Giteway 1.0

Technical specifications

Table of content

Deployment

Source code : *git://github.com/Couettos/giteway.git*

Deployed application: http://giteway.cloudfoundry.com

If you wish to deploy giteway in your local environment, please check the deployment instructions in the README file at the root of the project.

Technology stack

Spring 3.1: *Spring-core, Spring-webmvc, Spring-oxm, Spring-context, Spring-aop*

AOP: *Spring-AspectJ*

Logging: *SLF4J – Log4j*

Serializers/Deserializers: *Jackson, Castor*

Rest client: *Apache Commons-http*

Utilities: *LambdaJ-Hamcrest*

Caching: *Spring-EHCache*

Testing: *Junit, Mockito, Spring-test*

Front: *JSP, JSTL, JQuery, Plot*

Global Architecture

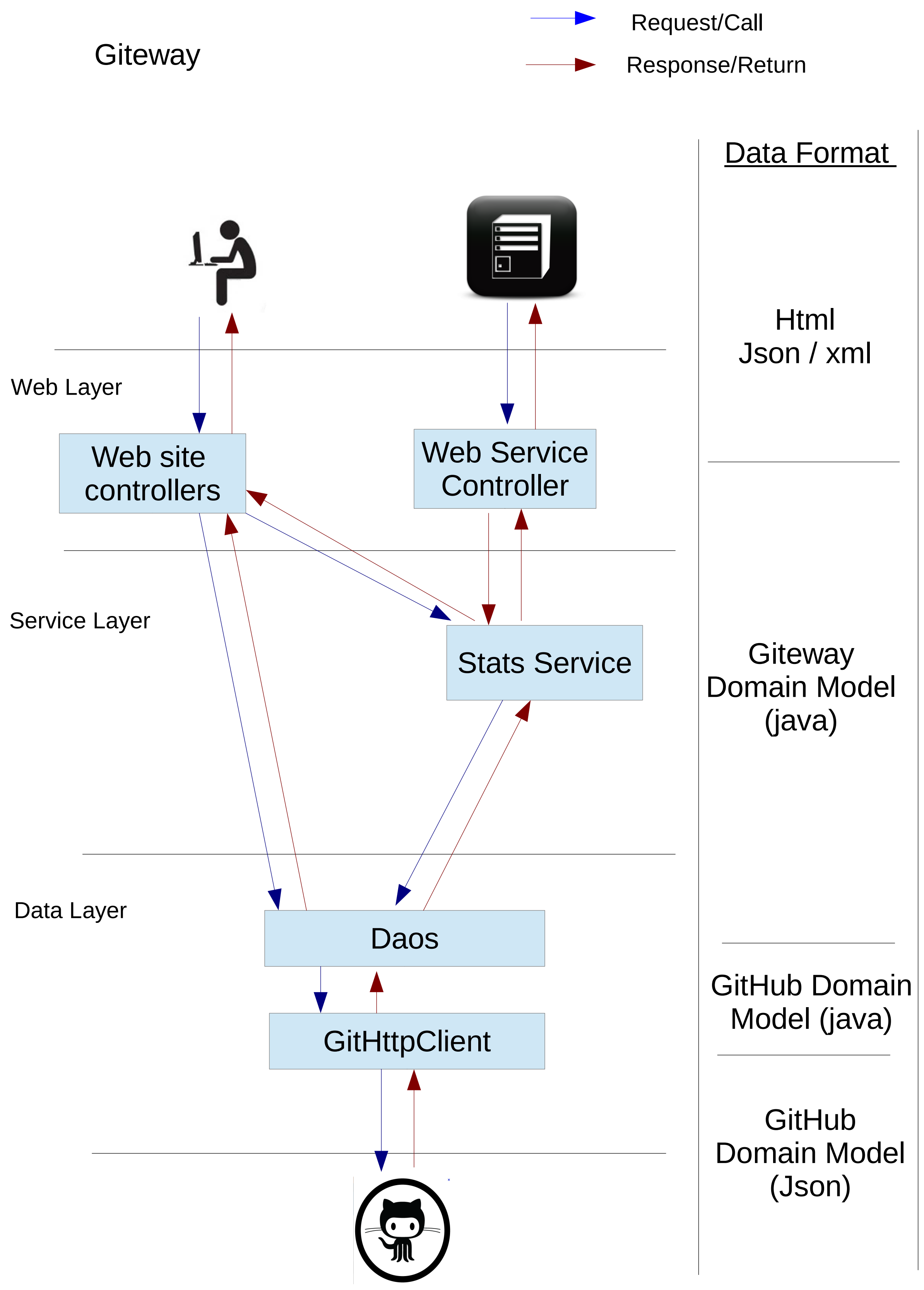


Figure 1 - Global architecture

Model

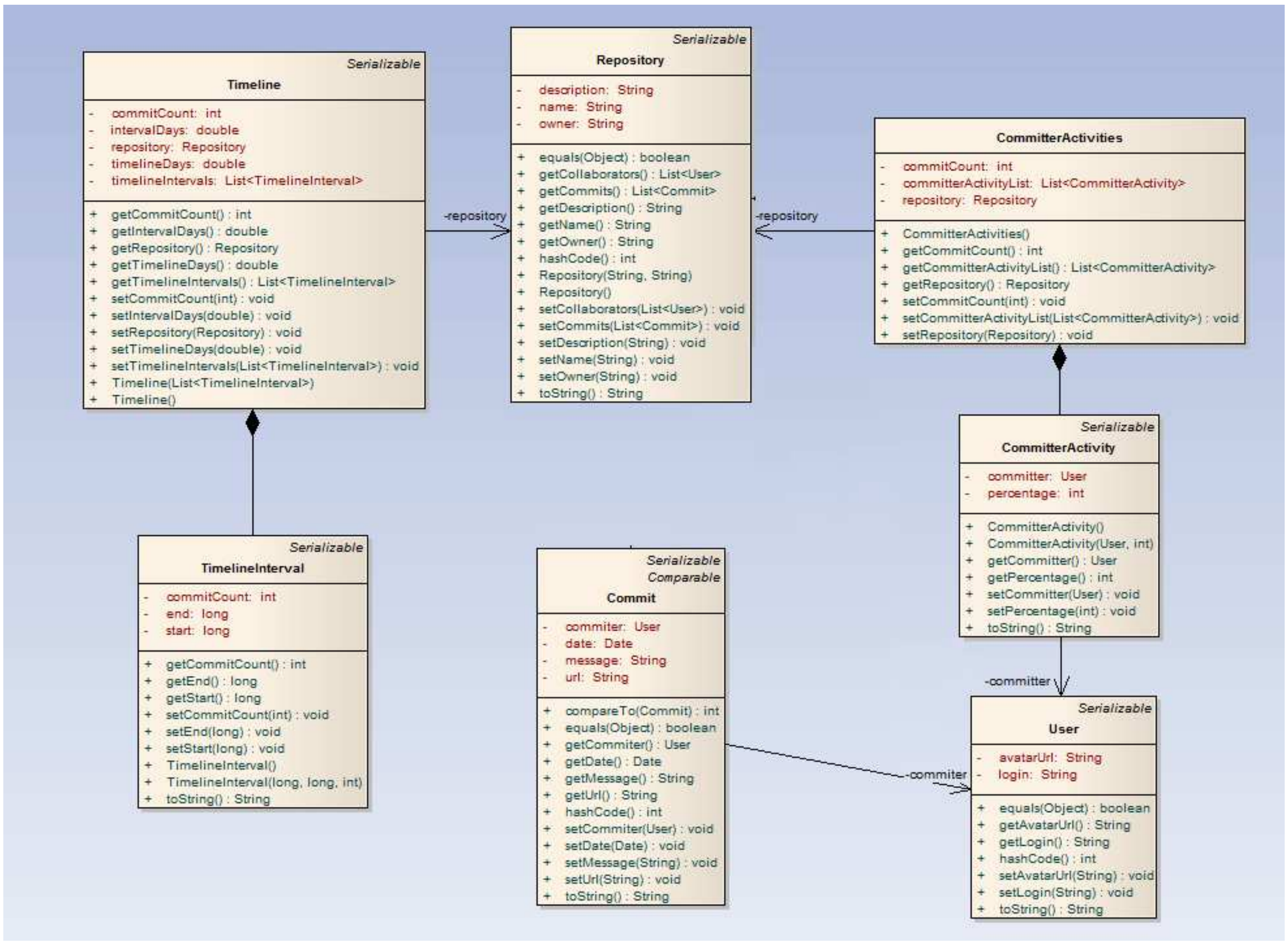

Serializable
Timeline
- commitCount: int
- intervalDays: double
- repository: Repository
- timelineDays: double
- timelineIntervals: List<TimelineInterval>
+ getCommitCount() : int
+ getIntervalDays() : double
+ getRepository() : Repository
+ getTimelineDays() : double
+ getTimelineIntervals() : List<TimelineInterval>
+ setCommitCount(int) : void
+ setIntervalDays(double) : void
+ setRepository(Repository) : void
+ setTimelineDays(double) : void
+ setTimelineIntervals(List<TimelineInterval>) : void
+ Timeline(List<TimelineInterval>)
+ Timeline()
-repository
Serializable
Repository
- description: String
- name: String
- owner: String
+ equals(Object) : boolean
+ getCollaborators() : List<User>
+ getCommits() : List<Commit>
+ getDescription() : String
+ getName() : String
+ getOwner() : String
+ hashCode() : int
+ Repository(String, String)
+ Repository()
+ setCollaborators(List<User>) : void
+ setCommits(List<Commit>) : void
+ setDescription(String) : void
+ setName(String) : void
+ setOwner(String) : void
