**FIT5046  
Mobile Distributed System**

Assignment 1

Name: VU ANH UYEN LE

Student ID: 27685713

Date: 26 Mar 2018

Table of Contents

[1. Task 1 3](#_Toc509838084)

[1.1 SQL Code 3](#_Toc509838085)

[1.2 Screenshot data 3](#_Toc509838086)

[2. Task 2 5](#_Toc509838087)

[3. Task 3 6](#_Toc509838088)

[3.1 GET methods 6](#_Toc509838089)

[3.2 Dynamic query for ResidentBySurnameAndEmail 17](#_Toc509838090)

[3.3 Dynamic query for ResidentByEmailAndHour 18](#_Toc509838091)

[3.4 Static query for ResidentByEmailAndHour 19](#_Toc509838092)

[4. Task 4 20](#_Toc509838093)

[4.1 The hourly power usage (kWh) of the appliance 20](#_Toc509838094)

[4.2 The hourly power usage (kWh) of ALL THREE appliances 23](#_Toc509838095)

[4.3 A list of hourly power usage (kWh) of ALL THREE appliances for all the residents 25](#_Toc509838096)

[4.4 The highest hourly power consumption 28](#_Toc509838097)

[5. Task 5 28](#_Toc509838098)

[5.1 Daily Usage of Appliances 28](#_Toc509838099)

[5.2 Hourly/Daily Usage 31](#_Toc509838100)

1. Task 1
   1. SQL Code

|  |  |
| --- | --- |
| TABLE ELECTRICITYUSAGE | CREATE TABLE ELECTRICITYUSAGE (USAGEID INTEGER NOT NULL, RESID INTEGER, "DATE" DATE, HOURUSAGE INTEGER, FRIDGE DOUBLE, AIRCONDITIONER DOUBLE, WASHINGMACHINE DOUBLE, TEMPERATURE DOUBLE, PRIMARY KEY (USAGEID));  CONSTRAINT FK\_UsageId FOREIGN KEY (RESD)     REFERENCES Resident (RESD); |
| TABLE RESIDENT | CREATE TABLE RESIDENT (RESID INTEGER NOT NULL, FIRSTNAME VARCHAR(50), SURNAME VARCHAR(50), DOB DATE, ADDRESS VARCHAR(100), POSTCODE VARCHAR(50), EMAIL VARCHAR(50), MOBILE VARCHAR(50), NUMBEROFRESIDENT INTEGER, ENERGYPROVIDER VARCHAR(50), PRIMARY KEY (RESID)); |
| TABLE RESIDENTCREDENTIALS | CREATE TABLE RESIDENTCREDENTIALS (USERNAME VARCHAR(50) NOT NULL, RESID INTEGER, PASSWORD VARCHAR(50), REGISTRATIONDATE DATE, PRIMARY KEY (USERNAME));  CONSTRAINT FK\_CreId FOREIGN KEY (RESD)     REFERENCES Resident (RESD); |

