Diario di lavoro

Luogo: Canobbio

Data: 26.11.2019

Lavori svolti

Ho finito le interfacce grafiche, creando la pagina di login, quella di gestione utenti e quella di gestione dell stato del gestore di frequenze. Queste hanno diversi controller i quali usano i servizi per richiedere al back-end i dati. Esempio di servizio di base:

```
app.factory('GeneratorStatusService', ['$http', function($http) {
var config = {headers: [{'Access-Control-Allow-Origin': ' *' }, {'Access-
Control-Allow-Credentials':'true'}]};
var service = {};
var address = "localhost";
var port = 8080;
var baseApi = "/freqline-be"
var urlBase = "http://" + address + ":" + port + baseApi;
urlBase += "/action/generatorStatus";
$http.defaults.withCredentials = true;
service.getMicGeneratorStatus = function() {
    let url = urlBase + "?t=" + new Date().getTime();
    let data = {};
    return $http.get(url, data, config).then(function (response) {
        return response.data;
    }, function(error) {
        if (error.status == 401) {
            $rootScope.error = error;
            $location.path('/401');
            return error;
        } else {
            return error;
        }
    })
};
return service;
}]);
```

Che serve per richiedere al back-end lo stato del generatore. Poi vi è il controller che setta lo stato nell'interfaccia grafica.

```
app.controller('StatusController', ['$scope', '$location',
'GeneratorFrequenceService', 'GeneratorMicService',
'GeneratorStatusService', function ($scope, $location,
```

```
generatorFrequenceService, generatorMicService, generatorStatusService) {
    $scope.load = function() {
        generatorStatusService.getGeneratorStatus().then(function(data) {
            $scope.status = data.message;
        });
    }
    $scope.setStatus = function(status) {

generatorStatusService.setGeneratorStatus(status).then(function(data) {
            $scope.load();
        });
    }
    $scope.load();
}]);
```

Questo fa il load dei dati dal server, e quando viene aggiornato il dato dalla grafica inviato al back-end e riaggiornato i dati della grafica.
Poi mi sono accorto che il database ed il back-end avevano il Timer del microfono (rappresentazione sql e java) che erano di tipo datetime. Cosa che non segue propriamente le convenzioni, quindi ho cambiato il tipo di dato in int. Questo per le classi:

- dbdao.DbMicDao
- model.Mic
- modeljson.MicJson
- queries.GeneratorQuery
-
- servlets.action.GeneratorMicServlet
- servlets.data.MicServlet

Poi ho iniziato ad installare il raspberry. Ho scaricato l'immagine dal sito (https://www.raspberrypi.org/downloads/raspbian/), che ho scritto sulla micro SD con il comando:

```
sudo dd bs=1m if=~/tmp/2019-09-26-raspbian-buster-lite.img of=/dev/disk2
conv=sync
```

Dopo di che ho avviato il raspberry ed ho provato ad aggiornare i repository. Prima di fare l'aggiornamenot ho configurato il proxy file `/etc/apt/apt.conf` aggiungere le seguenti linee:

```
Acquire::http::Proxy "http://username:password@10.20.0.1:8080";
Acquire::https::Proxy "https://username:password@10.20.0.1:8080";
```

Poi ho provato ad aggiornare i repository:

```
apt update
```

operazione che è fallita siccome la data era sbagliata, ho cercato come farla e sul sito: https://raspberrypi.stackexchange.com/questions/59860/time-and-timezone-issues-on-pi, ho trovato la soluzione, con il comando:

```
sudo timedatectl set-time 'yyyy-mm-dd hh-mm'
```

. Poi sono riuscito ad aggiornare i repository (apt update), ha funzionato, poi ho aggiornato il sistema operativo.

```
apt upgrade
```

Problemi riscontrati e soluzioni adottate

Aggiornamento non riuscito, siccome data vecchia.
https://raspberrypi.stackexchange.com/questions/59860/time-and-timezone-issues-on-pi

Punto della situazione rispetto alla pianificazione

Sono leggermente indietro con la pianificazione, non ho finito l'attivita 33, ma passo alla prossimo, la grafica (userfrendly) la faccio alla fine.

Programma di massima per la prossima giornata di lavoro

Finire di installare il raspberry.