+ toString() : String
-repository
CommitterActivities
- commitCount: int
- committerActivityList: List<CommitterActivity>
- repository: Repository
+ CommitterActivities()
+ getCommitCount() : int
+ getCommitterActivityList() : List<CommitterActivity>
+ getRepository() : Repository
+ setCommitCount(int) : void
+ setCommitterActivityList(List<CommitterActivity>) : void
+ setRepository(Repository) : void
Serializable
CommitterActivity
- committer: User
- percentage: int
+ CommitterActivity()
+ CommitterActivity(User, int)
+ getCommitter() : User
+ getPercentage() : int
+ setCommitter(User) : void
+ setPercentage(int) : void
+ toString() : String
-committer
Serializable
TimelineInterval
- commitCount: int
- end: long
- start: long
+ getCommitCount() : int
+ getEnd() : long
+ getStart() : long
+ setCommitCount(int) : void
+ setEnd(long) : void
+ setStart(long) : void
+ TimelineInterval()
+ TimelineInterval(long, long, int)
+ toString() : String
Serializable
Comparable
Commit
- commiter: User
- date: Date
- message: String
- url: String
+ compareTo(Commit) : int
+ equals(Object) : boolean
+ getCommiter() : User
+ getDate() : Date
+ getMessage() : String
+ getUrl() : String
+ hashCode() : int
+ setCommiter(User) : void
+ setDate(Date) : void
+ setMessage(String) : void
+ setUrl(String) : void
+ toString() : String
-commiter
Serializable
User
- avatarUrl: String
- login: String
+ equals(Object) : boolean
+ getAvatarUrl() : String
+ getLogin() : String
+ hashCode() : int
+ setAvatarUrl(String) : void
+ setLogin(String) : void
+ toString() : String

Figure 2 - The model

The model is loosely coupled from GitHub domain model. This architectural choice has been motivated by the inconsistency of GitHub DM. For example, the repositories returned by the keyword search and by the single request do not have the same attributes.

