Semester 3

Teun Visser

442651

Design Document

Pollstar

Contents

[Project Description 2](#_Toc83648126)

[Architecture 3](#_Toc83648127)

[How is Solid Guaranteed: 4](#_Toc83648128)

[C4 Architecture diagrams 5](#_Toc83648129)

[Design decisions and applied research 6](#_Toc83648130)

[UML Class diagram 7](#_Toc83648131)

|  |  |  |
| --- | --- | --- |
| Version | Edits: | Time |
| .1 | Document Creation and setup | 27-09 |
| 1.0 | Actually making the document, filling it with necessary diagrams and related info | 08-10 |
|  |  |  |
|  |  |  |

# Project Description

## 

## User Problem:

Making a decision where there are several options and there is no clear answer, getting input from a lot of users will be a difficult task.

## Pollstar as solution:

A simple question should have a simple answer, set a question and the answers, and let everyone from the crowd pick his or her favorite from your provided options. This was, the most voted option will come forward, together with the info about how many people voted for it. This is also compared to the other options, and comments will be shown from the voters to elaborate on their chosen answers.

## Description:

A tool for users that need the opinion of a large amount of people, like an audience. But also possible for use in group projects, classrooms or events. This can be used for evaluation, but its main focus is to make collaborative decisions.

The application will focus on speed and ease of use, this means people won’t have to create an account to vote, and help making a decision. Instead, you will be directed to the poll immediately and be prompted to vote for your favorite option. Of course, you can only vote once to avoid manipulating results, but a user can always retract his vote and cast it on another option on the poll, as long as the poll isn’t locked.

What other people voted will only be available after you voted, this ensures our users don’t “Go with the pack”, but make a decision for themselves.

There also might be scenarios where you want to limit the sample size for your audience or get an opinion on a time limited issue. For times like these you can *lock* your vote, this will make the poll un-vote-able, meaning that new voters can’t cast a vote and people that already voted on your poll are unable to recast their vote.

These are all precautions to make sure that the pollster gets the fairest and most accurate response.

# Architecture

## How is Solid Guaranteed:

SOLID is about the principles of object-oriented programming

Single-Responsibility principle:

This has been accounted for since the start of the UML, therefor all classes but also separate applications have one purpose, and they will be built for that purpose only.

Open–closed principle:

I have to admit that at this point in the application this rule isn’t being upheld that well, once the simple application has been setup however, the UML will be updated and from that point on modification of the application will ensure the persistence of this rule.

Liskov substitution principle:

This rule can be tested in the test part of the application, which will be done on some parts, the UML has been created with this principles in mind.

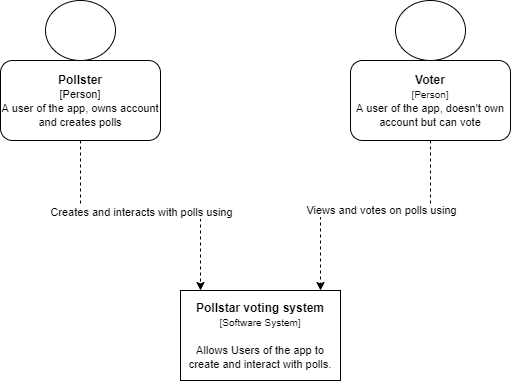
Interface segregation principle:

I proved to ensure this principle in the 3 diagram of the C4 architecture.

Dependency inversion principle:

I am unable to prove dependency inversion as of right now, because of the fact that there hasn’t been an opportunity to use it, yet. However I am sure planning on implementing this principle, to ensure nothing failing later.

## C4 Architecture diagrams



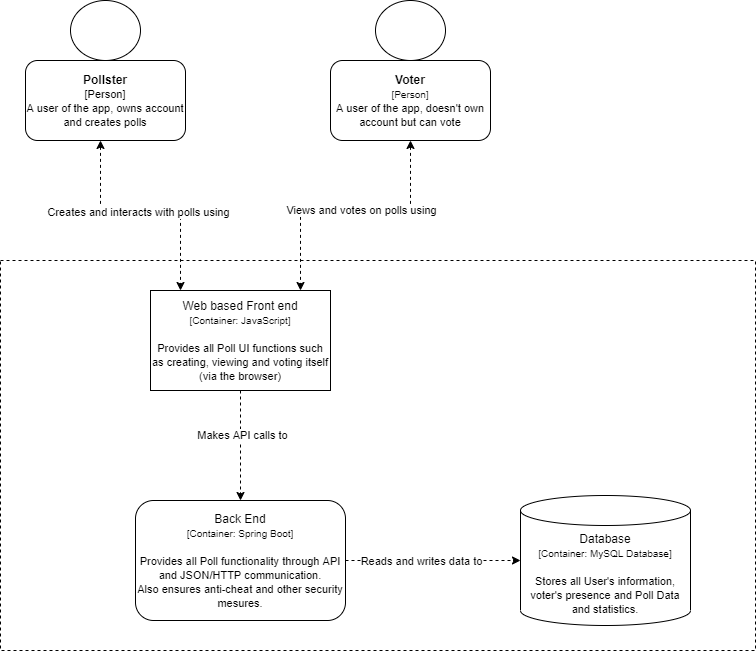
A simple C1 diagram to show user interaction with the application, as shown, there will be two main users that interact with the application:

The Pollster, this user will have to create an account to create polls.

