

```
665 lines (501 sloc) 28.6 KB
Ex:

public static User register(User user, string email, string password)
{
    user.setEmail(email);
    user.setPassword(password);

    using HttpResponseMessage response = await httpClient.PostAsync(userURL + "/register", user);
    if((int)response.StatusCode==404)
    {
        user.setEmail(NULL);
        user.setPassword(NULL);
        return false; //Deal with post failure(other status codes too)
    }
    using HttpResponseMessage response = httpClient.GetAsync(userURL + "?email=" + email);
    var jsonResponse = await response.Content.ReadAsStringAsync();
    JsonNode jsonNode = JsonNode.Parse(jsonResponse);
    user.setId(Convert.ToInt32(jsonNode["id"]));
    user.setLogged();

    return true;
}

DataManager

• getData(int,int):Motion - This method should allow the user to retrieve data that they've collected from the data database. The data
retrieved will be returned in a Json format that should then be parsed and used to create a Motion instance that the method then retu
```

id: string post json

change the phone number, email, and birthday at the same time

```
665 lines (501 sloc) 28.6 KB
failure.
Ex:

public Motion getData(int id, int motionId)//user id or what may be necessary to identify the data
{
    HttpResponseMessage response = await client.GetAsync(dataURL + "/" + id + "motionID");

    if(response.IsSuccessStatusCode){
        Motion motion = new Motion();
        string json = await response.Content.ReadAsStringAsync();
        JArray sensorDataArray = JsonConvert.DeserializeObject<JArray>(json);

        foreach (JObject sensorData in sensorDataArray)
        {
            //store each entry of the data returned by the API in Motion Records

            /*double timestamp = sensorData["timestamp"].Value<double>();

            foreach (var sensor in sensorData)
            {
                if (sensor.Key == "timestamp")
                    continue;

                JObject sensorValues = sensor.Value.Value<JObject>();

                Data data = new Data
                {
                    SensorId = sensor.Key,
                    X = sensorValues["X"].Value<double>(),
                    Y = sensorValues["Y"].Value<double>(),
                    Z = sensorValues["Z"].Value<double>()
                }
            }
        }
    }
}
```

only getAlldata

Don't get data when collect data

user intends to collect data for. This method returns a bool based on its success  
Ex:

```
public bool collectData(int userId, enum movementType){  
    using StringContent jsonContent = new(  
        JsonSerializer.Serialize(new  
        {  
            userID = $"{userId}",  
            label = $"{movementType}"  
        }  
    ),  
    Encoding.UTF8,  
    "application/json");  
  
    HttpResponseMessage response = await client.PostAsync(sensorsURL + "/collect/start", jsonContent);  
    if(response.StatusCode == "404") return false; //Deal with other possible status codes  
  
    this.seIsCollecting(true);  
    this.thread = new Thread(getDataLoop(userId, motionId));  
    this.thread.Start();  
    return true;  
}  
  
private void getDataLoop(int userId, int motionId)  
{  
    while (this.isCollecting)  
    {  
        Motion motion = this.getData(userId, motionId);  
        //draw graphs  
  
        Thread.Sleep(2000);  
    }  
}
```

discard post id and create time

equipment is available then the user should be able to connect to it, otherwise they should be warned of it's unavailability. This method takes as parameters a string and two integers. The string represents the type of equipment, and the integer represents the port of the sensor. This method returns a bool based on it's success

Ex:

```
public bool connectEquipment(string type, int ip, int port){
    using StringContent jsonContent = new(
        JsonSerializer.Serialize(new
        {
            ip = $"{ip}",
            type = $"{type}",
            port = $"{port}"
        }),
        Encoding.UTF8,
        "application/json");

    HttpResponseMessage response = await client.PostAsync(sensorsURL + "/connect" , jsonContent);

    return response.IsSuccessStatusCode //Deal with the possibility of failure to connect
}
```

- **disconnectEquipment():bool** - This method is supposed to allow the user currently calling it to disconnect from the equipment. It should either only be allowed to be called by a user that successfully connected to the equipment (it was available when they called it), or the user should be informed, when they call it, that they need to connect to the equipment first. This method takes, as a parameter, an integer representing the id of the equipment the user is calling in, and returns a bool based on it's success.



There should also be an id to post