Project Overview

Software Engineering (T-303-HUGB)
Fall term 2020

Instructor: Grischa Liebel
Teaching Assistant: Friðrik Örn Gunnarsson

Group 9
Aríel Jóhann Árnason
Egill Smári Snorrason
Gunnhildur Skarphéðinsdóttir
Haraldur Daði Þorvaldsson
Jóhann Hilmir Gunnarsson
Ægir Máni Hauksson

The Opportunity

In today’s society, we rely more and more on statistical analysis. Statistical polling analysis is a fundamental part of finance, medical fields, disease analysis, elections and politics as well as many other important fields. Polling data furthermore affects important decisions made by all industry and corporate scopes, as well as that of individuals.

This tremendous increase of opinionated data also creates a **huge increase in need for statistically presentable polling information**. Being able to store various poll data, and immediately see it in a statistically presented manner would therefore be an ideal solution for many companies, institutions or curious individuals.

Our vision

To combat the problem, we envision a web service. This web service will be hosted in the form of a server that can be used to access, create and view various polling statistics, via an easy to use and uniform API. Meaning that any website interface can use the service with ease, and thereby limit the amount of time it takes for companies as well as individuals in creating a website that displays statistical data.

The service will allow users to create new voting polls. It will enable data gathering from different sources. The users of the system will have the possibility to change statistical values temporarily to view a certain theoretical outcome graphically.

The service will also show live poll statistics in times of voting, and thus be an important source of information during elections. Finally the service will provide open unofficial online polls, which will be of further information for the users to see a little bit more information about the current issue from other users.

Goals

The following list presents our goals in this project, in a short and simple manner.

We want to…

- enable swift development of statistical websites
- create a user friendly polling data service
- make polling data of all kind readily available to customers
- make stored data easily presentable
- enable users to vote on various polls to create statistical data
- ensure efficient data collection

Project scope

The system will consist of a backend that will translate data from outside the system to a simple and easy to use API. Data from the outside system will come from various polling sources.
The frontend graphical interface outside of the system will then use the API's to display polling statistics and information. The system will also be able to take in new votes on newly created polls along with translating them to representable data.

Stakeholders

Stakeholders such as voters and users benefit from using the application, they get an overview of the possibilities and statistics which could help them in forming their opinion on each matter. Candidates and campaign workers will be able to view the polls and see where they stand in the election. That means that candidates who are losing the polls will see it as a negative impact but those who are winning will see it as a positive. The product owner would benefit from the profit of the product that we are developing. As the website would increase in popularity and traffic, the pollsters would gain recognition for their polls which will be a large benefit for them. And last but not least , the front end developers will receive data via Api in a representable form which is quite a positive benefit.

Scenarios

Scenario	A user views polling statistics
Starting Situation	User is using an application that uses our service
Normal Event flow	1. User clicks on some functionality in the application that lets him graphically view data. 2. The applications logic sends a request to our server with the parameters required to retrieve the wanted data. 3. The server responds by sending the application an array containing json objects that contain the requested data along a 200 status.. 4. The website displays the data received graphically.
Exception flow	3a. The application has been configured incorrectly and the server responds with a 400 status. 3b. The website requests data on a poll that does not exist and the server returns a 404 status. 3c. The user is not authorized to view the data and the server returns a 401 status.
Concurrent activities	The user waits while the data is getting fetched
Post scenario	The user gets the statistics he requested displayed to him graphically through the website/interface

Scenario	A user views a specific poll’s data in order of leading poll option
Starting Situation	User is using an application that uses our service and is currently viewing a certain poll.
Normal Event flow	1. Through some functionality in the application the user selects to view data ordered by percentage (city/state/country etc...) 2. The applications logic sends a request to our server with the parameters required to get a specific poll’s data sorted by percentage) 3. The server responds by sending the application the specific poll’s votes as an array of json objects ordered by ranking, each object in the array denoting a different polling option. 200 status (ok). 4. The application presents the data in graphical format.
Exception flow	3a. The application has been configured incorrectly and the server responds with a 400 status. 3b. The application requests data on a poll that does not exist and the server returns a 404 status. 3c. The user is not authorized to view the data and the server returns a 401 status.
Concurrent activities	The user waits while the data is getting fetched
Post scenario	The user gets the statistics he requested displayed to him graphically and sorted by leading poll option

Scenario	A user views a specific poll’s data sorted by voters location
Starting Situation	User is using an application that uses our service and is currently viewing a certain poll.
Normal Event flow	5. Through some functionality in the application the user selects to view data by location (city/state/country etc...) 6. The applications logic sends a request to our server with the parameters required to get a specific poll’s data sorted by location. 7. The server responds by sending the application the specific poll’s votes as an array of json objects, each object in the array denoting a different location and it’s statistics. 8. The application presents the data in graphical format.
Exception flow	3a. The application has been configured incorrectly and the server responds with a 400 status. 3b. The application requests data on a poll that does not exist and the server returns a 404 status. 3c. The user is not authorized to view the data and the server returns a 401 status. 3d. The poll does not contain location information and the server returns a 204 (no content)
Concurrent activities	The user waits while the data is getting fetched
Post scenario	The user gets the statistics he requested displayed to him graphically and ordered by voters location

