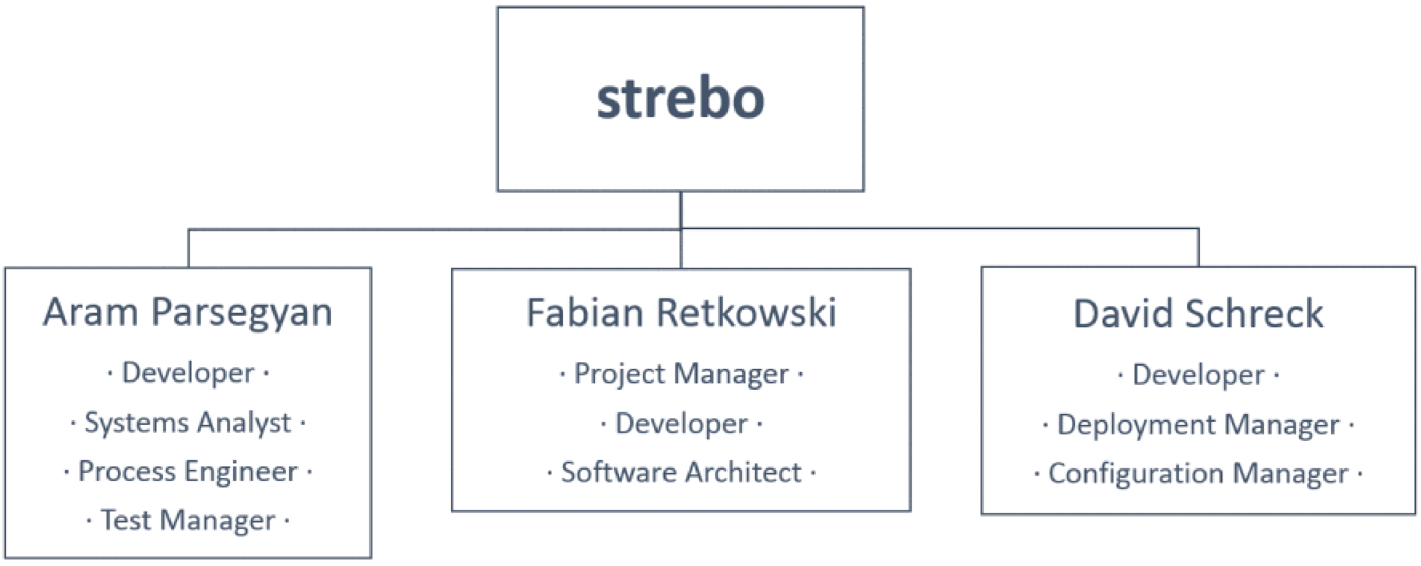
Our SRS documentation can be found here: SRS (https://github.com/strebo/strebo/wiki/Software-Requirements-Specification)

The document is also on the same repository on GitHub: https://github.com/strebo/strebo (https://github.com/strebo/strebo)

18. October 201512. November 2015 / 4 Comments /

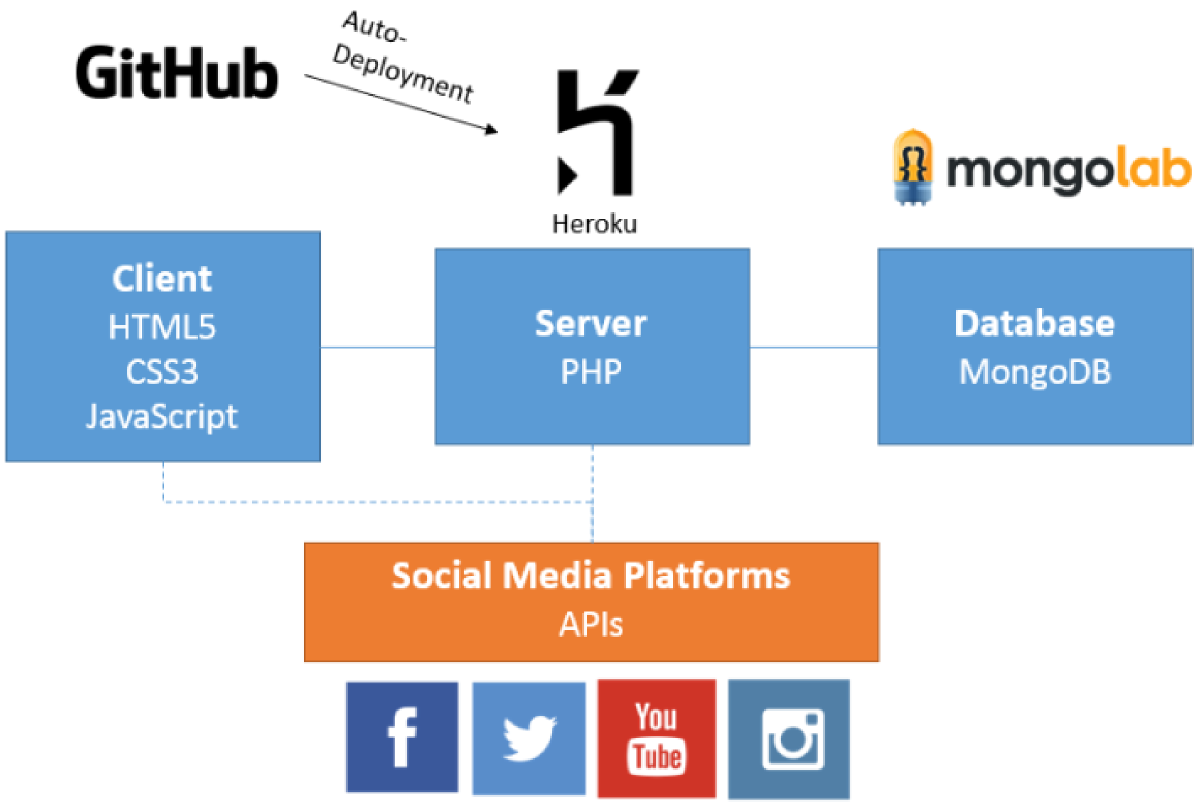
Team Roles and Technologies

The following scheme provides a rough overview about our team roles in RUP terminology:



For further explanation of these RUP terminologies you can look at http://www.ibm.com/developerworks/rational/library/apr05/crain/ (http://www.ibm.com/developerworks/rational/library/apr05/crain/).

Our application is web-based, so on client side, technologies like HTML5, CSS3 and JavaScript are used. Some Frameworks like jQuery will help us here. On serverside we use a PHP application which is hosted on Heroku and is conntected to a database (MongoDB) hosted by mongolab. The communication between serverside and clientside will be achieved with WebSockets (Ratchet framework). To get information from social media platforms we use several APIs, for example the Twitter API or Facebook API. The support of APIs and their functions are very different, so the integration of these APIs will be also very different and maybe it is also necessary to scrape data from websites.



(https://strebo.files.wordpress.com/2015/10/tech.png)

5. October 201522. December 2015 / 5 Comments /

Blog at WordPress.com. / The Shoreditch Theme.