* 1. Screenshot data

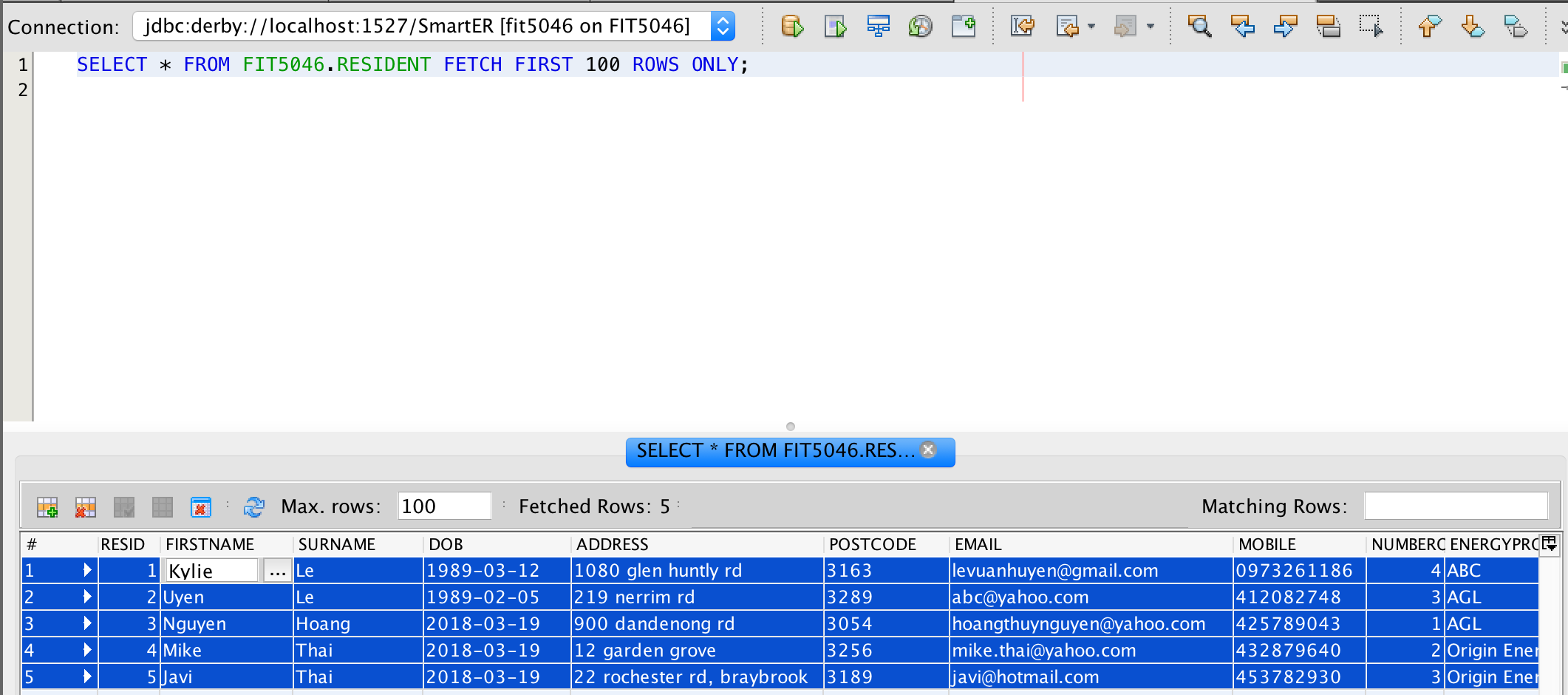


Figure 1.1: A RESTful web service is created

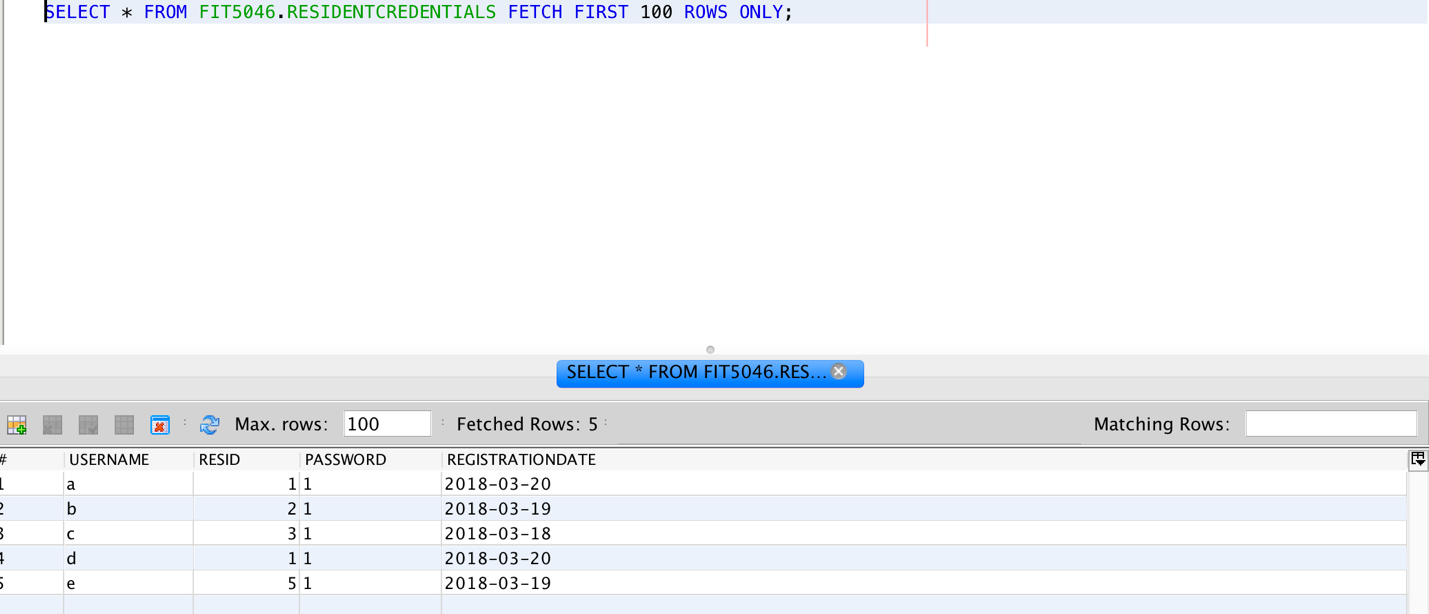


Figure 1.2: A RESTful web service is created

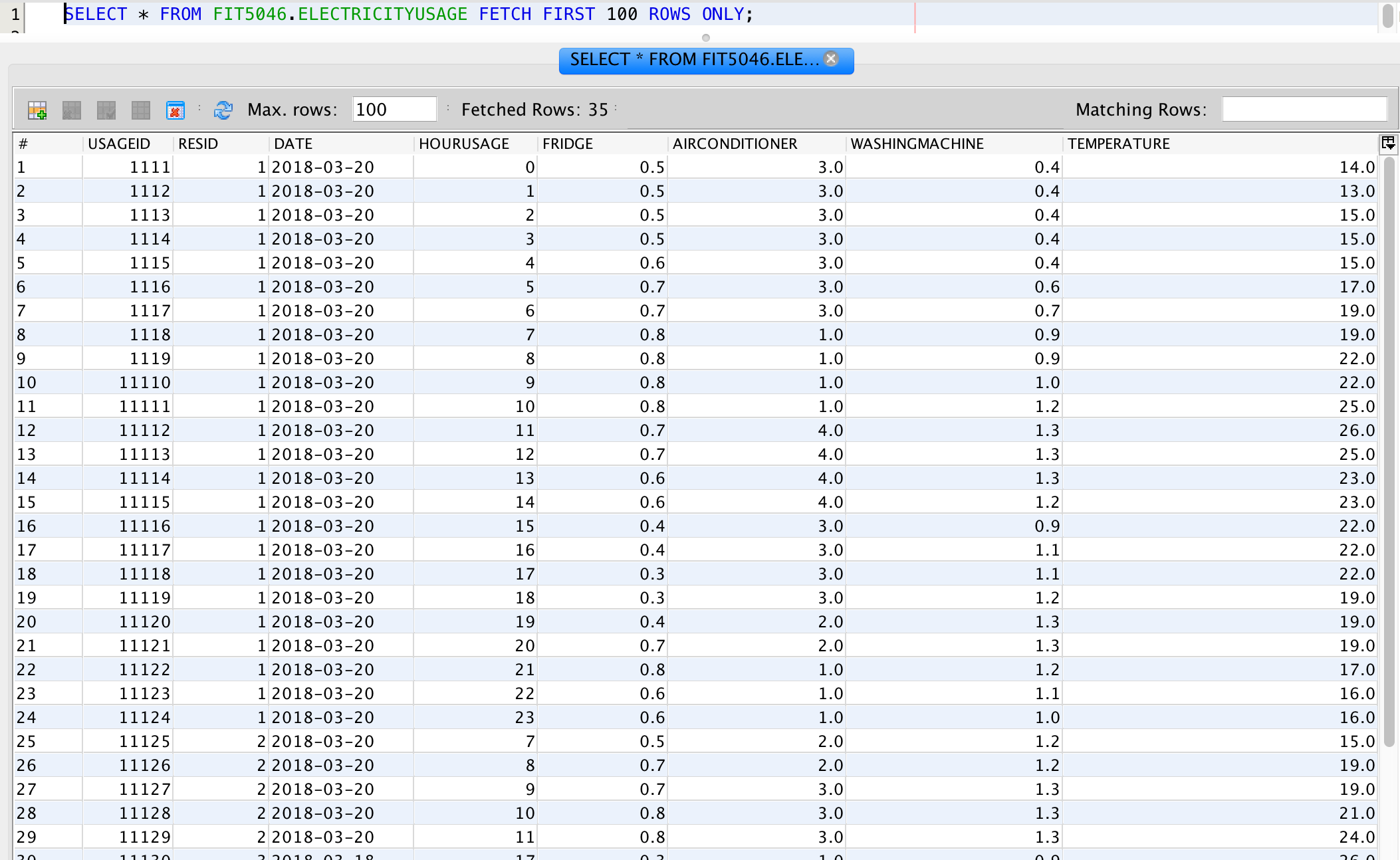
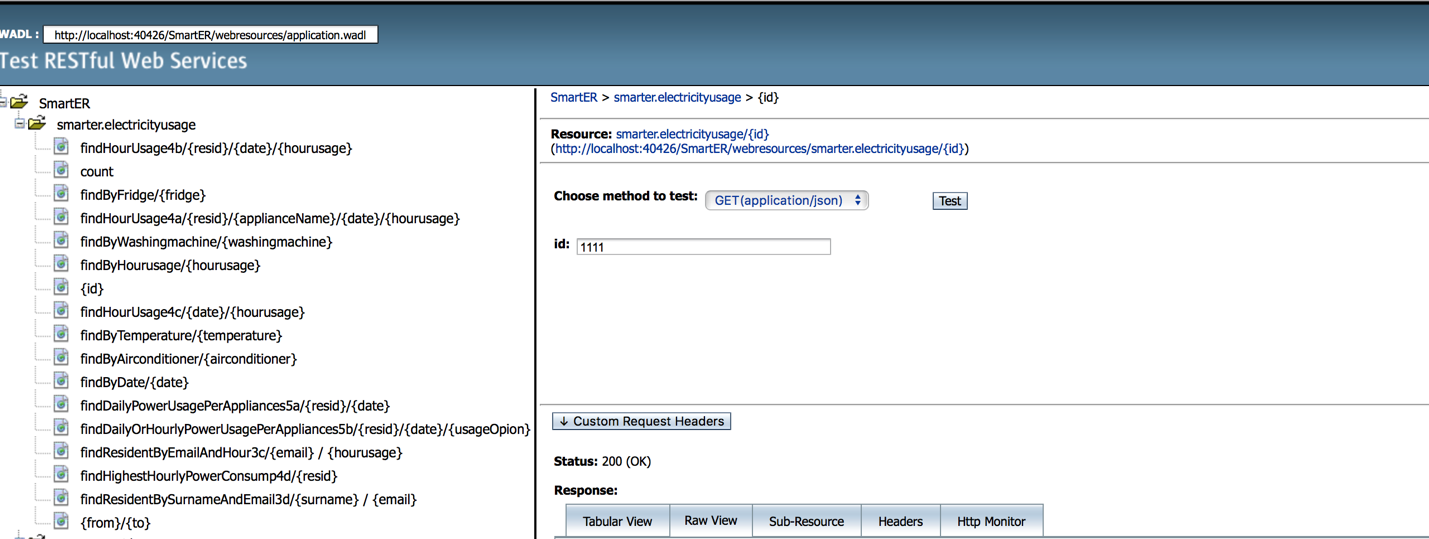


