Final Project

Programming Scalable Systems

Universidad Politécnica de Madrid

2021-2022

Reminder

No exams

personal work every two weeks: mandatory exercises
just some of them will be reviewed (20%)

1 **final** project
team of two
(50% git review + 30% presentation)

The Final Project

This document describes in detail a proposal for a concrete final project which you may work on (The Telegram Weather Bot proposal), and provides some limited information on interesting alternative projects (Chord, Betting site, Twitter clone, etc).

However, you are allowed, and indeed **encouraged**, to choose yourselves a theme for the course project. Please email us your own project ideas for approval – make sure that the difficulty of realising them is at least as hard as realising the Telegram Weather Bot Proposal detailed below.

The Telegram Weather Bot Proposal

This projects aims to implement a Telegram bot which provides tailored weather information sent as Telegram messages.

The bot should at least be capable of providing the following type of weather information (but you are allowed, and very much encouraged, to implement further functionalities):

1. **Forecasting**: the user introduces the name of a region (a province for instance) and the bot returns a text description with the forecast for that region.
2. **Alerts**: the user introduces the name of a region (a municipality for instance), a symbol “>” or “<” and a temperature and the bot will send a message to the user when the forecast in the next 6 hours triggers the alert. Other possible alerts that you may want to implement are heavy rain alerts, heavy wind alerts, etc. Experiment!

A bot in telegram is a third-party application that run *inside* Telegram. You will need to check the introduction to bot developers from Telegram at `https://core.telegram.org/bots`.

To implement your Telegram Bot you will need a library. We suggest you to use ExGram. ExGram is a powerful framework implemented by a former UPM student `:)`. The documentation of the library can be found at `https://hexdocs.pm/ex_gram`.

To obtain weather information you will need to access a system that offer such information. We suggest you to use the AEMET OpenData service from the AEMET (*Agencia Estatal de Meteorología*. AEMET OpenData is an API REST. You can obtain credentials to access the service as well as documentation from `https://opendata.aemet.es/`. The main REST resources to implement the proposal might be

- `/api/prediccion/especifica/municipio/horaria/{municipio}` and
- `/api/prediccion/provincia/hoy/{provincia}`.

Both theses are relatively well documented at `https://opendata.aemet.es/dist/index.html?`.

Norms and Deadlines

The deadline for completing the project is **June 5th 23:59:59**.

1. Create a git repository at Gitlab or Github. Your repository **must be private**.
2. Include a file `AUTHORS` with your names and UPM emails, one per line, in this form:

   ```
   "Sofía Huertas Arévalo" <sofia.huertas.arevalo@alumnos.upm.es>
   "Luis Fernández Araujo" <luis.fernandez.araujo@alumnos.upm.es>
   ```

3. Work in the project following good practices like *small and single-purpose commits*, and *detailed commit messages*.
4. Include a set of no more than ten slides in PDF: `presentation.pdf`. The contents of the slides should document e.g. your main decisions in developing the project, report on difficulties encountered, and should contain an evaluation (enumerating positive, negative aspect) of the programming frameworks and libraries used, based on your personal experiences when developing the project.

Code Review (50%)

To review your code, after the deadline, we will ask you to give us access to your repository.

Presentation (30%)

We will make an appointment with you to make a 5 minutes presentation on **June 10th** between 09:00 to 13:00.

Teams

We want you to work in teams of two. But, if you have a bigger proposal, two teams can work together.

Alternative projects

Apart from the Telegram Weather Bot proposal, below we enumerate a few other project ideas:

- Program a Chord implementation.

  Chord is a distributed key-value store – a distributed hash table. Basically Chord is implemented as a distributed ring network, where each node in the ring stores a separate part of the keys of the key-value store. A client connects to the local ring network node asking for the value corresponding to a key. If the key is not local, then the request is forwarded to remote nodes in the ring network according to a well-defined protocol.

  There is a lot of information on the Chord protocol in the internet (Chord is a pretty famous peer-to-peer protocol), in Wikipedia, and the original research article introducing Chord is also available online.

- Implements any other Telegram bot (find inspiration at `github.com/DenisIzmaylov/awesome-telegram-bots`).

- Clone a betting site (bet365 ;))

- Clone a survey web app, e.g. Kahoot.

- Clone your favourite social network: *Twitter*, *Whatsapp*, *Instagram*... To give you an idea of what we would expect, it would be enough that you to *implement a Twitter clone* including:

  1. *Registration* (signup) and *login* pages.
  2. *Profile* page with *username*, *name* and *password* (the user can change any of them).
  3. In the profile page, the user can *start following* other user (if knows the username, i.e. no search needed).
  4. A page with the list of *followed* users.
  5. A page with the list of *followers* (could be the same page in two columns).
  6. A home page with *all the posts*: the own user posts and the followed users' posts (first the newest).
  7. A page *to post*.
  8. Proper and sensible *navigation* among pages.

- Papibank: Web app/REST API for daddies and childrens to manage grandma money donations `;)`

  For any web application or web service you need a Web Framework. Of course, our recommendation is to use `https://phoenixframework.org/`.