GitHub Objects are considered as Value Objects or DTOs.

Advantages:

- Clearer architecture and object responsibilities
- Reduce the complexity of GitHub Domain Model
- Isolates the impacts of attribute modification

Drawbacks:

- Glue code
- Transfer Object instantiation per request
- More code to write if a new attribute has to be mapped

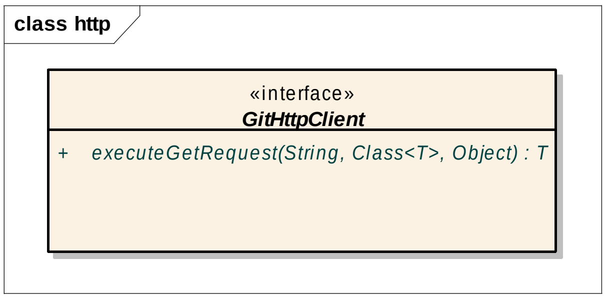
Data Access Layer

The data access layer requests data from GitHub API. It is an abstraction layer over HTTP and GitHub DM.

The data access tier provides two mechanisms:

1st: Request Github over HTTP, map it to DTOs

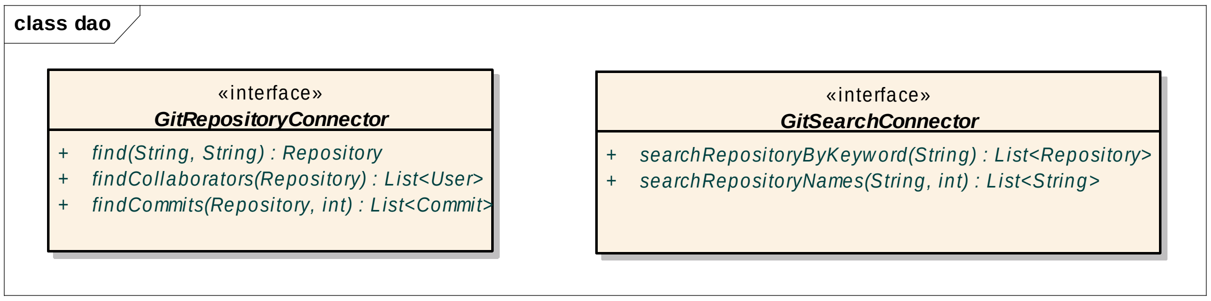
The data is accessible over HTTP in a json format. The interface GitHttpClient requests the data and converts it into a set of mapped class representing the GitHub DM (DTOs).



Note : GitHttpClient encapsulates a DefaultHttpClient thread safe singleton object which is instantiated with a PoolingClientConnectionManager. This approach enables DefaultHttpClient to reuse connections from the pool. Between to requests, the connections are kept alive. This is of great interest as the SSL handshake can take a long time.

2nd: Map GitHub DM to Giteway DM

The DAOs receive DTOs from GitHttpClient and convert those into Giteway domain model.

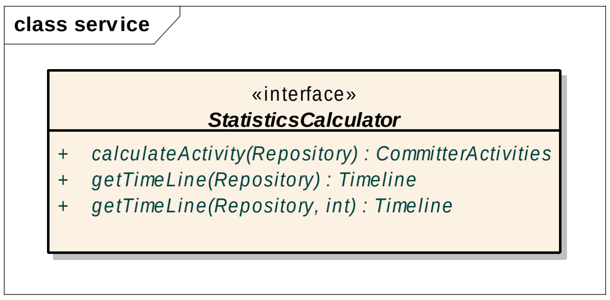

class dao
«interface»
GitRepositoryConnector
+ find(String, String) : Repository
+ findCollaborators(Repository) : List<User>
+ findCommits(Repository, int) : List<Commit>
«interface»
GitSearchConnector
+ searchRepositoryByKeyword(String) : List<Repository>
+ searchRepositoryNames(String, int) : List<String>

Service Layer

The service layer encapsulates the business logic of Giteway. The component calculate statistics. The data is accessed through the data access layer.

The API:

- Defines the timeline
- Calculates the user's collaboration

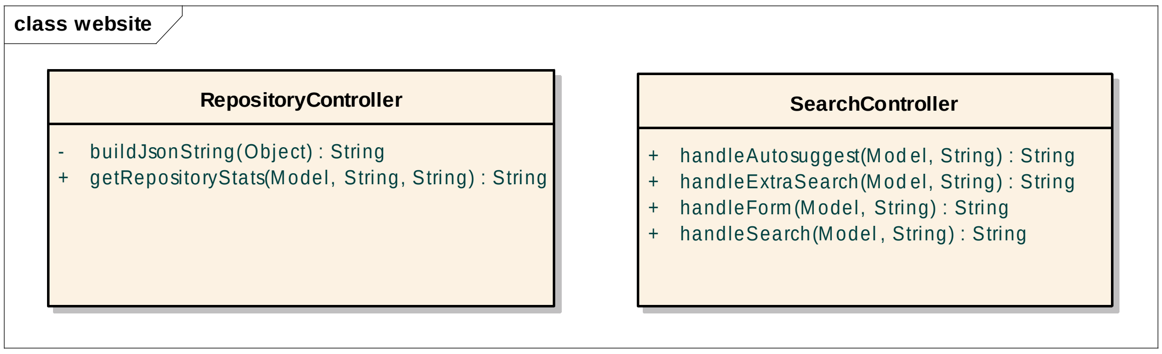

class service
«interface»
StatisticsCalculator
+ calculateActivity(Repository) : CommitterActivities
+ getTimeLine(Repository) : Timeline
+ getTimeLine(Repository, int) : Timeline