Figure 1.3: A RESTful web service is created

1. Task 2



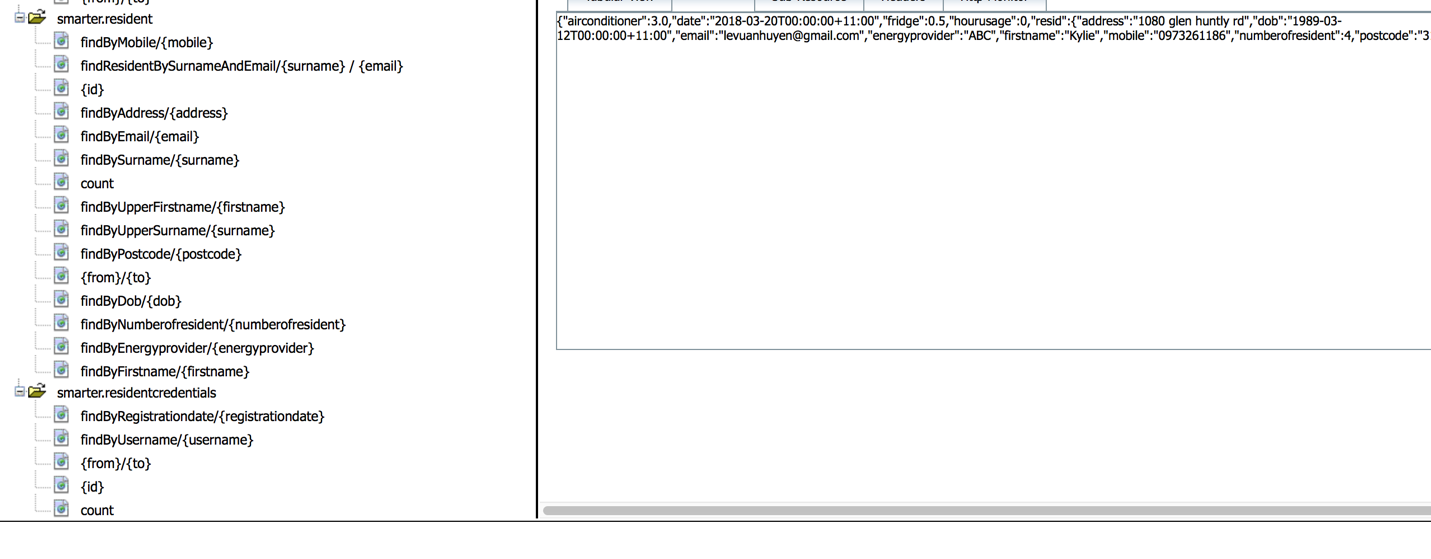


Figure 2: A RESTful web service is created

1. Task 3
   1. GET methods

|  |  |
| --- | --- |
| RESIDENT TABLE | ResidentFacadeREST.java |
| //Get firstname | @GET  @Path("findByFirstname/{firstname}")  @Produces({"application/json"})  public List<Resident> findByFirstname(@PathParam("firstname") String  firstname) {  Query query = em.createNamedQuery("Resident.findByFirstname");  query.setParameter("firstname", firstname);  return query.getResultList();  } |
| //GET surname | @GET  @Path("findBySurname/{surname}")  @Produces({"application/json"})  public List<Resident> findBySurname(@PathParam("surname") String  surname) {  Query query = em.createNamedQuery("Resident.findBySurname");  query.setParameter("surname", surname);  return query.getResultList();  } |
| //Get DOB findByDob | @GET  @Path("findByDob/{dob}")  @Produces({"application/json"})  public List<Resident> findByDob(@PathParam("dob") Date  dob) {  Query query = em.createNamedQuery("Resident.findByDob");  query.setParameter("dob", dob);  return query.getResultList();  } |
| // Get address findByAddress | @GET  @Path("findByAddress/{address}")  @Produces({"application/json"})  public List<Resident> findByAddress(@PathParam("address") String  address) {  Query query = em.createNamedQuery("Resident.findByAddress");  query.setParameter("address", address);  return query.getResultList();  } |
| //Get Postcode findByPostcode | @GET  @Path("findByPostcode/{postcode}")  @Produces({"application/json"})  public List<Resident> findByPostcode(@PathParam("postcode") String  postcode) {  Query query = em.createNamedQuery("Resident.findByPostcode");  query.setParameter("postcode", postcode);  return query.getResultList();  } |
| //Get email findByEmail | @GET  @Path("findByEmail/{email}")  @Produces({"application/json"})  public List<Resident> findByEmail(@PathParam("email") String  email) {  Query query = em.createNamedQuery("Resident.findByEmail");  query.setParameter("email", email);  return query.getResultList();  } |
| //Getphone findByMobile | @GET  @Path("findByMobile/{mobile}")  @Produces({"application/json"})  public List<Resident> findByMobile(@PathParam("mobile") String  mobile) {  Query query = em.createNamedQuery("Resident.findByMobile");  query.setParameter("mobile", mobile);  return query.getResultList();  } |
| //Get number of resident findByNumberofresident | @GET  @Path("findByNumberofresident/{numberofresident}")  @Produces({"application/json"})  public List<Resident> findByNumberofresident(@PathParam("numberofresident") Integer  numberofresident) {  Query query = em.createNamedQuery("Resident.findByNumberofresident");  query.setParameter("numberofresident", numberofresident);  return query.getResultList();  } |
| //Get energy provider findByEnergyprovider | @GET  @Path("findByEnergyprovider/{energyprovider}")  @Produces({"application/json"})  public List<Resident> findByEnergyprovider(@PathParam("energyprovider") String  energyprovider) {  Query query = em.createNamedQuery("Resident.findByEnergyprovider");  query.setParameter("energyprovider", energyprovider);  return query.getResultList();  } |

|  |  |
| --- | --- |
| Residentcredentials Table | ResidentcredentialsFacadeREST.java |
| //Get username findByUsername | @GET  @Path("findByUsername/{username}")  @Produces({"application/json"})  public List<Residentcredentials> findByUsername(@PathParam("username") String  username) {  Query query = em.createNamedQuery("Residentcredentials.findByUsername");  query.setParameter("username", username);  return query.getResultList();  } |
| //Get registration date findByRegistrationdate | @GET  @Path("findByRegistrationdate/{registrationdate}")  @Produces({"application/json"})  public List<Residentcredentials> findByRegistrationdate(@PathParam("registrationdate") Date  registrationdate) {  Query query = em.createNamedQuery("Residentcredentials.findByRegistrationdate");  query.setParameter("registrationdate", registrationdate);  return query.getResultList();  } |

|  |  |
| --- | --- |
| Electricityusage Table | ElectricityusageFacadeREST.java |
| // Get date findByDate | @GET  @Path("findByDate/{date}")  @Produces({"application/json"})  public List<Electricityusage> findByDate(@PathParam("date") Date date) {  Query query = em.createNamedQuery("Electricityusage.findByDate");  query.setParameter("date", date);  return query.getResultList();  } |
| // Get hour findByHourusage | @GET  @Path("findByHourusage/{hourusage}")  @Produces({"application/json"})  public List<Electricityusage> findByHourusage(@PathParam("hourusage") Integer hourusage) {  Query query = em.createNamedQuery("Electricityusage.findByHourusage");  query.setParameter("hourusage", hourusage);  return query.getResultList();  } |
| // Get findByFridge | @GET  @Path("findByFridge/{fridge}")  @Produces({"application/json"})  public List<Electricityusage> findByFridge(@PathParam("fridge") Double fridge) {  Query query = em.createNamedQuery("Electricityusage.findByFridge");  query.setParameter("fridge", fridge);  return query.getResultList();  } |
| // Get findByAirconditioner | @GET  @Path("findByAirconditioner/{airconditioner}")  @Produces({"application/json"})  public List<Electricityusage> findByAirconditioner(@PathParam("airconditioner") Double airconditioner) {  Query query = em.createNamedQuery("Electricityusage.findByAirconditioner");  query.setParameter("airconditioner", airconditioner);  return query.getResultList();  } |
| // Get findByWashingmachine | @GET  @Path("findByWashingmachine/{washingmachine}")  @Produces({"application/json"})  public List<Electricityusage> findByWashingmachine(@PathParam("washingmachine") Double washingmachine) {  Query query = em.createNamedQuery("Electricityusage.findByWashingmachine");  query.setParameter("washingmachine", washingmachine);  return query.getResultList();  } |
| // Get findByTemperature | @GET  @Path("findByTemperature/{temperature}")  @Produces({"application/json"})  public List<Electricityusage> findByTemperature(@PathParam("temperature") Double temperature) {  Query query = em.createNamedQuery("Electricityusage.findByTemperature");  query.setParameter("temperature", temperature);  return query.getResultList();  } |

