How to use the REST API

Version 1.0.0
04/28/2016

Santiago Fernández Madero A01320343
Oscar Gaytán Montero A01327292
Nahúm Andrés Espinosa Solís A01322193
Iván Eduardo García Aguirre A01325126

Introduction

This project was made in order to provide an useful tool to medical developers and game developers so that they can focus in helping kids with mental disabilities. Medical developers or any developer interested in the REST API can develop a tool (web, iOS, android, etc) that can consume data from the API that is feed from games. The main idea of this tool it would be that they read the patients, doctors and parents, and design an intranet where parents and doctors can see the kid progress on the games the doctor recommended.

Games can be developed with this API, they just need a token from the API in order to be able to POST data. They can post custom metrics with a measure value represented as a string. This value can be an score, time, tries, etc. This makes the API flexible because a lot of different games can be developed at top of it and with a lot of metrics can help the kids to improve.

Structure

Models

User:

- id
- name
- username (to login)
- mail
- password
- active (0 is inactive, 1 is active)
- token (This is returned when login done so that it can keep the session alive)
- kind (1 is a doctor, 2 is a patient’s parent)

Patient:

- id
- name
- username (to login from games so that metrics can be saved)
- password
- active
- token (To keep session alive from a game – to be done, accept multiple game sessions).

Game:

- id
- name
- developer (Name of developer – thinking on registering a developer so that they can do login and register their own games).
- mail
- token (This so that it can do request to the REST API)

Applications:

- id
- name
- developer (Name of developer – thinking on registering a developer so that they can do login and register their own applications).
- mail
- token (This so that it can do request to the REST API)

Game Metrics:

- id
- metric (name of the metric to measure, i.e “score”)
- value (Value to measure the metric, i.e for time the value would be seconds)
- id_patient
- date
- id_game

Requests

All the requests on this REST API are POST http request. They have to POST the parameters and every request is done to the same route XXX/controller.php.

User Sessions

login_user

This function helps developers to allow users (doctor or parents) to log in into the app.

Request

Data:

- application_token
- username
- password
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be login_user).

Returns a JSON:

- message
- success (true if login correctly, false if there was an error)
- token (to keep a session alive)
- user_type (1 doctor, 2 parent, -1 if there was an error)
- user_fullname
- id_user

Example:

```
{
 "message": "Login successful",
 "success": true,
 "token": "b65a057244bac59faca6962121ef46ec70c1352c",
 "user_type": "1",
 "user_fullname": "Santiago Fernandez",
 "id_user": "1"
}
```

register_user

This functions is to register new users (doctors or parents)

Request

Data:

- name
- mail
- username
- password
- cellphone
- kind (1 for doctors, 2 for parents)
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be register_user).

Returns a JSON:

- message
- success (true if registered correctly, false if there was an error)
- user_type (1 doctor, 2 parent, -1 if there was an error)
- user_fullname
- id_user

Example:

```
{
 "message": "The user was registered successful",
 "success": true,
 "token": "",
 "user_type": "2",
 "user_fullname": "Daniel Ruiz",
 "id_user": "7"
}
```

update_user

This function is to update users (doctors or parents)

Request

Data:

- id
- name

- mail
- username
- password
- cellphone
- application_token
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be update_user).

Returns a JSON:
- message
- success (true if updated correctly, false if there was an error)
- token (it will always be empty)

Example:
```
{
  "message": "The user was updated successful",
  "success": true,
  "token": ""
}
```

verify_user_session

This function is to verify if a session is alive with the stored token.

Data:
- application_token
- token
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be verify_user_session).

Returns a JSON:
- message
- success (true if correctly, false if there was an error)
- token (to keep a session alive)
- user_type (1 doctor, 2 parent, -1 if there was an error)
- user_fullname
- id_user

Example:

```
{
  "message": "The session is valid",
  "success": true,
  "token": "b65a057244bac59faca6962121ef46ec70c1352c",
  "user_type": "1",
  "user_fullname": "Santiago Fernandez",
  "id_user": "1"
}
```

logout_user

This function is to kill a current user session with a given token.

Data:

- application_token
- token
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be logout_user).

Returns a JSON:

- message
- success
- token (Empty variable)

Example:

```
{
 "message": "Logout from user successful",
 "success": true,
 "token": ""
}
```

deactivate_user

This function is to deactivate an user so now it will not be able to do login.

Data:

- application_token
- username
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be deactivate_user).

Returns a JSON:

- message
- success (True if correctly)
- token (Empty Value)

Example:
```
{
  "message": "User was deactivated correctly",
  "success": true,
  "token": ""
}
```

activate_user

This function is to activate an user so now it will be able to do login again.

Data:
- application_token
- username
-

Returns a JSON:
- message
- success (true if correctly)
- token (Empty Value)
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be activate_user).

