INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Rapport Project WEB

Planning Maker Version 2

Alban PRATS
Bedredin CELINA
Denis MAHMUTOVIC
Matthieu RAUX

Planning Maker V2

This project is made on behalf of the course “Projet WEB dynamique”, with the objective of having a full stack web project i.e having structured data, a database system and data querying in order to generate and manage dynamic information.

The concept is to have a website in which event managers can create events, tasks, and attribute these tasks to organisers which are able to select their free time slots. This way, task management is easier to take care of when an event has a lot of organisers and tasks.

Used Technologies

In order to put the project into place two main technologies have been used. Django for the back-end and React for the front-end. Communication between the two is done via a restful API. Hence the backend is written in python while the frontend in javascript.

Django

For our back-end of our project we decided to go with Django since it is Python and we are all familiar with it. It is easy to learn and has a well suited REST framework which is needed for our restful API. Our system is connected with a SQLite3 database.

Therefore we manipulate the information as django objects, while in the background django handles the sql queries.

Under “/back/urls.py”, all api paths are defined along with their corresponding resources. These resources are organised into different directories having a “views.py” file and a “serializers.py” file. The view file handles all the possible HTML requests while the serializers formats the data that we need to work on.

Under “models.py” we have all of our tables/objects of data defined and their corresponding attributes.

React

For the front-end of the project, we went with React since it is easy to learn in a short time and it is suitable for our projects needs. In addition, the community is rather big, so we can get inspired by many examples and other open-source publications.

In order to connect our front-end and back-end we use the implemented API.

Information System

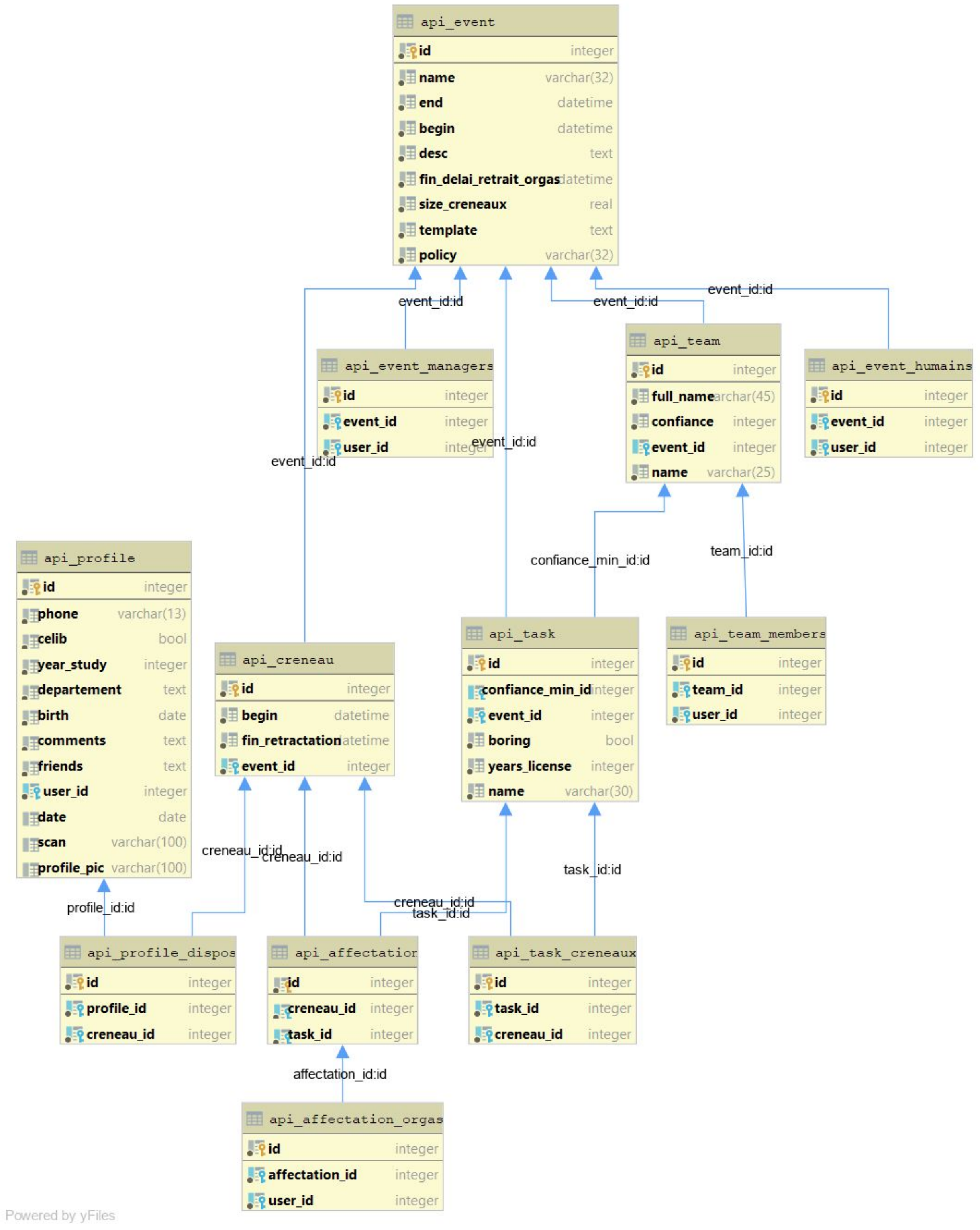
SQLite

Since we are using Django as the back-end web framework. We also decided to go with SQLite as our database system since it is the default system for Django. And it is well suited for our database needs.