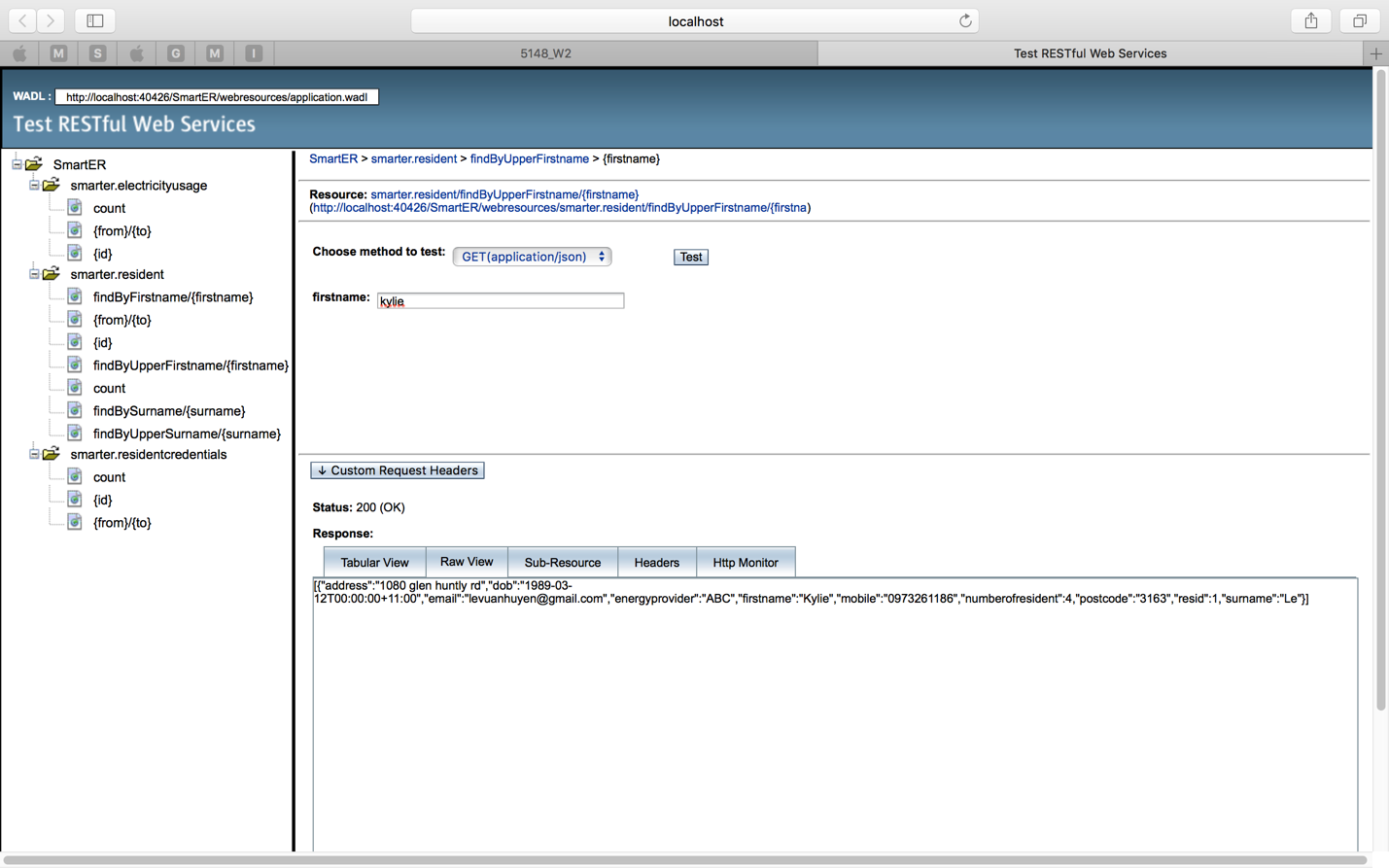


Figure 3.1.1: Query First name on Resident Table

Output:

[{"address":"1080 glen huntly rd","dob":"1989-03-12T00:00:00+11:00","email":"levuanhuyen@gmail.com","energyprovider":"ABC","firstname":"Kylie","mobile":"0973261186","numberofresident":4,"postcode":"3163","resid":1,"surname":"Le"}]

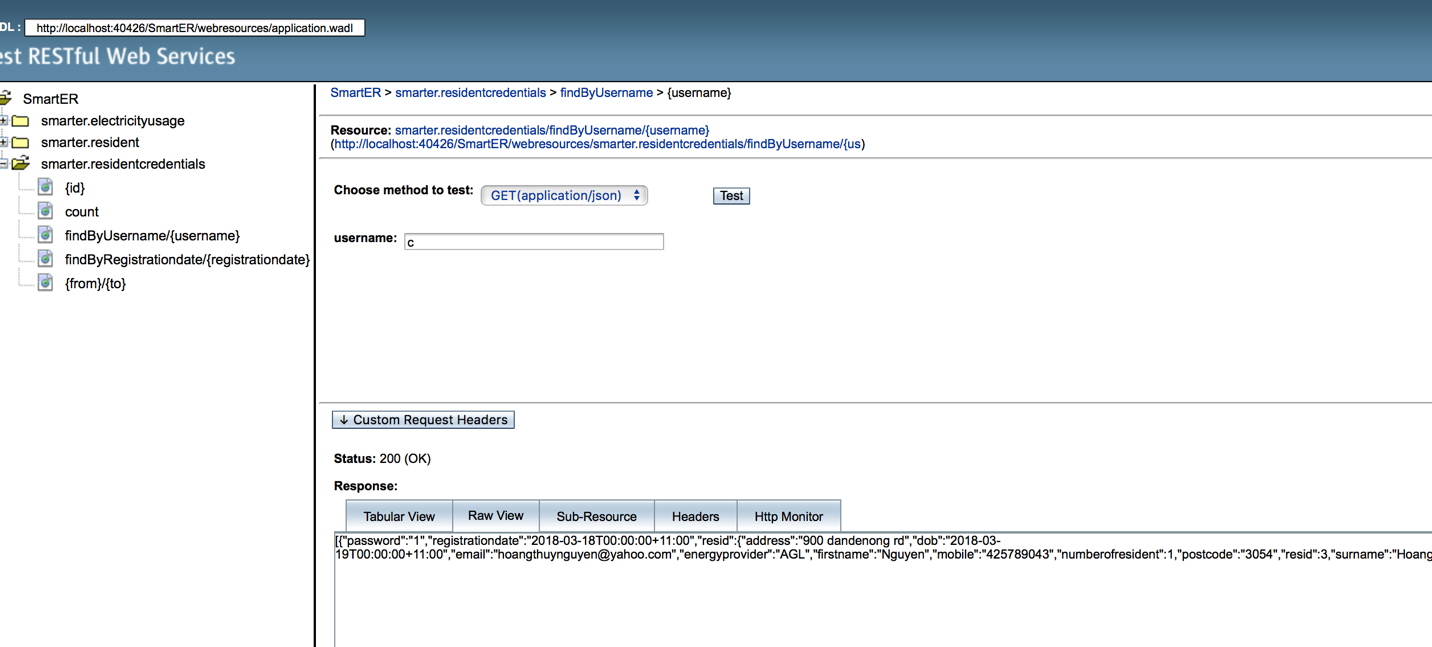


Figure 3.1.2: Query Username on Residentcredentials Table

Output:

{"password":"1","registrationdate":"2018-03-18T00:00:00+11:00","resid":{"address":"900 dandenong rd","dob":"2018-03-19T00:00:00+11:00",