Example:
```
{
  "message": "User was activated correctly",
  "success": true,
  "token": ""
}
```

verify_user_username_exists

This function helps developers when registering a new user to know if the current username already exists, with this developers will be able to tell the new user if it already exists.

Data:
- application_token
- username
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be verify_user_username_exists).

Returns a JSON:
- message
- success

- username

```
{
  "message": "The username already exists",
  "exists": true,
  "username": "santifdezm"
}
```

Patient Sessions

login_patient

This function helps developers to allow patients to log in into a game so that his doctor can look at his metrics.

Request

Data:

- gameToken
- username
- password
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be login_patient).

Returns a JSON:

- message
- success (true if correctly, false if there was an error)
- token (to keep a session alive)
- user_type (always -1, not needed)
- user_fullname
- id_user

Example:

```
{
  "message": "Login successful",
  "success": true,
  "token": "1cb85e5adde643ea0123663dda0abfa7abceed1c",
  "user_type": -1,
  "user_fullname": "Santiago Paciente Actualizado",
  "id_user": "3"
}
```

register_patient

This function is to register new patients. It is needed to know it’s doctor id to register the patient. It wouldn’t make sense if we have a patient registered with no doctor.

Request

Data:

- name
- username
- password
- application_token
- id_doctor
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be register_patient).

Returns a JSON:

- message
- success (true if correctly, false if there was an error)
- token (empty)

Example:

```
{
  "message": "The patient was registered successful",
  "success": true,
  "token": ""
}
```

update_patient

This function is to update patients.

Request

Data:

- name
- username
- password
- application_token
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be update_patient).

Returns a JSON:

- message
- success (true if login correctly, false if there was an error)
- token (empty)

Example:

```
{
  "message": "The patient was updated successful",
  "success": true,
  "token": ""
}
```

verify_patient_session

This function is to verify if a session is alive with the stored token.

Data:

- game_token
- token
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be verify_patient_session).

Returns a JSON:

- message
- success (true if correctly, false if there was an error)
- token (to keep a session alive)
- user_type (-1 if there was an error, not needed).
- user_fullname
- id_user

Example:

```
{
  "message": "The session is valid",
  "success": true,
  "token": "ee24df7c1153e14d15454c9cce6a05c41144f9b5",
  "user_type": -1,
  "user_fullname": "Patient James",
  "id_user": "7"
}
```

logout_patient

This function is to kill a current patient session with a given token.

Data:

- game_token
- token
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be logout_patient).

Returns a JSON:

- message
- success
- token (Empty variable)

Example:

```
{
  "message": "Logout from patient successful",
  "success": true,
  "token": ""
}
```

deactivate_patient

This function is to deactivate a patient so now it will not be able to do login.

Data:

- application_token
- username
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be deactivate_patient).

Returns a JSON:

- message
- success (True if correctly)
- token (Empty Value)

Example:

```
{
  "message": "Patient was deactivated correctly",
  "success": true,
  "token": ""
}
```

activate_patient

This function is to activate a patient so it is able to do login again.

Data:

- application_token
- username
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be activate_patient).

Returns a JSON:

- message
- success (True if correctly)
- token (Empty Value)

Example:

```
{
  "message": "Patient was activated correctly",
  "success": true,
  "token": ""
}
```

verify_patient_username_exists

This function helps developers when registering a new patient to know if the current username already exists, with this developers will be able to tell the new patient if it already exists.

Data:

- application_token
- username
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be verify_patient_username_exists).

Returns a JSON:

- message
- success
- username

```
{
  "message": "The username already exists",
  "exists": true,
  "username": "santifdezm"
}
```

Control Requests

register_application
This function is to register a new application to read and post data. This will give them a valid token to do so.

Data:

- name
- developer (his name)
- mail (developer's mail)
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be register_application).

Returns JSON:

- message
- success (true if it was registered, false if there was an error)
- token (token to do requests)

Example:
```
{
  "message": "The application was registered successful",
  "success": true,
  "token": "f41d5e5d1fe30b11da6a7fca0cb5f8f252680d88"
}
```

register_game
This function is to register a new game to post data. This will give them a valid token to do so.

Data:

- name
- developer (his name)
- mail (developer's mail)
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be register_game).

Returns JSON:

- message
- success (true if it was registered, false if there was an error)
- token (token to do requests)

Example:
```
{
  "message": "The game was registered successful",
  "success": true,
  "token": "f41d5e5d1fe30b11da6a7fca0cb5f8f252680d88"
}
```

add_game_patient_metric

This request is to register a new game metric to a patient so that the doctor can keep track of the his progress.

Data:

- metric (Name of metric, i.e: score)
- id_patient
- game_token
- date - The format for the date should be as follows:
  - YYYY-MM-DD H:MM:SS.mm
  - Example: 2018-03-22 12:02:38.000
  - The date is requested so that we don't have problems with the different time zones, so that the date is precise for when it really was taken that metric.
- value (The value to measure that metric, if it is time it should be in seconds)
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be add_game_patient_metric).

