Documentation Webservice for Event Planner

- **eventPlannerServer.php**
  - This script is located on the web server and should only be available via HTTPS.
    In the attached .htaccess all http calls are forwarded to https.
  - The database connection must be adjusted in the php script:
    ```
    $ strDbLocation = 'mysql: dbname = activities; host = localhost' ;
    $ strDbUser = 'root' ;
    $ strDbPassword = '' ;
    ```
  - The request must be sent as a POST request.
  - The answer is a string that returns the database query in CSV (comma-separated) format.
  - Example:
    ```
    SELECT synonym , first name , last name FROM `user` WHERE userid IN ( 1 , 4 )
    ```
    Results in phpMyAdmin :

Synonym	Vorname	Nachname
Softterrier	Falk	Hertel
hertel5	Gerd	Duplo

    The return of the script looks like this:
    ```
    Softterrier;Falk;Hertel
    hertel5;Gerd;Duplo
    ```
    The line break is implemented with '\ r \ n'. Problems may arise when interpreting on Unix systems.
- **remoteClient.php:**
  - that would be the call of a php client.
  - The url has to be adjusted. Example:
    ```
    $ url = 'https: // event_planner:
    e5V3E9n6T@127.0.0.1/ws/eventPlannerServer.php' ;
    ```
    Event_Planner is the username, e5V3E9n6T is the password for the http Basic Authentication. By replacing the .htpasswd it can be customized.
  - The select statement is submitted using the POST method:
    ```
    $ sqlString = "SELECT * FROM` user` WHERE userid IN (1, 4) " ;
    $ httpRequest_OBJ -> addPostFields ( array ( $ sqlString ));
    ```
  - The result is obtained by:
    ```
    $ result = $ httpRequest_OBJ -> send ();
    echo $ result -> getBody ();
    ```
- **.htaccess:**
  - Here is the absolute path to .htpasswd
    This must be adjusted

- **LAST_INSERT_ID ()** :
  For example, the return value is "LAST_INSERT_ID () 1" for the 1.
- **File upload:**
  The file *eventPlannerServerImages.php* (similar to SQL) must be called with a string as a POST request.
  For example a call to **save** 3 files would look like this (without line breaks):
  SAVE-IMAGE <- DATA -> target / Test.png <- DATA -> {IMAGE-STRING} <- FILE -> SAVE IMAGE <- DATA -> target / Test2.png <- DATA -> {IMAGE} STRING <- FILE -> SAVE IMAGE <- DATA -> target / Unbenannt.png < --DATA -> {IMAGE-STRING}
  So < **-DATA ->** is the delimiter of the data of a file and <- File -> the delimiter between several files.
  If only *one* file is transferred, it is not needed. {IMAGE-STRING} must be replaced with the file content.
  As a return, you receive the transferred bytes per file separated by a line break (same line break as in SQL Select, ie a "\ r \ n").
  If an error occurs, 0 is returned.
  If you would like to **download** one or more file (s) , you need to transfer the following string (here 2 files):
  GET-IMAGE <- DATA -> Source / Test.jpg <- FILE -> GET-IMAGE <- DATA -> Source / Test.jpg
  Here you get as a return per line the file content as a string.
  Please do not use special characters, umlauts and spaces in filenames.
  Please also pay attention to sufficient folder permissions.
  If files are not saved correctly, it could be because of magic_quotes. These are activated locally for me.