"email":"hoangthuynguyen@yahoo.com","energyprovider":"AGL","firstname":"Nguyen","mobile":"425789043","numberofresident":1,"postcode":"3054","resid":3,"surname":"Hoang"},"username":"c"}]

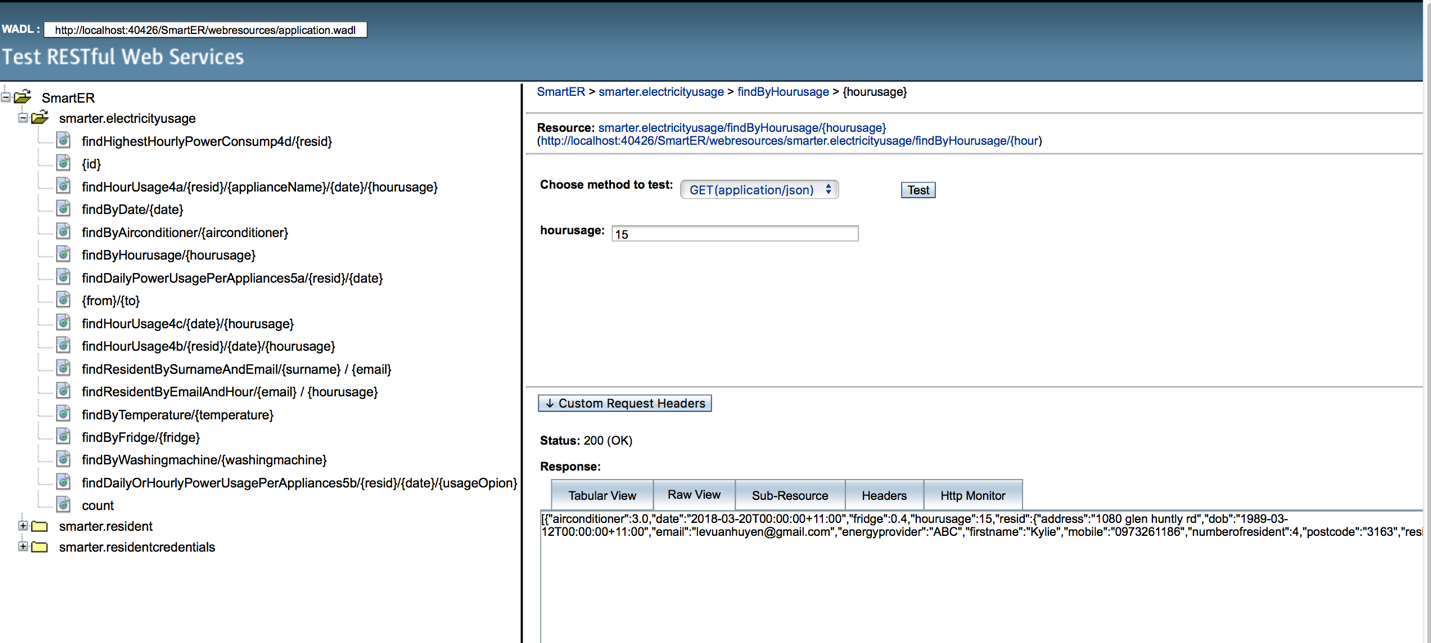


Figure 3.1.3: Query hourusage on Electricityusage

Output:

[{"airconditioner":3.0,"date":"2018-03-20T00:00:00+11:00","fridge":0.4,"hourusage":15,"resid":{"address":"1080 glen huntly rd","dob":"1989-03-12 T00:00:00+11:00", "email":"levuanhuyen@gmail.com","energyprovider":"ABC","firstname":"Kylie","mobile":"0973261186","numberofresident":4,"postcode":"3163","resid":1,"surname":"Le"},"temperature":22.0,"usageid":11116,"washingmachine":0.9}]

* 1. Dynamic query for ResidentBySurnameAndEmail

|  |  |
| --- | --- |
| Restful method | @GET  @Path("findResidentBySurnameAndEmail/{surname} / {email}")  @Produces({"application/json"})  public List<Resident> findResidentBySurnameAndEmail(@PathParam("surname") String surname, @PathParam("email") String email) {  TypedQuery<Resident> q = em.createQuery("SELECT r FROM Resident r WHERE r.surname = :surname AND r.email = :email", Resident.class);  q.setParameter("surname", surname);  q.setParameter("email", email);  return q.getResultList();  } |

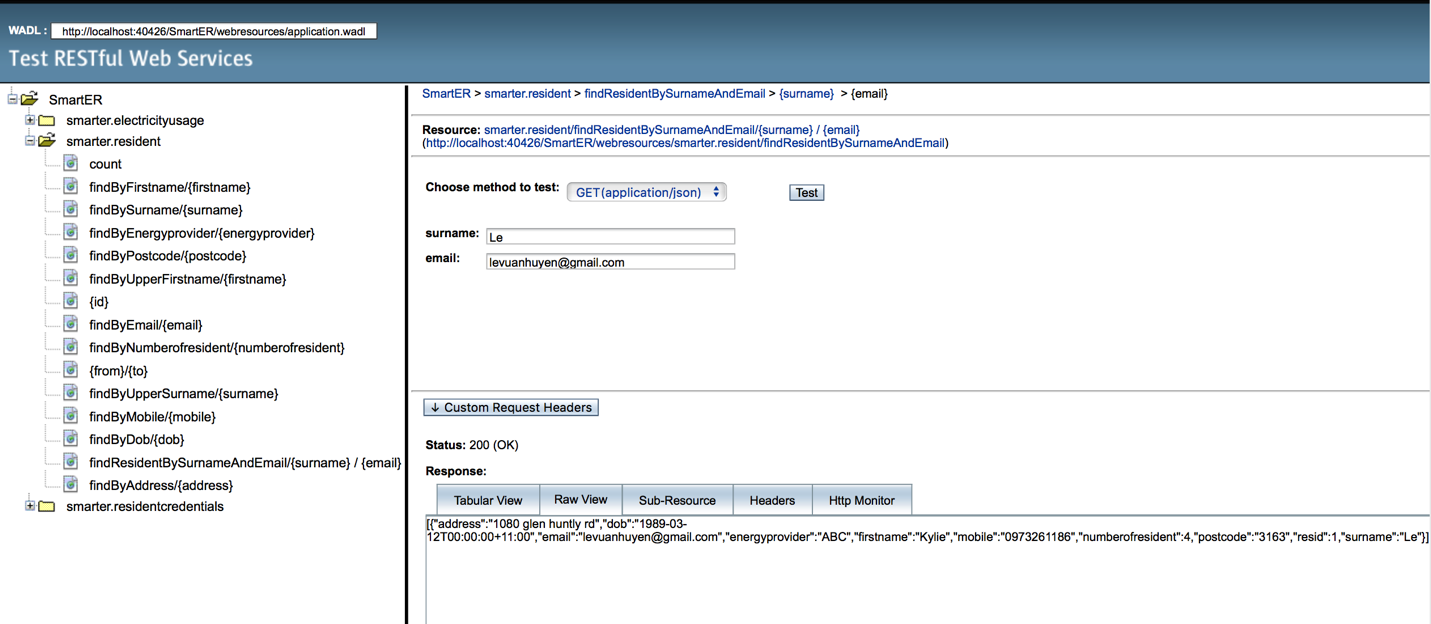


Figure 3.2: Dynamic Query by Surname and Email

* 1. Dynamic query for ResidentByEmailAndHour

|  |  |
| --- | --- |
| Restful method | @GET  @Path("findResidentByEmailAndHour/{email} / {hourusage}")  @Produces({"application/json"})  public List<Electricityusage> findResidentByEmailAndHour(@PathParam("email") String email, @PathParam("hourusage") Integer hourusage) {  TypedQuery<Electricityusage> q = em.createQuery("SELECT e FROM Electricityusage e WHERE e.resid.email = :email AND e.hourusage = :hourusage", Electricityusage.class);    q.setParameter("email", email);  q.setParameter("hourusage", hourusage);  return q.getResultList();  } |



