Introduction to Software Engineering

First and Second Iteration - Trend It

Filipe Gonçalves, 98083 - Team Manager
Pedro Lopes, 97827 - DevOps
Vicente Costa, 98515 - Architect
João Borges, 98155 - Product Owner

Universidade de Aveiro - 2021/2022

First Iteration

In this first iteration, we want to have a skeleton of the application, and define the key ideas for the project

Personas and User Stories

Made three personas, each one with a case scenario and a user story associated: One persona was made for lazure utilization of the application (browsing and knowing what's happening around the world and which trends to follow), while the other two, for business purposes (management of trends and/or knowing what's trending with the intention of creating a new business).

Example of user story:

- As an aspirant fashionist, Anabela would like to cope up with the latest trends in the fashion world with the minimum time possible.
- While creating an account on the app, she put as a topic of interest "fashion", and with that, on her home page, the most popular topics with the that hashtag should appear
- She can filter the fashion topics to only "carhartt", for example, so that she can see what is trending on her favorite brand

Backlog

In software development, a product backlog is a prioritized list of deliverables (such as new features, bugfixes or hotfixes) that should be implemented as part of a project or product development.
It's usually a way for team members to organize themselves in a way that two members are doing the same thing or someone is doing something more than once.
This tool is also great to maintain an overview of how the project is going, since for every entry in the backlog we can assign points that should reflect the time/complexity of the work developed.
It's also great to include ideas in the backlog even if they don't get implemented. When thinking about backlogging we should think about it as a wishlist where everyone can review everyone's work in order to move forward with the product.
There are many tools to help us develop a backlog, for now we will be using the project tab in github alongside with repository branching to fix bugs or misconceptions, and to report issues developed by the team members.

Project

In the project Trend It we did the index, home, login and register page, in accordance with the user stories defined before.

The index page works as a welcome to the webpage, which will redirect to login.html for login purposes, or in case the user doesn't have an account, to the register.html. With the login/register made, it will show the home page which will be used to show both tweets and statistics.

Because it's still a prototype, it's still too early to know if we will add more/remove some.

Architect Notebook

Software project management and comprehension tool

Maven: maven is a build tool, it is also a software project management and comprehension tool, based on the concept of a project object model (POM), which should make things easier in organization and compilation.

Web java-based application tool

Java Spring boot
Java Spring
Spring Initializr
Spring Tools
Thymeleaf

Each one of these, will make it easier to acquire a RESTful application, with both a normal Java Spring Controller and a Java Spring Boot REST Controller, Model and Views with the help of Thymeleaf and Spring Initializr to help us build the project

Container

Docker: Using docker will help us create a space for both the database and the project to run, whilst not being completely on our computer.

Database

Requisites:

- Database oriented to lines
- Very high performance
- High availability
- Horizontal scalability
- Easy filter of content

MySql

Characteristics	Positive or Negative aspects
Oriented to lines	Positive
Great understanding from the developers	Positive
Scale well if configured correctly	Positive/Negative
Vertical scalability	Negative
Very high availability	Positive

MongoDB

Characteristics	Positive or Negative aspects
Document database	Positive/Negative
Vertical and horizontal scalability and high availability (with sharding and replica sets)	Positive
Great for transactional, search, analytics	Positive
Performance limitations can happen due to inadequate or inappropriate indexing strategies, or as a consequence of poor schema design patterns	Negative
The developers have experience and understanding of mongodb	Positive/Negative

Cassandra

Characteristics	Positive or Negative aspects
Oriented to column	Positive
distributed (Maybe overkill)	Positive
horizontal scalability	Positive
high performance and availability	Positive
developers of the group are less experienced with this tool	Negative

Looking at the pros and cons of all types of databases, we can say that, for this project, mysql is the worst option, as the application does not require a relational database, with complex schema. From the last two options, we decided to choose MongoDB because of the experience the group members have in comparison with Cassandra, and because of the document-oriented way the database is made. While column-oriented database could help, working with tweets, will make us get the whole tweet object and not a single column from the database.