Presentation Layer

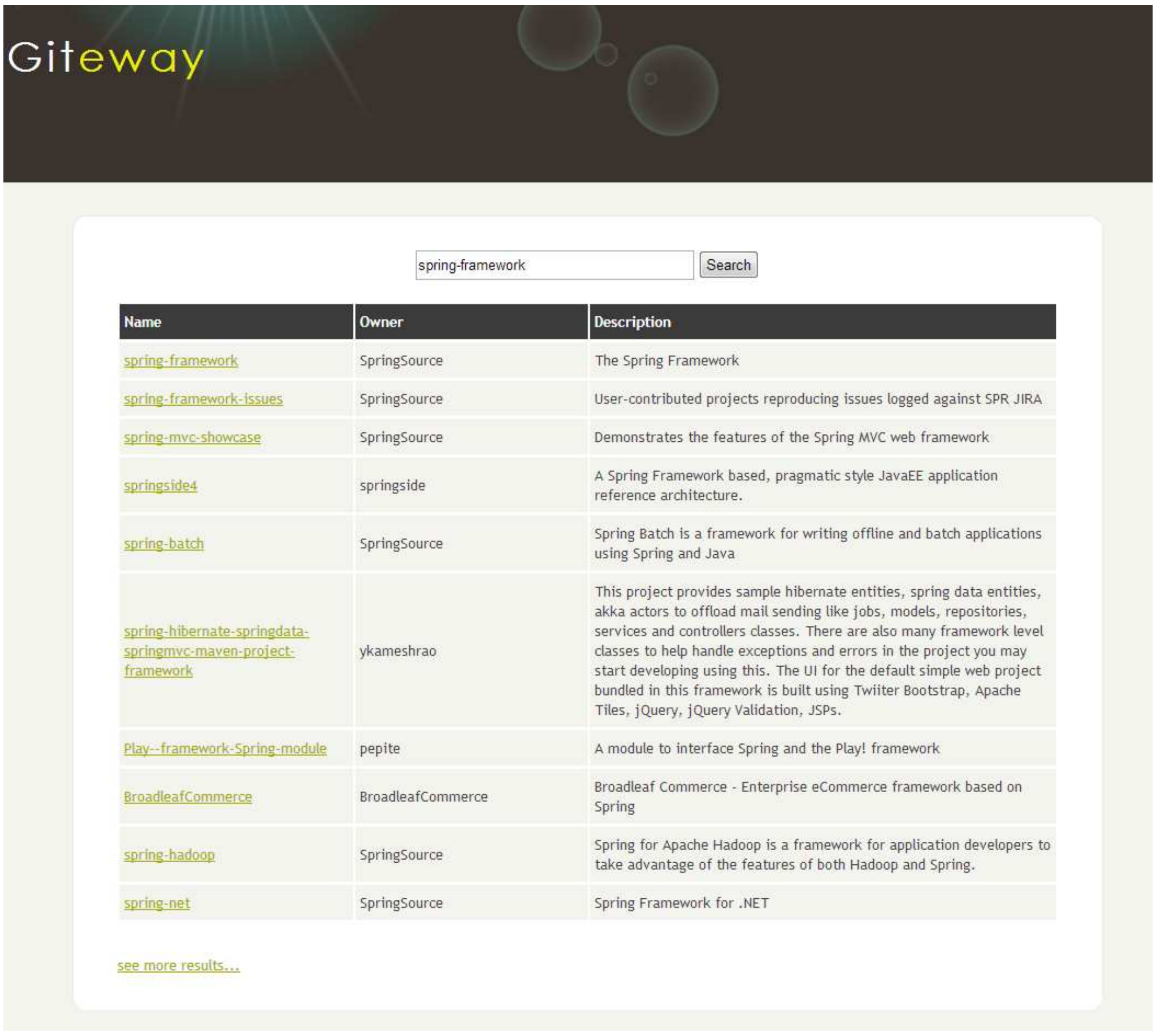
The presentation tier is divided in two parts: the website and the restful WS.

1. ***The Website***

Two Spring webmvc controllers handles the two views.


class website
RepositoryController
- buildJsonString(Object) : String
+ getRepositoryStats(Model, String, String) : String
SearchController
+ handleAutosuggest(Model, String) : String
+ handleExtraSearch(Model, String) : String
+ handleForm(Model, String) : String
+ handleSearch(Model, String) : String