Scenario	A user creates a new poll
Starting Situation	User is using an application that uses our service
Normal Event flow	1. User clicks on some functionality in the application that lets him create a new poll. 2. The applications logic sends a request to our server with the parameters required to create a new poll 3. Our server responds by sending the application the id of the new poll as a json response along with a 200 status. 4. Through some application functionality the user gets to add polling options to his poll. 5. The application logic sends the options created by the user in the request body of a request. This request has the parameters required to add the options to the specific poll. 6. The server responds by sending the application an array containing the updated poll as a json object along a 200 status. 7. The website displays the data received graphically.
Exception flow	3a. & 6a The application has been configured incorrectly and the server responds with a 400 status. 3b. & 6b The requested data does not exist and the server returns a 404 status. 3c. & 6c The user is not authorized to view the data and the server returns a 401 status.
Concurrent activities	The user waits while the data is getting fetched
Post scenario	The user has created a poll

Scenario	A user votes on a poll
Starting Situation	User is using an application that uses our service
Normal Event flow	1. User clicks on some functionality in the application that lets him view an active poll. 2. The applications logic sends a request to our server with the parameters required to view the specific poll. 3. Our server responds by sending the application the poll and its data as a json object along with a 200 (ok) status. 4. Through some application functionality the user gets to vote on one of the available poll options. 5. The applications logic sends a request to our server with the parameters required to vote on one of the available options in the specific poll. 6. The server responds by sending the poll and it's updated data as a json object to the application along with a 200(ok)status. 7. The application registers the success and displays feedback accordingly.
Exception flow	3a. & 6a The application has been configured incorrectly and the server responds with a 400 (bad request) status. 3b. The requested data does not exist and the server returns a 404 status. 3c. The user is not authorized to view the poll and the server returns a 401 (unauthorized) status. 6b. The user is not authorized to vote on the poll and the server returns a 401 (unauthorized) status. 6c. The user already voted on the poll where only one vote is allowed and the server returns a 405 (not allowed) status.
Concurrent activities	The user waits while the poll is getting fetched The user waits while vote is being registered
Post scenario	The user has voted on a poll.

