SDIS Trabalho #2: Relatório

A nossa aplicação chama-se Social Crowd e as suas funcionalidades/arquitetura estão descritas na respectiva especificação.

Contrariamente ao que foi estabelecido na especificação, optámos por desenvolver a nossa própria REST API, recorrendo à linguagem PHP e à *framework* Slim para a implementação. Em anexo, encontra-se a especificação da nossa API.

Em Java, foi elaborado um *package* com o nome *pt.up.fe.socialcrowd.API* que possui os métodos necessários para a comunicação com o servidor.

Embora tenhamos criado a nossa própria API, não deixamos de usar a API QuickBlox para registo e autenticação de utilizadores. O registo é feito pela aplicação, que comunica directamente com a API QuickBlox enquanto que a autenticação é feita pelo nosso servidor. Para autenticar um utilizador, a aplicação precisa de criar uma sessão na nossa API, passando os parâmetros *login* e *password* no pedido https respectivo. A nossa API fica encarregue de comunicar com a API QuickBlox para autenticar o utilizador e, caso a autenticação seja bem sucedida, fornece à aplicação uma *session id.*

Como mecanismos de segurança, optámos por usar o protocolo HTTPS para estabelecer a comunicação entre a aplicação e a API e, para além disso, para efectuar algum pedido PUT, POST ou DELETE é necessário passar uma *session id* válida (obtida através da autenticação de um utilizador na nossa API) no *header* do respectivo pedido.

Todos os membros do grupo esforçaram-se ao máximo para implementar o máximo número de funcionalidades possível no tempo que nos foi fornecido.

O João e o José ficaram encarregues da implementação da API. Posteriormente, ficaram encarregues da integração do Google Maps na aplicação.

O Tiago e o Miguel ficaram encarregues do desenvolvimento da interface da aplicação e da sua integração com o Quickblox.

# Especificação API

Módulos:

* Event
* Comment
* Vote
* Rating
* Subscription
* Session

Módulo Event

|  |  |  |  |
| --- | --- | --- | --- |
| URL | HTTP Verb | Action Description | Success HTTP Status Code |
| /events | GET(\*) | Retrieve all events | 200 |
| /events/:id | DELETE | delete :id event | 200 |
| /events/:id | GET | Retrieve :id event | 200 |
| /events | POST(\*) | Create new Event | 200 |

GET(\*) Filtros opcionais:

* owner\_id
* subscriber\_id
* category
* name
* tags
* type

POST(\*) Parâmetros:

* location
* type
* category
* name
* description
* start\_date
* end\_date

Módulo Session

|  |  |  |  |
| --- | --- | --- | --- |
| URL | HTTP Verb | Action Description | Success HTTP Status Code |
| /sessions | POST | Authenticate User | 200 |
| /sessions/:id | DELETE | signout user | 200 |

Módulo Subscription

|  |  |  |  |
| --- | --- | --- | --- |
| URL | HTTP Verb | Action Description | Success HTTP Status Code |
| /subscriptions | POST | create a subscription | 200 |
| /subscriptions/:id | DELETE | delete :id subscription | 200 |

Módulo Vote

|  |  |  |  |
| --- | --- | --- | --- |
| URL | HTTP Verb | Action Description | Success HTTP Status Code |
| /votes | POST | create vote | 200 |
| /votes/:id | DELETE | delete :id vote | 200 |

Módulo Comment

|  |  |  |  |
| --- | --- | --- | --- |
| URL | HTTP Verb | Action Description | Success HTTP Status Code |
| /comments | GET(\*) | get all comments | 200 |
| /comments | POST | create a comment | 200 |
| /comments/:id | GET | get :id comment | 200 |
| /comments/:id | DELETE | delete :id comment | 200 |
| /comments/:id | PUT | update :id comment | 200 |

GET(\*) Filtros opcionais:

* created\_at
* eventid

Módulo Rating

|  |  |  |  |
| --- | --- | --- | --- |
| URL | HTTP Verb | Action Description | Success HTTP Status Code |
| /ratings | POST | create a user rating | 200 |
| /ratings/:id | DELETE | delete a user rating | 200 |