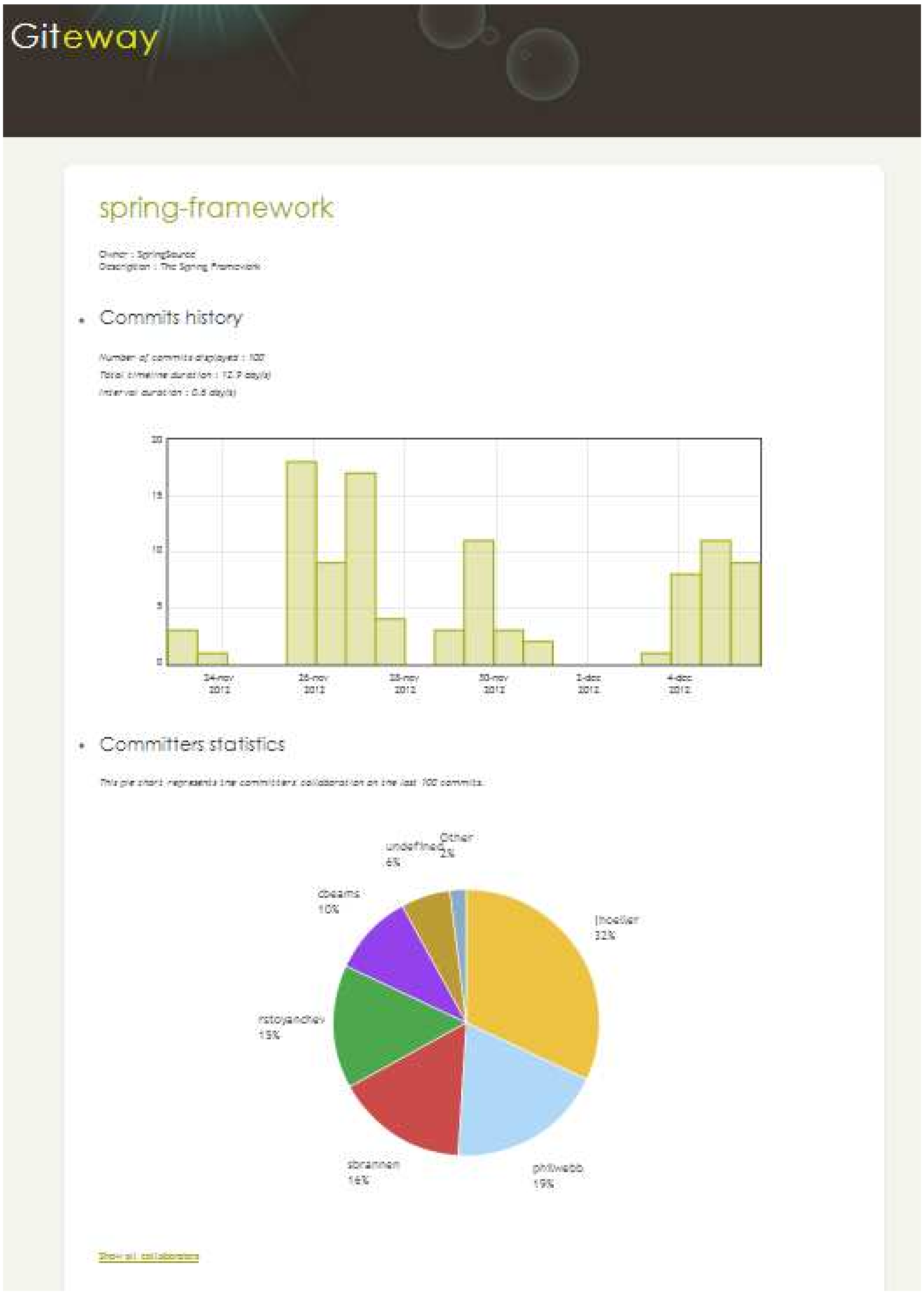
The website is divided in two JSP.



Name	Owner	Description
spring-framework	SpringSource	The Spring Framework
spring-framework-issues	SpringSource	User-contributed projects reproducing issues logged against SPR JIRA
spring-mvc-showcase	SpringSource	Demonstrates the features of the Spring MVC web framework
springside4	springside	A Spring Framework based, pragmatic style JavaEE application reference architecture.
spring-batch	SpringSource	Spring Batch is a framework for writing offline and batch applications using Spring and Java
spring-hibernate-springdata-springmvc-maven-project-framework	ykameshrao	This project provides sample hibernate entities, spring data entities, akka actors to offload mail sending like jobs, models, repositories, services and controllers classes. There are also many framework level classes to help handle exceptions and errors in the project you may start developing using this. The UI for the default simple web project bundled in this framework is built using Twiiter Bootstrap, Apache Tiles, jQuery, jQuery Validation, JSPs.
Play--framework-Spring-module	pepite	A module to interface Spring and the Play! framework
BroadleafCommerce	BroadleafCommerce	Broadleaf Commerce - Enterprise eCommerce framework based on Spring
spring-hadoop	SpringSource	Spring for Apache Hadoop is a framework for application developers to take advantage of the features of both Hadoop and Spring.
spring-net	SpringSource	Spring Framework for .NET

see more results...

Figure 3 - The search view



spring-framework

Owner : SpringSource
Description : The Spring Framework

- Commits history

Number of commits displayed : 100
Total timeline duration : 12.9 day(s)
Interval duration : 0.6 day(s)

- Committers statistics

This pie chart represents the committers' collaboration on the last 100 commits.

Figure 4 - The statistic view

The views are based on the following technologies/Framework:

- JSP / JSTL
- JQuery

- JQueryUI (autosuggest)
- Plot (charts)
- Ajax (autosuggest and “search extra repositories”)

Note: The Website has been designed with a restful approach. Every page is bookmarkable as URLs define which information is displayed. No session or history is held on the server.

2. *The restful web service*

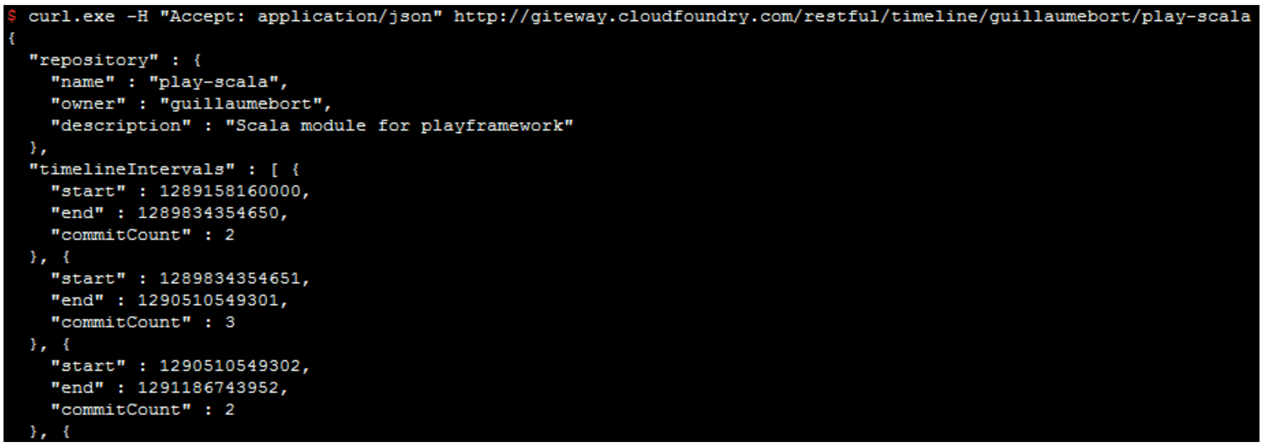
Giteway also exposes a restful WS which provides statistics.