In order to set the database, we have multiple tables which are linked between them. The main tables in order to run the system are:

User(id,password,last_login,is_superuser,username,first_name, email, is_staff, is_active, date_joined, last_name)
Profile(id, phone, celib, year_study, departement birth, comments, friends, user_id, date, scan, profile_pic)
Event(id, name, end, begin, desc, fin_delai_retrait_orgas, size_creneaux, template)
Teams(id, name, full_name, confiance, event_id)
Task(id, name, confiance_min_id, event_id, boring, years_license)
Affectation(id, creneau_id, task_id)
Creneau(id, begin, fin_retractation, event_id)

It is easier to understand the relation between the tables using the diagram below. Notice that there are some other additional tables, this is due to a many to many relation between certain tables.


api_event
id integer
name varchar(32)
end datetime
begin datetime
desc text
fin_delai_retrait_orgas datetime
size_creneaux real
template text
policy varchar(32)
event_id:id
event_id:id
event_id:id
api_event_managers
id integer
event_id integer
user_id integer
event_id:id
event_id:id
api_team
id integer
full_name varchar(45)
confiance integer
event_id integer
name varchar(25)
api_event_humains
id integer
event_id integer
user_id integer
confiance_min_id:id
team_id:id
api_profile
id integer
phone varchar(13)
celib bool
year_study integer
departement text
birth date
comments text
friends text
user_id integer
date date
scan varchar(100)
profile_pic varchar(100)
api_creneau
id integer
begin datetime
fin_retractation datetime
event_id integer
api_task
id integer
confiance_min_id integer
event_id integer
boring bool
years_license integer
name varchar(30)
api_team_members
id integer
team_id integer
user_id integer
creneau_id:id
creneau_id:id
task_id:id
profile_id:id
creneau_id:id
task_id:id
api_profile_dispos
id integer
profile_id integer
creneau_id integer
api_affectation
id integer
creneau_id integer
task_id integer
api_task_creneaux
id integer
task_id integer
creneau_id integer
affectation_id:id
api_affectation_orgas
id integer
affectation_id integer
user_id integer
Powered by yFiles

API

In the implementation each path of the API relates to a resource.
Our resources are related to our database tables. Therefore we have:

- Profile
- Task
- Event
- Me
- Token
- Auth

The “/api/me” gives the information related to the user, like their personal information and their assigned tasks.

The “/api/profile” is used in order to post/update information about the profile such as when the data is modified.

The “/api/event” is used in order to get/post/update information about Events.

The “/api/event/tasks” is used in order to get/post/update information about tasks of an Event.

The “/api/auth” is used for authentication processes.

There are many more api links, but these are the main ones.

Google API

In addition to standard authentication methods. We decided to add the google authentication method for quick sign-ups.
This is implemented under API auth where we connect to the Google API in order to get the basic information in order to sign up.

When the client signs-up, a query is sent to the google server, which responds with an id. This id is then sent by the client to our server, the server uses this id to do a query to the google server to get access for profile data. When the data is received, a confirmation is sent to the client to confirm that the account was set up.

Project management

Working methodology

The project started with a big brainstorming meeting where we proposed ideas and analysed them. Overall we went with a difficult project which was not finishable by the end of the dedicated time, but we chose it since it will be a useful project and we want to work on it even after the subject finishes and gets evaluated.

We had 28h of sessions in class. And at least double of that outside of class.

The first session revolved in the conception of the database and how our objects would be related to each other by still respecting a restful api model. Programming of the back started the same day. On the second session, we noticed some stuff that had to be rethought, therefore some minor changes of the design were done in order to make up for this.
The rest of the sessions were done on the front end, nevertheless some work on the back end was always done as we disc

GitLab

Since we were multiple programmers at the same time, we used a version control repository manager called GitLab. Here two projects were dedicated. The back and front project. All of the development was done in separate branches and merged onto main after the code was properly done and the code was verified by the main branch manager.

Working sessions

In the end, working together proved the most effective since we were able to ask questions to each other on the go, and troubleshoot our problems together.
Hence we developed during our dedicated time in class, and met up out of class in order to continue working since the project was bigger than the reserved time.
We met at least an additional meeting per week to do a follow up of our progress, and our communications were done in a dedicated channel.

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Bibliography

https://www.django-rest-framework.org/
https://reactjs.org/
https://developers.google.com/identity/
https://github.com/thebeaverhead/react-modal-login
https://github.com/intljusticemission/react-big-calendar