Result JSON:

- message
- success (true if it was registered, false if there was an error)
- token (token to do requests)

Example:
```
{
  "message": "The game was registered successful",
  "success": true,
  "token": "f41d5e5d1fe30b11da6a7fca0cb5f8f252680d88"
}
```

get_all_games

This request will return all the games registered in the database so that doctors can see what games are available with this API.

Data:

- application_token
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be get_all_games).

Result JSON:

- message
- success (true if it was successful, false if there was an error)
- array
    - [{ id, name, developer, mail }]

Example:

```
{
  "message": "Games fetched correctly",
  "success": true,
  "array": [
    {
      "id": "1",
      "name": "rompecabezas",
      "developer": "Santiago",
      "mail": "santiatlas11@hotmail.com"
    },
    {
      "id": "2",
      "name": "Sudoku",
      "developer": "Estefania",
      "mail": "estefi_ma94@hotmail.com"
    },
    {
      "id": "3",
      "name": "Snake",
      "developer": "Diego Ramirez",
      "mail": "cooldeveloper@hotmail.com"
    }
  ]
}
```

get_all_patient_game_metrics

This request gets all the patient metrics for the given game id.

Data:

- application_token
- id_patient
- id_game
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be get_all_patient_game_metrics).

Result JSON:

- message
- success (true if it was succesful, false if there was an error)
- array
  - [{ id, id_game, id_patient, patient_name, game_name, metric, value, date }]

Example:

```
{
 "message": "Games fetched correctly",
 "success": true,
 "array": [
  {
   "id": "5",
   "id_game": "3",
   "id_patient": "7",
   "patient_name": "Patient James",
   "game_name": "Snake",
   "metric": "SCORE",
   "value": "238",
   "date": "2016-04-28 09:31:38"
  },
  {
   "id": "6",
   "id_game": "3",
   "id_patient": "7",
   "patient_name": "Patient James",
   "game_name": "Snake",
   "metric": "SCORE",
   "value": "302",
   "date": "2016-04-28 09:41:38"
  }
 ]
}
```

get_all_patient_games

This request will return the games that a patient has played in the past.

Data:

- application_token
- id_patient
- id_game
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be get_all_patient_games)

Result JSON:

- message
- success (true if it was succesful, false if there was an error)
- array
    - [{ id, name, developer, mail }]

Example:

```
{
 "message": "Games fetched correctly",
 "success": true,
 "array": [
  {
   "id": "3",
   "name": "Snake",
   "developer": "Diego Ramirez",
   "mail": "cooldeveloper@hotmail.com"
  }
 ]
}
```

get_patient_doctors

This request will get all the doctors for a given patient.

Data:

- application_token
- id_patient
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be get_patient_doctors)

Result JSON:

- message
- success (true if it was succesful, false if there was an error)
- array
    - [{ id, name, mail, cellphone, active, kind }]

Example:

```
{
 "message": "Doctors fetched correctly",
 "success": true,
 "array": [
  {
   "id": "8",
   "name": "Juan Perez",
   "mail": "juan@hotmail.com",
   "cellphone": "4444444444",
   "active": "1",
   "kind": "1"
  }
 ]
}
```

get_patient_parents

This request returns all the parents registered for the given patient

Data:

- application_token
- id_patient
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be get_patient_parents)

Result JSON:

- message
- success (true if it was succesful, false if there was an error)
- array
    - [{ id, name, mail, cellphone, active, kind }]

Example:

```
{
 "message": "Parents fetched correctly",
 "success": true,
 "array": [
  {
   "id": "8",
   "name": "Juan Perez",
   "mail": "juan@hotmail.com",
   "cellphone": "4444444444",
   "active": "1",
   "kind": "2"
  }
 ]
}
```

get_all_doctor_patients

This request returns all the patients for the given doctor.

Data:

- application_token
- id_doctor
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be get_all_doctor_patients)

Result JSON:

- message
- success (true if it was succesful, false if there was an error)
- array
    - [{ id, name, username, active}]

Example:

```
{
  "message": "Patients fetched correctly",
  "success": true,
  "array": [
    {
      "id": "7",
      "name": "Patient James",
      "username": "patientjames",
      "active": "1"
    }
  ]
}
```

get_all_parent_patients

This request returns all the patients for the given doctor.

Data:

- application_token
- id_parent
- funcion (This value is so that the controller knows what function you want to execute, in this case it has to be get_all_parent_patients)

Result JSON:

- message
- success (true if it was succesful, false if there was an error)
- array
    - [{ id, name, username, active}]

Example:

```
{
  "message": "Patients fetched correctly",
  "success": true,
  "array": [
    {
      "id": "7",
      "name": "Patient James",
      "username": "patientjames",
      "active": "1"
    }
  ]
}
```