Services:

1. The timeline data at : *<app>/restful/repos /<owner>/<repositoryName>/timeline*
2. The committers’ impact at : *<app>/restful /repos/<owner>/<repositoryName>/activity*

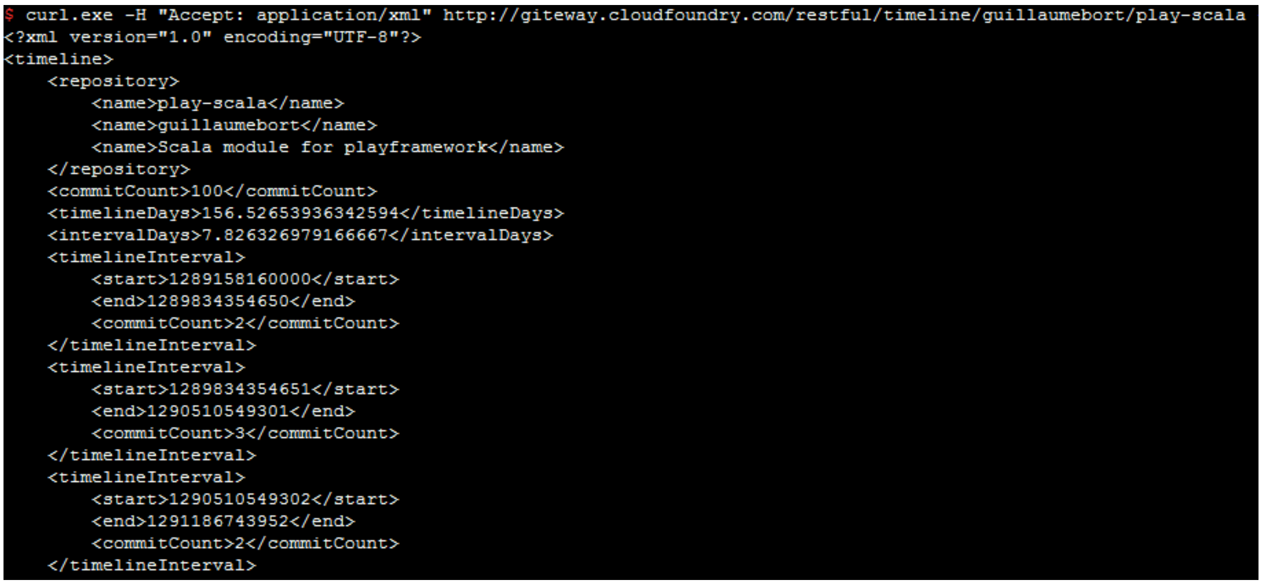
The data can be returned in two different formats by specifying a different header value:

1. Json (Accept: application/json)
2. Xml (Accept: application/xml)



```
$ curl.exe -H "Accept: application/json" http://giteway.cloudfoundry.com/restful/timeline/guillaumebort/play-scala
{
  "repository" : {
    "name" : "play-scala",
    "owner" : "guillaumebort",
    "description" : "Scala module for playframework"
  },
  "timelineIntervals" : [ {
    "start" : 1289158160000,
    "end" : 1289834354650,
    "commitCount" : 2
  }, {
    "start" : 1289834354651,
    "end" : 1290510549301,
    "commitCount" : 3
  }, {
    "start" : 1290510549302,
    "end" : 1291186743952,
    "commitCount" : 2
  }, {
```