Figure 3.3: Dynamic Query by Email and Hour

Output: [{"airconditioner":3.0,"date":"2018-03-20T00:00:00+11:00","fridge":0.5,"hourusage":3,"resid":{"address":"1080 glen huntly rd","dob":"1989-03-12T00:00:00+11:00","email":"levuanhuyen@gmail.com","energyprovider":"ABC","firstname":"Kylie","mobile":"0973261186","numberofresident":4,"postcode":"3163","resid":1,"surname":"Le"},"temperature":15.0,"usageid":1114,"washingmachine":0.4}]

* 1. Static query for ResidentByEmailAndHour

|  |  |
| --- | --- |
| Restful method | @GET  @Path("findResidentBySurnameAndEmail3d/{surname}/{email}")  @Produces({"application/json"})  public List<Electricityusage> findResidentBySurnameAndEmail3d(@PathParam("surname") String surname, @PathParam("email") String email) {  Query q = em.createNamedQuery("Electricityusage.findResidentBySurnameAndEmail3d");  q.setParameter("surname", surname);  q.setParameter("email", email);  return q.getResultList();  } |
| Query in entity | @NamedQuery(name = "Electricityusage.findResidentBySurnameAndEmail3d", query = "SELECT e FROM Electricityusage e WHERE e.resid.surname = :surname AND e.resid.email = :email") |

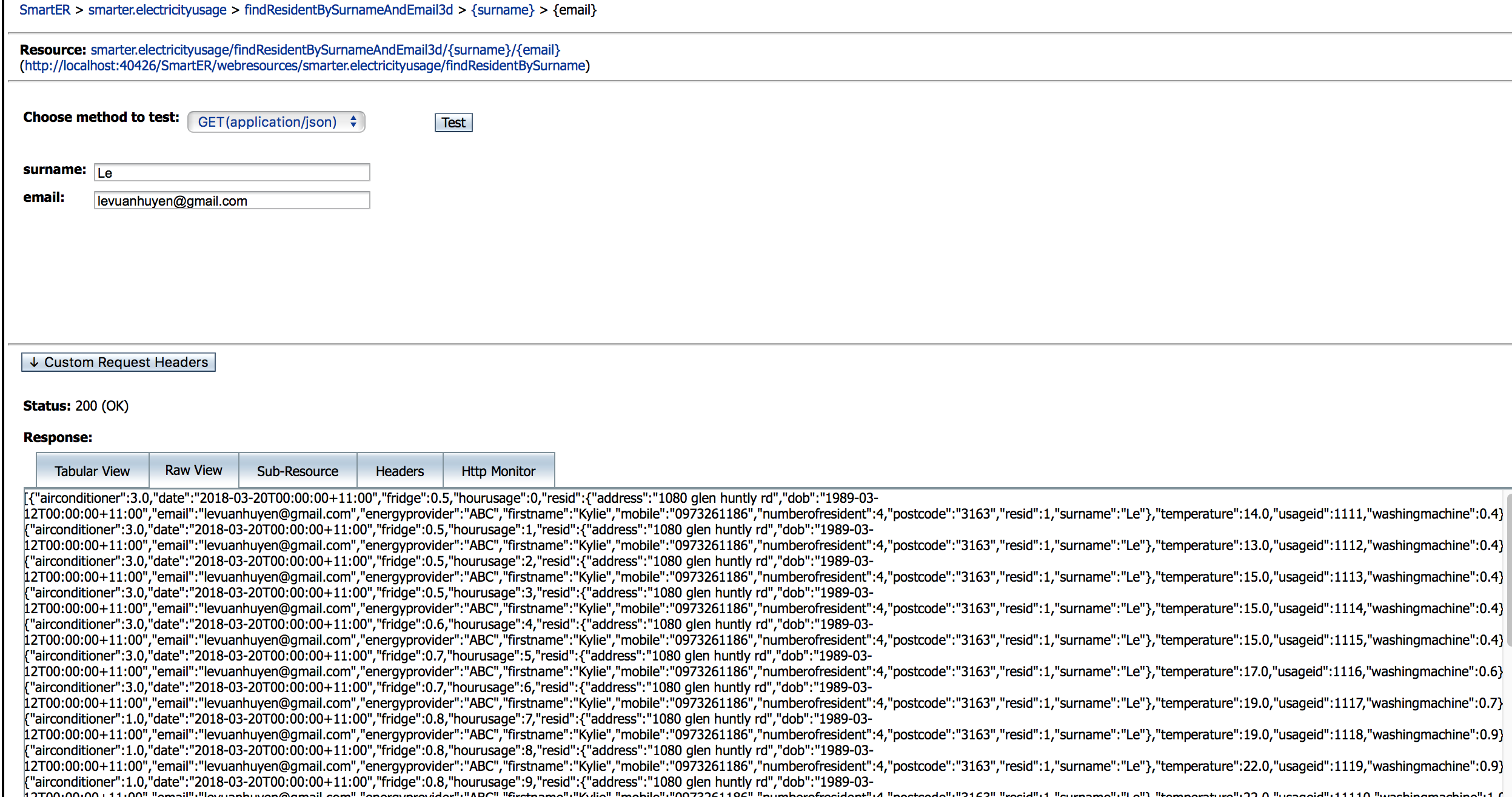


Figure 3.4: Static Query by Surname and Email

1. Task 4
   1. The hourly power usage (kWh) of the appliance

* Input: a resident id, an appliance name, a date, and an hour.
* Output: the hourly power usage (kWh) of the appliance specified for that date and time (and for that resident)

(Note: applianceName should be exactly match with applianceName: fridge, airconditioner, washingmachince)

Restful method:

|  |  |
| --- | --- |
|  |  |
| Restful Method on ElectricityusageFacadeRest.java | @GET  @Path("findHourUsage4a/{resid}/{applianceName}/{date}/{hourusage}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public Object findHourUsage4a(@PathParam("resid") Integer resid, @PathParam("applianceName") String applianceName, @PathParam("date") Date date, @PathParam("hourusage") Integer hourusage) {  TypedQuery<Object[]> query = em.createQuery("Select e.resid.resid, e.date, e.hourusage, e.airconditioner, e.fridge, e.washingmachine from Electricityusage e where e.resid.resid = :resid and e.date = :date and e.hourusage = :hourusage", Object[].class);  query.setParameter("resid", resid);  query.setParameter("date", date);  query.setParameter("hourusage", hourusage);  List<Object[]> queryList = query.getResultList();  //JsonArrayBuilder arrayUsage = Json.createArrayBuilder();  JsonObject findHourUsage = Json.createObjectBuilder().build();  double appianceUsage = 0;  for (Object[] row : queryList) {  if (applianceName.contains("airconditioner")) {  appianceUsage = (double) row[3];  }  if (applianceName.contains("fridge")) {  appianceUsage = (double) row[4];  }  if (applianceName.contains("washingmachine")) {  appianceUsage = (double) row[5];  }  JsonObject findUsageMachine = Json.createObjectBuilder()  .add("resid", (int) row[0])  .add("date", row[1].toString())  .add("hourusage", (int) row[2])  .add("applianceName", applianceName)  .add("hourlyPowerUsage", appianceUsage)  .build();  //arrayUsage.add(findUsageMachine);  findHourUsage = findUsageMachine;  }  return findHourUsage;  } |
| Return an object of The hourly power usage (kWh) of the appliance | {"resid":1,"date":"Tue Mar 20 00:00:00 AEDT 2018","hourusage":17,"applianceName":"fridge","hourlyPowerUsage":0.3} |

Screenshot on web service:

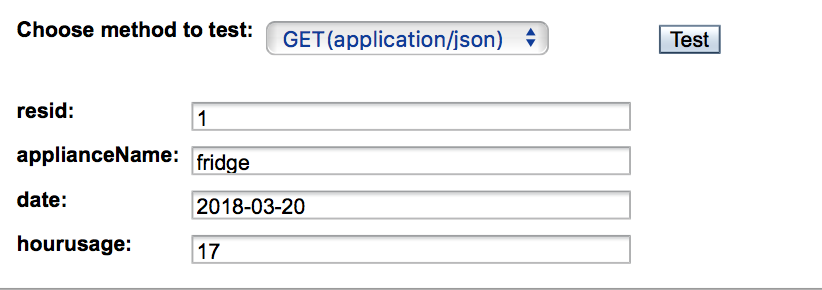


Figure 4.1.1: A screenshot of Input information

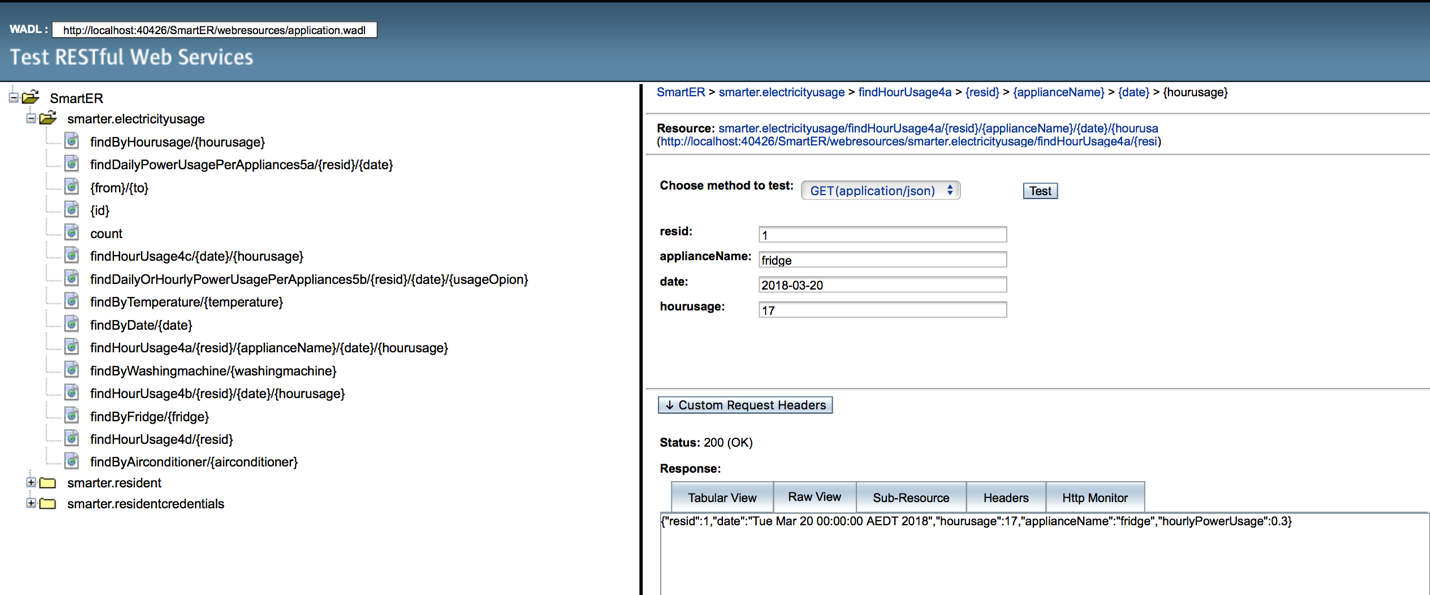


Figure 4.1.2: A record of the hourly power usage (kWh) of the appliance

* 1. The hourly power usage (kWh) of ALL THREE appliances
* Input: a resident id, a date, and an hour.
* Output: the hourly power usage (kWh) of ALL THREE appliances for that date and time (and for that resident).

Restful method:

|  |  |
| --- | --- |
|  |  |
| Restful Method on ElectricityusageFacadeRest.java | @GET  @Path("findHourUsage4b/{resid}/{date}/{hourusage}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public Object findHourUsage4b(@PathParam("resid") Integer resid, @PathParam("date") Date date, @PathParam("hourusage") Integer hourusage) {  TypedQuery<Object[]> query = em.createQuery("Select e.resid.resid, e.date, e.hourusage, e.airconditioner, e.fridge, e.washingmachine from Electricityusage e where e.resid.resid = :resid and e.date = :date and e.hourusage = :hourusage", Object[].class);  query.setParameter("resid", resid);  query.setParameter("date", date);  query.setParameter("hourusage", hourusage);  List<Object[]> queryList = query.getResultList();  //JsonArrayBuilder arrayUsage = Json.createArrayBuilder();  JsonObject findHourUsage = Json.createObjectBuilder().build();  for (Object[] row : queryList) {  JsonObject findUsageMachine = Json.createObjectBuilder()  .add("resid", (int) row[0])  .add("date", row[1].toString())  .add("hourusage", (int) row[2])  .add("airconditioner", (double) row[3])  .add("fridge", (double) row[4])  .add("washingmachine", (double) row[5])  .build();  //arrayUsage.add(findUsageMachine);  findHourUsage = findUsageMachine;  }  return findHourUsage;  } |
| Return an object of the hourly power usage (kWh) of ALL THREE appliances | {"resid":1,"date":"Tue Mar 20 00:00:00 AEDT 2018","hourusage":4,"airconditioner":3.0,"fridge":0.6,"washingmachine":0.4} |

Screenshot on web service:

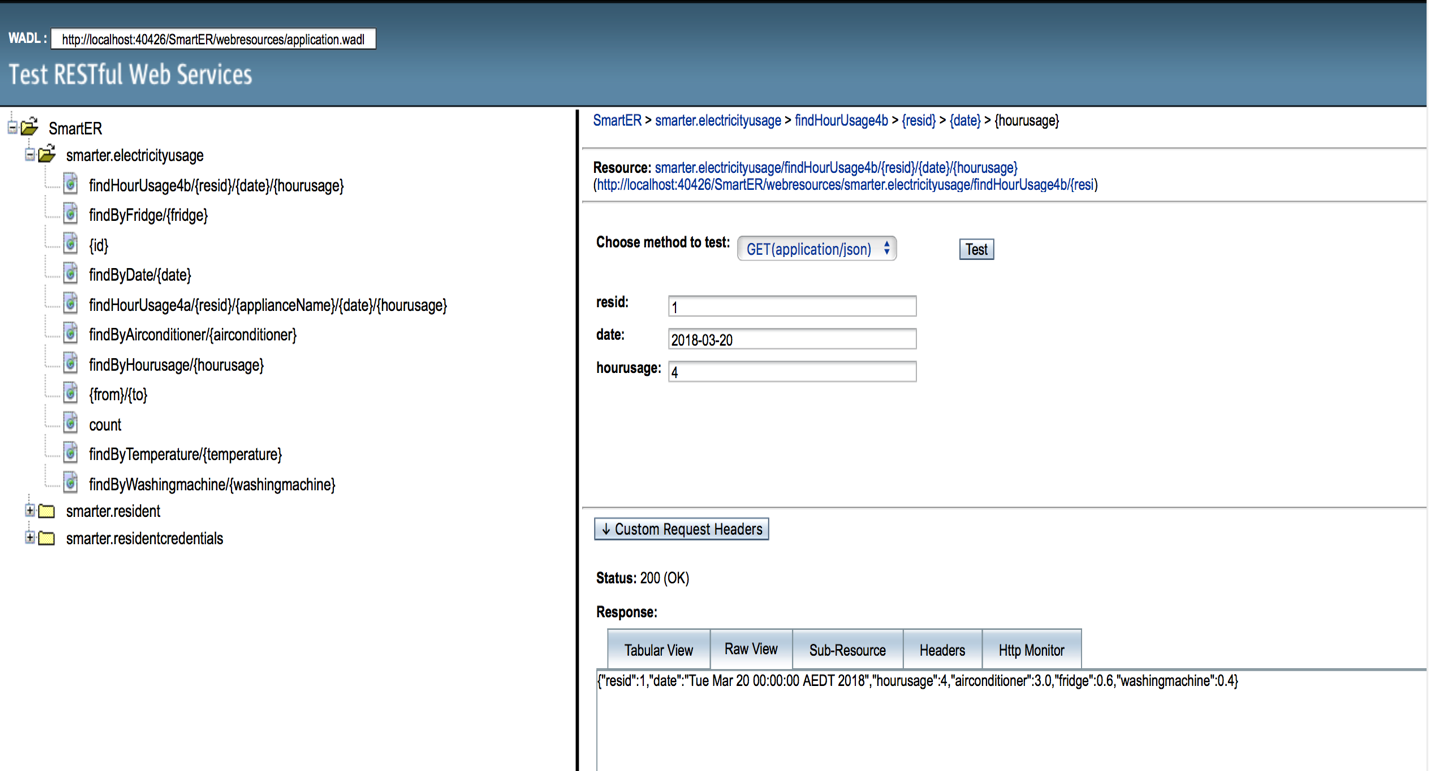


Figure 4.2: A record of the hourly power usage (kWh) of ALL THREE appliances

* 1. A list of hourly power usage (kWh) of ALL THREE appliances for all the residents
* Input: a date and an hour
* Output: A list of hourly power usage (kWh) of ALL THREE appliances for all the residents