Detailed Domain model

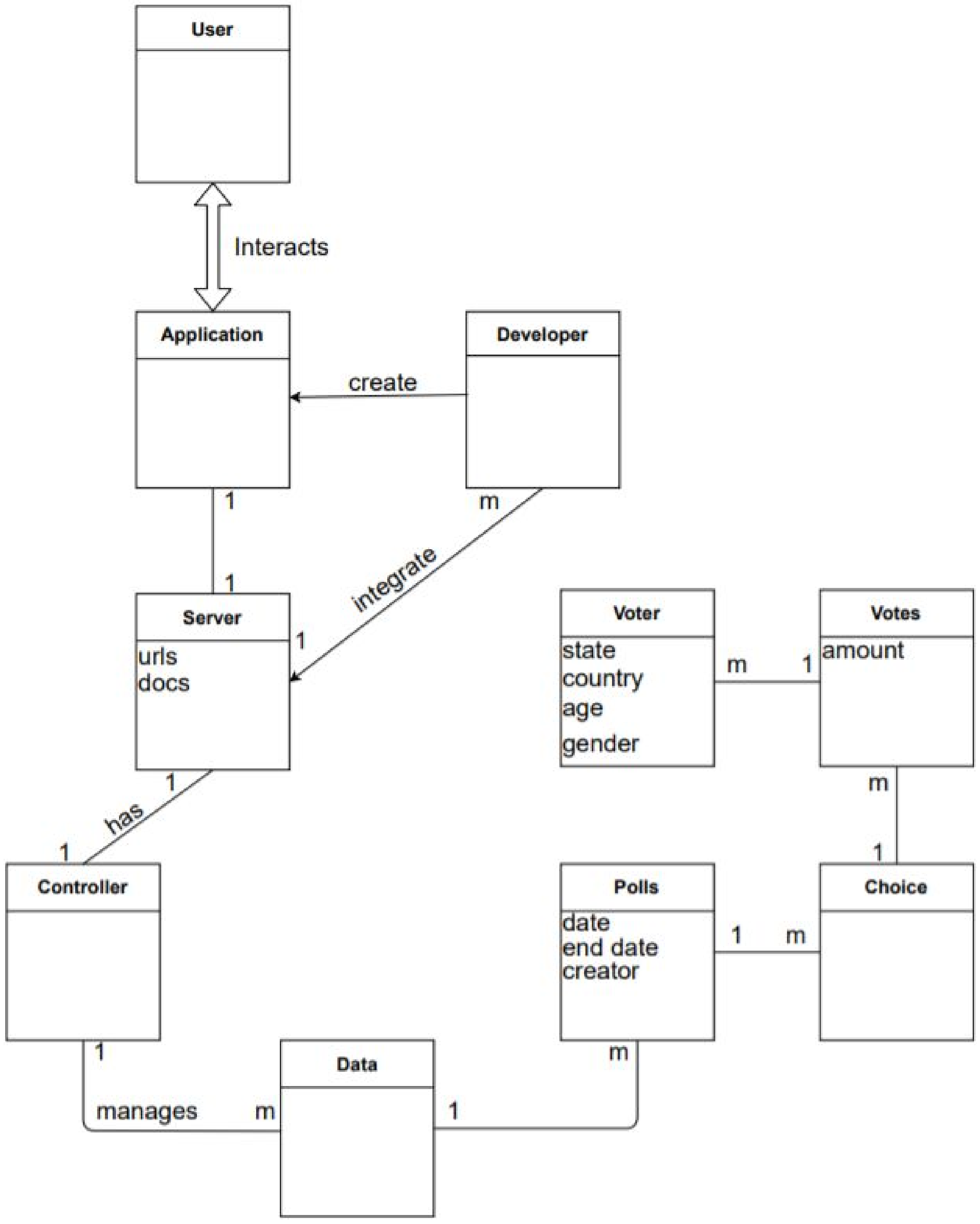

Image 1

Abstract Domain model

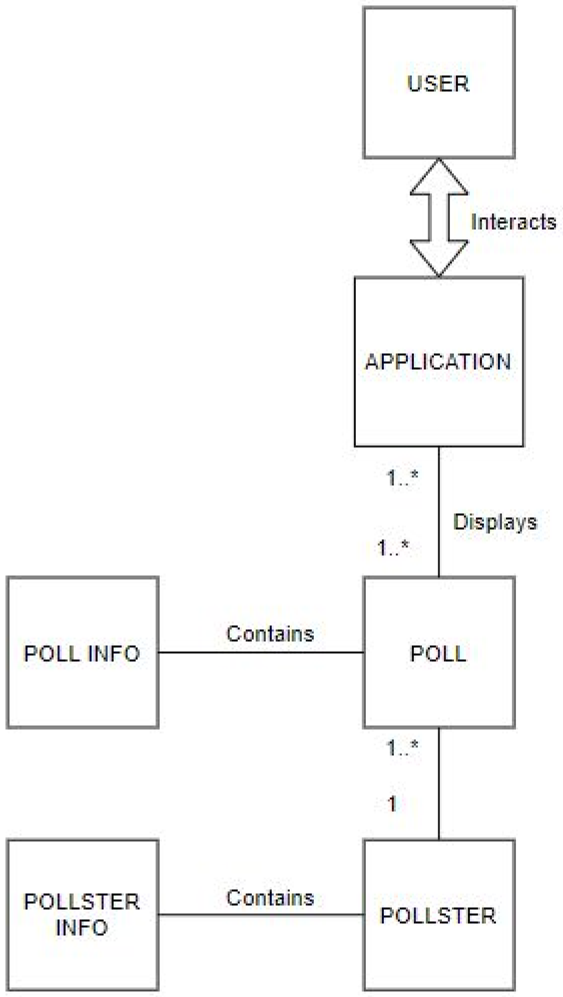

Image 2

Appendix

The flowchart seen on image 2 is how the system behaves when a user views a poll from a specific state in a certain timeframe.

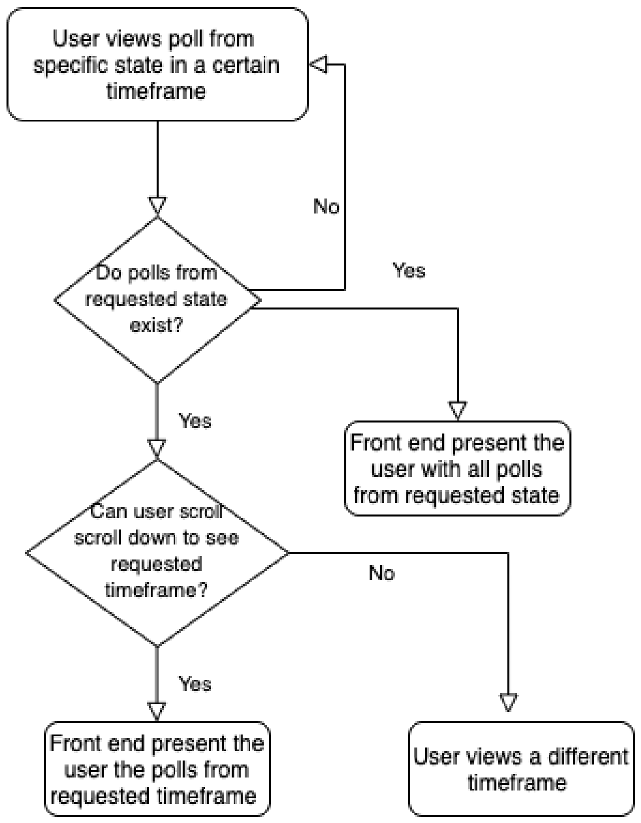
User views poll from specific state in a certain timeframe
No
Yes
Do polls from requested state exist?
Yes
Front end present the user with all polls from requested state
Can user scroll scroll down to see requested timeframe?
No
Yes
Front end present the user the polls from requested timeframe
User views a different timeframe

Image 3