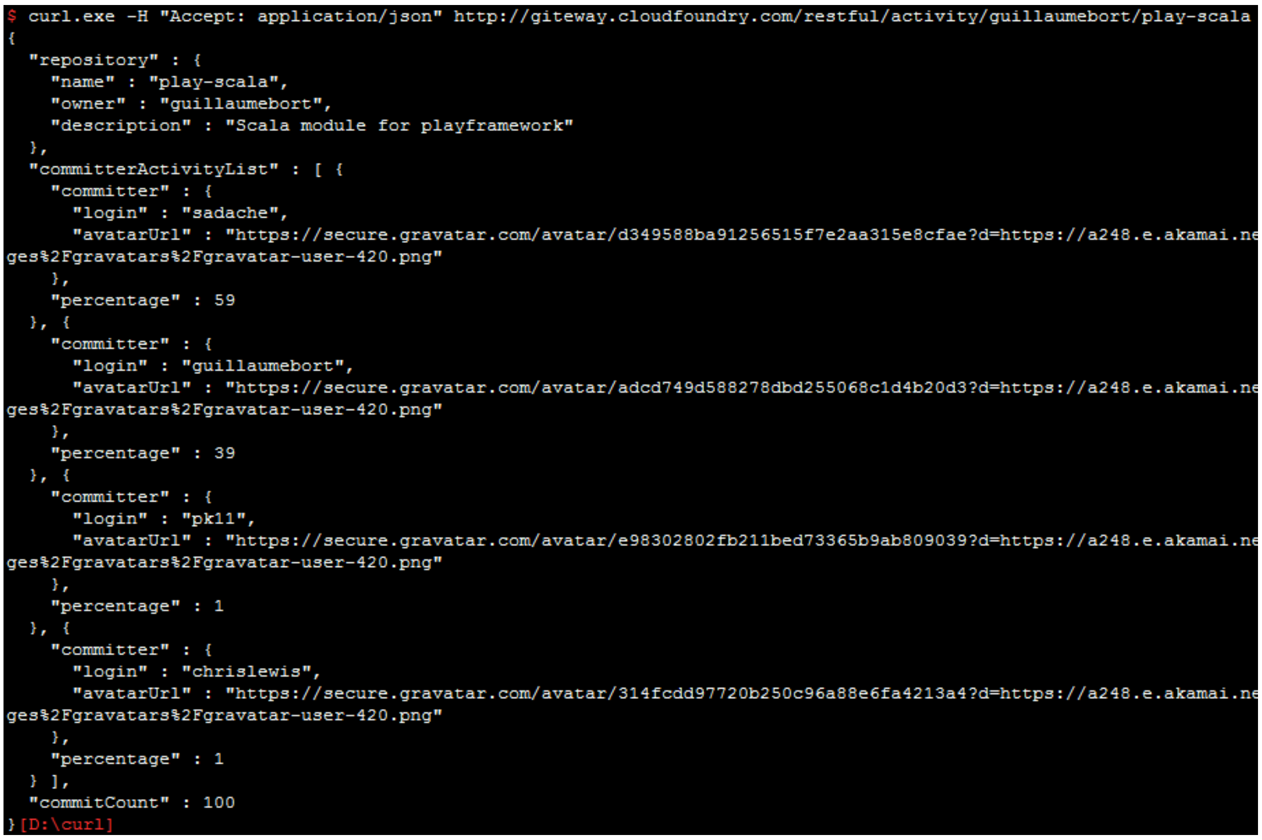
Figure 5 - Timeline json



```
$ curl.exe -H "Accept: application/xml" http://giteway.cloudfoundry.com/restful/timeline/guillaumebort/play-scala
<?xml version="1.0" encoding="UTF-8"?>
<timeline>
    <repository>
        <name>play-scala</name>
        <name>guillaumebort</name>
        <name>Scala module for playframework</name>
    </repository>
    <commitCount>100</commitCount>
    <timelineDays>156.52653936342594</timelineDays>
    <intervalDays>7.826326979166667</intervalDays>
    <timelineInterval>
        <start>1289158160000</start>
        <end>1289834354650</end>
        <commitCount>2</commitCount>
    </timelineInterval>
    <timelineInterval>
        <start>1289834354651</start>
        <end>1290510549301</end>
        <commitCount>3</commitCount>
    </timelineInterval>
    <timelineInterval>
        <start>1290510549302</start>
        <end>1291186743952</end>
        <commitCount>2</commitCount>
    </timelineInterval>
```