The Voter, this user can vote on polls anonymously

There will also be some users upgraded to Admin, but they are a bit of an anomaly, they can lock and block off existing polls.

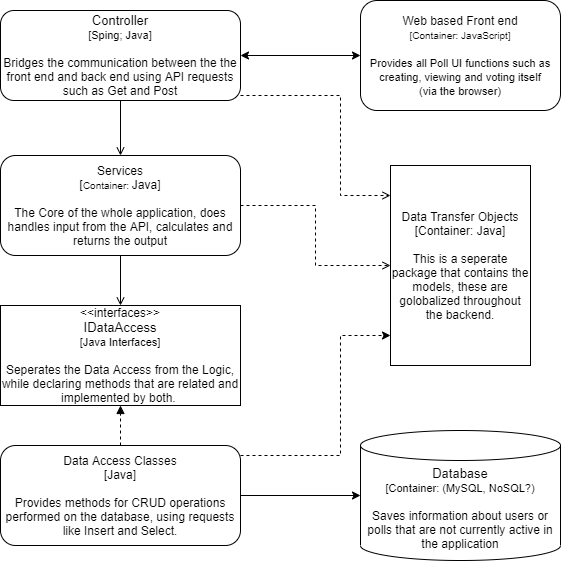
This would make them only available to the pollster that created the poll, so he can edit it and re-publish it again



The C2 Diagram, all the components I am planning to use are listed here.

The arrows represent how data is called and the path it takes from storage to end-user.

Here in this C3 Part, the data flow in the whole application is explained.

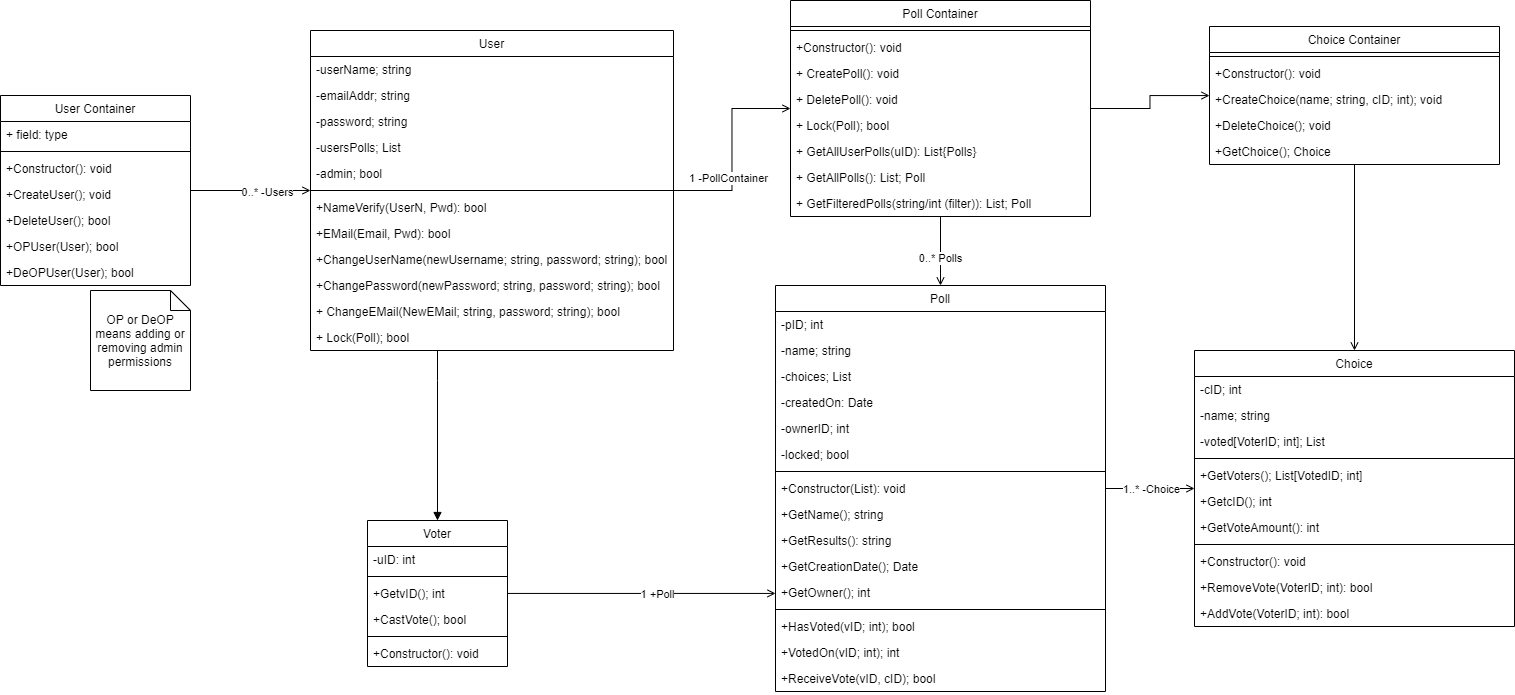


There are multiple ways of tackling one of these diagrams, but showing the components like this helps me defend my application the best.

The database for now has MySQL, NoSQL because I decided too late what Database I am going to use, because of a misunderstanding on my part I don’t have any plans for that *yet*.

## UML Class diagram

Doubling as a placeholder C4 Diagram but also my UML, my UML:

(it is also included in the documentation folder as Pollstar ClassDiagram if the size is too small)

Construction on this UML has been started quite early in the application, it has been changed a lot and is still subject to change according to the agile way of working :)