HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Méthodes d'accès aux données 2019–2020
Professor : Nastaran Fatemi
Assistants : Christopher Meier & Guillaume Hochet

MAC Project

1 Introduction

1.1 Objectives

The goal of this project is to put into practice what you've learned in this course by creating a social media application with a focus on the storage and analytics using NoSQL technology. For this purpose, you will be using already existing clients like *Telegram, Slack* or even *WhatsApp* to create bots and automatic agents that can answer requests and queries.

1.2 Organization

This project should be realized in groups of maximum three students.

Report: We'll ask for documentation along with the source code which should include a small user guide on how to use your application, what kind of implementation choices you've made, and a good readme describing the steps needed to deploy your application.

Presentation: All the members of your group will present the work at the end of the semester, explaining the project along with a small demonstration.

2 The backend application

2.1 Working with resources

The objective is to develop a social media application that allows users to interact with other *users* and with *resources*. A resource can be anything, a picture, a PDF file, a film, an artist or any kind of data. We distinguish two kinds of resources, which will also give you different directions for your project.

1. **Resources you generate**. Those are data that you take care of creating, storing and retrieving via your application. You will have to think of uploading pictures, saving a todo list or simply storing exchanged messages.
2. **Resources you consume**. Those resources are provided by a third-party API like IMDB for example, or using a pre-existing dataset that you import into your database. You will have to reference them later in your system.

2.2 Storing resource metadata

Your resources will have metadata of all kind, e.g. clicks count, date, description, hashtags and so on. You will have to store them server side in an appropriate database.

2.3 Linking everything

Once you have defined users, resources and associated metadata, you'll have to specify the *relationships* between them. Users can be related to multiple resources and descriptions via different interactions. For example a user can "share" a post, "like" a post shared by another user, "add descriptions", etc. It is important that you

clearly define what kind of interactions and relationships you take into account based on queries an analytic requirements of your application. This will directly impact your choice of data model and database technology.

2.4 NoSQL Data model specification

How you develop the backend is entirely up to you. You can choose any language or framework. The only requirements are:

1. You must store metadatas (and resources if you choose to manage them by yourselves) in a document-based database.
2. You must store interactions and relationships between your resources (and eventually metadatas) in a graph-based database.

In case your application requires more advanced search capabilities (ranking and text indexing), you can integrate Lucene (or derivates like ElasticSearch) within your project and receive bonus points.

Below is a list of databases you can use. If you want to use another one you're very welcome to come and discuss with us.

- Document-based : MongoDB, OrientDB, ArangoDB, CouchDB...
- Graph-based : ArangoDB, OrientDB, Neo4j...

The logic of your application is up to you, but we want to see a real usage of those databases, both document-based and graph-based database. For example, given a *social network on music*, we expect a feature showing us trending artists my friends liked and that I like too. **But don't panic!** We don't expect a complete social network or full-featured application, start simply, defining a single resource with its schema and progressively build upon it the more you go.

3 Client part

For this project, you will be using existing clients that offer a development API rather than coding your own desktop, mobile or Web-based frontend. We want you to use an existing platform, for example Telegram, Slack or even Facebook messenger to develop a bot with which users will interact. Your back-end will be connected to the client API and be able to handle different types of queries depending on your application logic.

3.1 Choosing the client

Choosing the client is very deeply connected to what kind of application you want to create. For example, given Telegram, a bot can be used in three different contexts:

1. A one-to-one discussion with the bot
2. A group with the bot in it
3. A bot message inserted in a discussion offering interaction possibilities

So you must begin by gathering information on each client's functionalities and what can be done with their API. Some are focused on discussion while others on reporting data and so on.

3.2 List of clients

Here is a list of a few clients you can use to develop your application. They all offer a free-tier API, at least for development. This is purely to give you some ideas on what kind of client you can use, if you want to use another one don't hesitate to discuss it with us.

- Telegram
- WhatsApp (using Twilio)
- Twilio (SMS, calls, emails...)
- Facebook Messenger
- Instagram
- Slack
- Mattermost (an open-source Slack alternative)
- Rocket.Chat (another open-source alternative)
- Discord
- Github

4 Planning

Here are the first steps of your project and the related deadlines:

1. Study of the existing clients and their APIs (you should begin immediately!)
2. Description of your proposed project, including a summary of goals and functionalities and your choice of client API. (Due: 03.11.2019 23:59)
3. Specifications of your project: cahier des charges and a link to your git (before 08.12.2019 23:59)
4. Data model and analytic requests (15.12.2019)
5. Discussion on the status of your project (06.01.2020)
6. Presentations, max. 15 minutes each group. (20.01.2020)

All deliverables are to be uploaded on Cyberlearn before due date. Your code, along with your slides and your report must be available to us on an git platform and latest commit must be dated as before 12:00 - 20,01.2020.