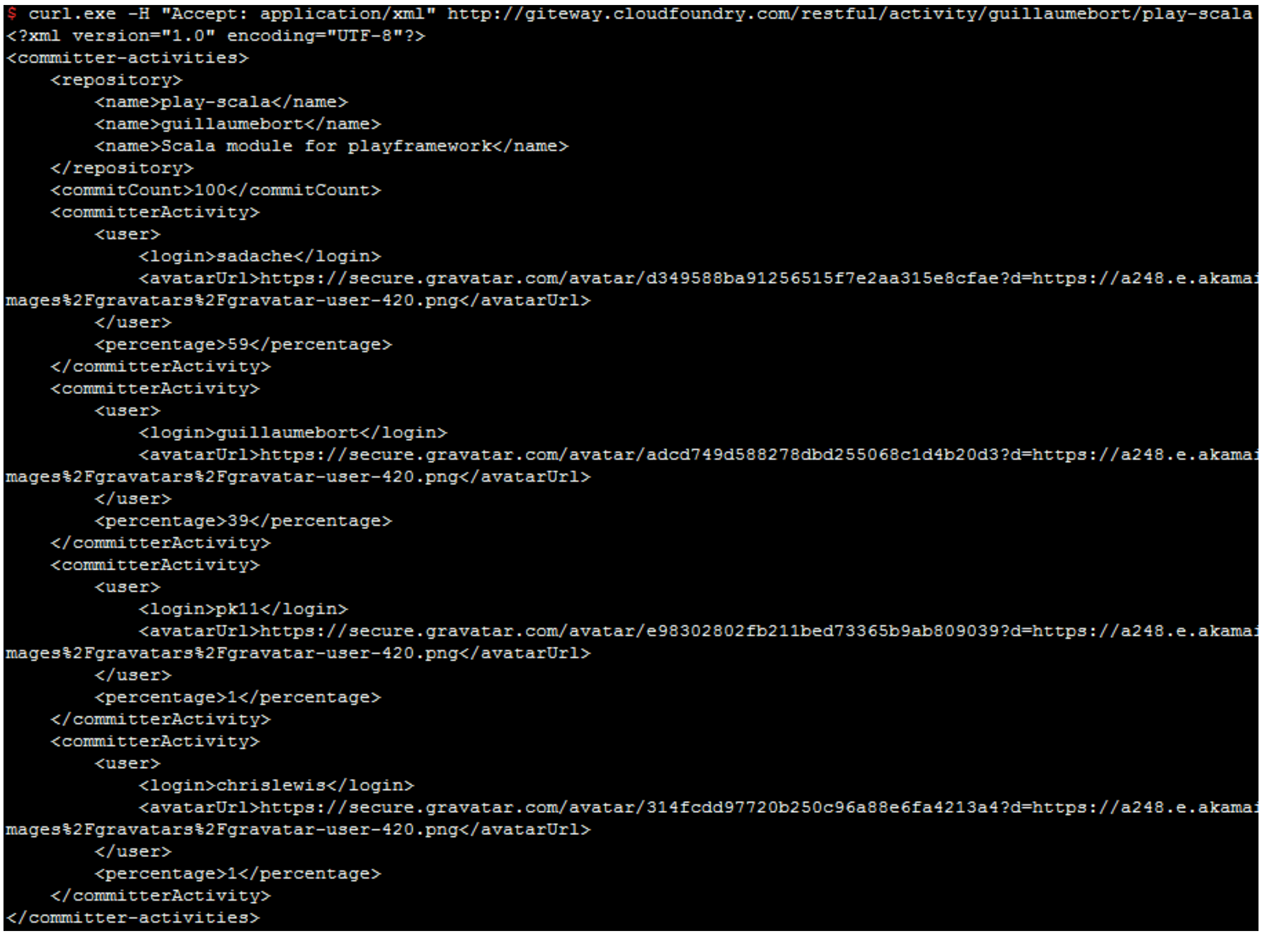
Figure 6 - timeline xml

Committer's impact :



```
$ curl.exe -H "Accept: application/json" http://giteway.cloudfoundry.com/restful/activity/guillaumebort/play-scala
{
  "repository" : {
    "name" : "play-scala",
    "owner" : "guillaumebort",
    "description" : "Scala module for playframework"
  },
  "committerActivityList" : [ {
    "committer" : {
      "login" : "sadache",
      "avatarUrl" : "https://secure.gravatar.com/avatar/d349588ba91256515f7e2aa315e8cfae?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 59
  }, {
    "committer" : {
      "login" : "guillaumebort",
      "avatarUrl" : "https://secure.gravatar.com/avatar/adcd749d588278dbd255068c1d4b20d3?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 39
  }, {
    "committer" : {
      "login" : "pk11",
      "avatarUrl" : "https://secure.gravatar.com/avatar/e98302802fb211bed73365b9ab809039?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 1
  }, {
    "committer" : {
      "login" : "chrislewis",
      "avatarUrl" : "https://secure.gravatar.com/avatar/314fcdd97720b250c96a88e6fa4213a4?d=https://a248.e.akamai.ne
ges%2Fgravatars%2Fgravatar-user-420.png"
    },
    "percentage" : 1
  } ],
  "commitCount" : 100
}[D:\curl]
```

Figure 7 – Committers' impact json



```
$ curl.exe -H "Accept: application/xml" http://giteway.cloudfoundry.com/restful/activity/guillaumebort/play-scala
<?xml version="1.0" encoding="UTF-8"?>
<committer-activities>
    <repository>
        <name>play-scala</name>
        <name>guillaumebort</name>
        <name>Scala module for playframework</name>
    </repository>
    <commitCount>100</commitCount>
    <committerActivity>
        <user>
            <login>sadache</login>
            <avatarUrl>https://secure.gravatar.com/avatar/d349588ba91256515f7e2aa315e8cfae?d=https://a248.e.akama
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>59</percentage>
    </committerActivity>
    <committerActivity>
        <user>
            <login>guillaumebort</login>
            <avatarUrl>https://secure.gravatar.com/avatar/adcd749d588278dbd255068c1d4b20d3?d=https://a248.e.akama
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>39</percentage>
    </committerActivity>
    <committerActivity>
        <user>
            <login>pk11</login>
            <avatarUrl>https://secure.gravatar.com/avatar/e98302802fb211bed73365b9ab809039?d=https://a248.e.akama
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>1</percentage>
    </committerActivity>
    <committerActivity>
        <user>
            <login>chrislewis</login>
            <avatarUrl>https://secure.gravatar.com/avatar/314fcdd97720b250c96a88e6fa4213a4?d=https://a248.e.akama
mages%2Fgravatars%2Fgravatar-user-420.png</avatarUrl>
        </user>
        <percentage>1</percentage>
    </committerActivity>
</committer-activities>
```

Figure 8 - Committers' impact xml

Caching

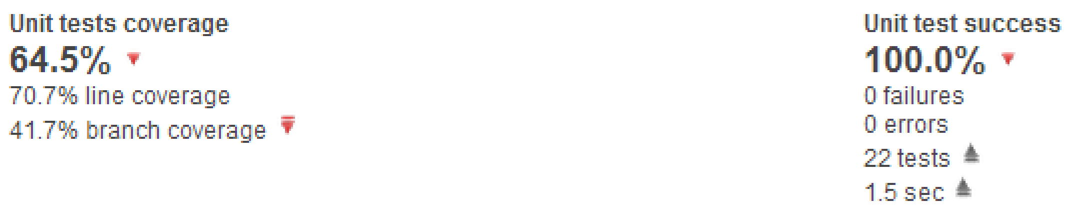
Giteway accesses Github data remotely. Because the communication can be slow, a cache has been set up on the DAOs method returns. Spring provides an easy, non-intrusive way of handling caching without any coding. The spring-ehcache approach has been chosen.

AOP

AOP handle the logging with an around advice weaved on all methods of the application except the front methods. Spring-AspectJ approach has been used.

Testing - Check-Style - Code coverage

Every tier of giteway is unit tested.

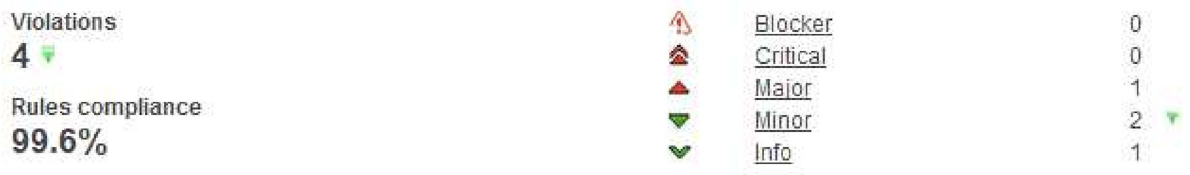

Unit tests coverage
64.5% ▼
70.7% line coverage
41.7% branch coverage ▼

Unit test success
100.0% ▼
0 failures
0 errors
22 tests ▲
1.5 sec ▲

Figure 9 - Unit test analyse

The components have been isolated from each other by mocking the dependencies with mockito. Spring-test has been used for the tooling but no integration tests are performed.

The application code has been analyzed with Sonar.


Violations
4 ▼

Rules compliance
99.6%

Blocker	0	
Critical	0	
Major	1	
Minor	2	▼
Info	1	

Figure 10 - CheckStyle