Protocolo interno

# Cliente – API

Llamadas tipo GET:

* GET("api/user/:username") => devuelve OK o ERR
* GET("api/event/:eventid") => devuelve JSON del evento o ERR
* GET("api/list/:username") => devuelve JSON con la lista (puede estar vacía) de eventos filtrada por (organizer === :username) o ERR (si no existe usuario

Llamadas tipo POST:

* POST("api/add/event", JSON con todos los datos de event menos eventid) => devuelve eventid o ERR
* POST("api/remove/event", JSON con :eventid) => devuelve OK o ERR
* POST("api/add/assistant", JSON con :username y :eventid) => devuelve OK o ERR
* POST("api/remove/assistant", JSON con :username y :eventid) => devuelve OK o ERR
* POST("api/add/user", JSON con :username) => devuelve OK o ERR

# API – BBDD

Datos del procolo BD-API:

-> Se realiza la conexión mediante sockets TCP con la librería 'zmq' de Nodejs.

-> IP: localhost o 127.0.0.1 y puerto 1234

-> Se envían 'strings' en los mensajes que parten de formato JSON.

-> En los mensajes de petición de la API a la BD se indican los siguientes parámetros:

-> 'component': indica si se trata de una petición para usuarios o para eventos

-> 'id': el identificador del elemento que quiera introducirse o consultarse en la BD

-> 'body': es el cuerpo de la petición, el cual contiene otros parámetros según el tipo de elemento:

-> Sea el tipo que sea, siempre se indica 'op': la consulta a realizar en la base de datos. Puede ser únicamente 'put' o 'get'

-> Si se trata de un usuario no se indica ningún parámetro adicional, puesto que con el 'id' antes indicado es suficiente.

-> Si se trata de un evento, sólo en el caso en el que sea 'put', se indicarán los detalles del mismo (name, datetime, description, organizer)

-> 'organizer' es el identificador de un usuario ya registrado en el sistema que cree dicho evento.

-> Como respuesta a las operaciones la BD devuelve a la API:

-> Para 'put' de 'user': 'Done' la operación se ha realizado correctamente y 'Failed' en caso contrario.

-> Para 'put' de 'event': 'id' aleatorio generado para el nuevo evento registrado (de ser correcto) y 'Failed' en caso contrario.

-> Para 'get' de 'user': 'OK' si existe el usuario y 'Failed' en caso contrario.

-> Para 'get' de 'event': los detalles que lo definen y 'Failed' en caso contrario.

# Puertos utilizados

* Client = 4000
* API = 3000
* BD = 1234

# Entidades

* user
  + Name (string)
* event
  + id (string)
  + name (string)
  + datetime (string)
  + description (string)
  + organizer (string)
  + assistants (list of string)