Twitter Bot

languages	Java	Python
Useful Sites	https://gist.github.com/dueyfinster/2469810	https://realpython.com/twitter-bot-python-tweepy/
	https://medium.com/@SeloSlav/how-to-make-a-scary-russian-twitter-bot-with-java-b7b62768a3ac	https://auth0.com/blog/how-to-make-a-twitter-bot-in-python-using-tweepy/

The twitter bot will be made in java because it will help in simplifying the database connection, with the creation of another maven + spring boot project application.

Second Iteration

In this second iteration, we started working on the concrete application, building the twitter bot to get dynamic data and containerize the project in docker.

Twitter Bot

The twitter bot application is implemented in java using maven and spring boot. It contains only a model, the Tweet class, which has an id, a tag, a timestamp and a description. It also contains a simple controller that can create and delete a tweet and find tweets by id or by tag.

To handle exceptions, we also created simple classes like the one we used on the practical classes (ErrorDetails, GlobalExceptionHandler and ResourceNotFound). For now we only use the ResourceNotFound in the controller.

For the TweetRepository, we decided to store the data with mongodb so our TweetRepository is a mongo repository and it has a function to find tweets by tag.

Finally, our TwitterBotApp uses the library twitter4j and for now it makes two queries using the twitter API to get a tweet with a certain id and tweets from "hyperlegen", "KingJames" and "DailyNASA"

Twitter API

For now we are using 3 endpoints of the Twitter API, which give us the possibility to search in the last 7 days: all tweets matching a certain query; the number of tweets posted daily matching a certain query; twitter trends for a certain country, city or even worldwide.

In order for us to interact with the api we are using Retrofit2 and Gson.
This allows us to properly authenticate to certain endpoints that must use a Bearer Token, which Twitter4j2 doesn't support.

Container

As of the containers we used Docker, docker allows simple deployment and ensures that every member in the team is capable of running the app independent of the system they are using.

So far we have two containers, one for the Spring Boot web application, called Trendit, that will have a database connection to a mongodb server, and the other one for the database service. In order for all containers to communicate we need them to be in the same network and for that we use docker-compose, which is a simpler way of deploying a multi container application, allowing the user to define every service they need in a single file, instead of using a bash script or a batch to initialize every component in the right order and attaching them to the right network manually, since docker-compose offers the possibility of configuring each one of these aspects.

Web Application

The application contains, as of now, two Controllers, with Spring Boot annotation: The first, ViewController, with @Controller, works as the base for the html Views and Models (User and Tweet), so we can have access to the web application; The second, ApiCOntroller, with @RestController, works as the middleman for the CRUD actions (Create, Read, Update, Delete), where we can insert both users and tweets, see them, etc.

We also have two repositories, one for each model, which will save us the users/tweets in our database, find our users by username and our tweets by id (per example).

For the Models, we have a User Model, to represent everyone who wishes to register/login on the website, we save the username and the password when registering and compare them with the inputs obtained when signing in. We also have a Tweet Model, to represent our data, and it will be mostly shown in the home page, after the user signs in; all tweets which have the same "hashtags" that the User also has, will be shown, else they won't.

Bibliography

First Iteration:

https://www.w3schools.com/w3css/w3css_templates.asp

https://www.mongodb.com/

https://cassandra.apache.org/_/index.html

https://www.mysql.com

https://www.perforce.com/resources/hns/agile-product-backlog-basics

Second Iteration:

https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query

https://developer.twitter.com/en/docs/twitter-api/tweets/counts/api-reference/get-tweets-counts-recent

https://developer.twitter.com/en/docs/twitter-api/v1/trends/trends-for-location/api-reference/get-trends-place

https://developer.twitter.com/en/docs/twitter-api/tweets/counts/api-reference/get-tweets-counts-recent

https://developer.twitter.com/en/docs/twitter-api/v1/trends/trends-for-location/api-reference/get-trends-place

https://github.com/twitterdev/Twitter-API-v2-sample-code

https://json2csharp.com/json-to-pojo

https://learning.oreilly.com/library/view/docker-deep-dive/9781800565135/chap11.xhtml#leanpub-auto-compose-files