Restful method:

|  |  |
| --- | --- |
|  |  |
| Restful Method on ElectricityusageFacadeRest.java | @GET  @Path("findHourUsage4c/{date}/{hourusage}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public Object findHourUsage4c(@PathParam("date") Date date, @PathParam("hourusage") Integer hourusage) {  TypedQuery<Object[]> query = em.createQuery("Select e.usageid,e.resid.resid,e.date,e.hourusage, e.airconditioner, e.fridge, e.washingmachine from Electricityusage e where e.date = :date and e.hourusage = :hourusage", Object[].class);  query.setParameter("date", date);  query.setParameter("hourusage", hourusage);  List<Object[]> queryList = query.getResultList();  JsonArrayBuilder arrayUsage = Json.createArrayBuilder();  for (Object[] row : queryList) {  Electricityusage electricityusage = find(row[0]);  String address = electricityusage.getResid().getAddress();  String postcode = electricityusage.getResid().getPostcode();  JsonObject findUsageMachine = Json.createObjectBuilder()  .add("resid", (int) row[1])  .add("postcode", postcode)  .add("address", address)  .add("totalPowerUsage", ((double) row[4] + (double) row[5] + (double) row[6]))  .build();  arrayUsage.add(findUsageMachine);  }  JsonArray jArray = arrayUsage.build();  return jArray;  } |
| Return A list of hourly power usage (kWh) of ALL THREE appliances for all the residents | [{"resid":1,"postcode":"3163","address":"1080 glen huntly rd","totalPowerUsage":3.0},  {"resid":2,"postcode":"3289","address":"219 nerrim rd","totalPowerUsage":5.1}] |

Screenshot on web service:

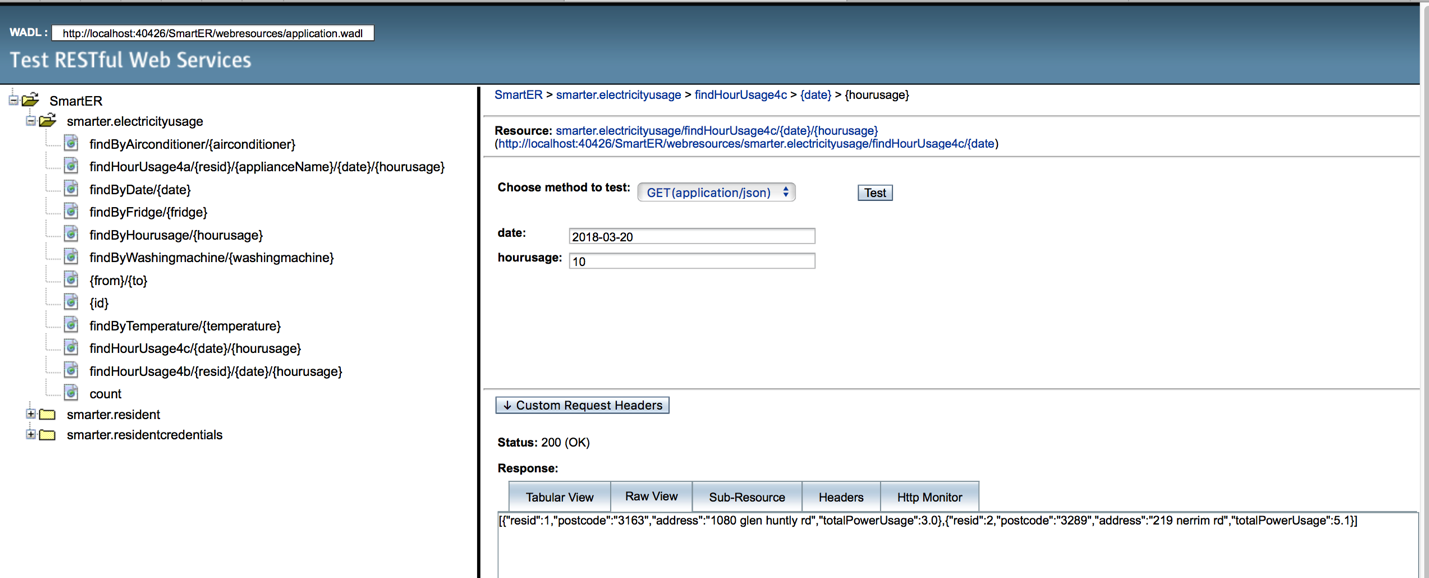


Figure 4.3: A list of hourly power usage (kWh) of ALL THREE appliances for all the residents

* 1. The highest hourly power consumption
* Input: a date and an hour
* Output: A list of hourly power usage (kWh) of ALL THREE appliances for all the residents

Restful method:

|  |  |
| --- | --- |
|  |  |
| Restful Method on ElectricityusageFacadeRest.java | @GET  @Path("findHourUsage4c/{date}/{hourusage}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public Object findHourUsage4c(@PathParam("date") Date date, @PathParam("hourusage") Integer hourusage) {  TypedQuery<Object[]> query = em.createQuery("Select e.usageid,e.resid.resid,e.date,e.hourusage, e.airconditioner, e.fridge, e.washingmachine from Electricityusage e where e.date = :date and e.hourusage = :hourusage", Object[].class);  query.setParameter("date", date);  query.setParameter("hourusage", hourusage);  List<Object[]> queryList = query.getResultList();  JsonArrayBuilder arrayUsage = Json.createArrayBuilder();  for (Object[] row : queryList) {  Electricityusage electricityusage = find(row[0]);  String address = electricityusage.getResid().getAddress();  String postcode = electricityusage.getResid().getPostcode();  JsonObject findUsageMachine = Json.createObjectBuilder()  .add("resid", (int) row[1])  .add("postcode", postcode)  .add("address", address)  .add("totalPowerUsage", ((double) row[4] + (double) row[5] + (double) row[6]))  .build();  arrayUsage.add(findUsageMachine);  }  JsonArray jArray = arrayUsage.build();  return jArray;  }  @GET  @Path("findHighestHourlyPowerConsump4d/{resid}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public highestTotalHourlyPowerConsumption findHighestHourlyPowerConsump4d(@PathParam("resid") Integer resid) {  TypedQuery<Electricityusage> query = em.createQuery("Select e from Electricityusage e where e.resid.resid = :resid", Electricityusage.class);  query.setParameter("resid", resid);    List<Electricityusage> queryList = query.getResultList();  highestTotalHourlyPowerConsumption highestPower = new highestTotalHourlyPowerConsumption();  Electricityusage finalUsage = new Electricityusage();    for(int i =0 ; i<queryList.size();i++)  {  Electricityusage tempUsage = CompareHouseConsumption(queryList.get(i), queryList.get(i+1));  if (i==0)  {  finalUsage = tempUsage;    }  }  double totalHourlyPowerConsumption = finalUsage.getAirconditioner() + finalUsage.getFridge() + finalUsage.getWashingmachine();  highestPower.setResid(resid);  highestPower.setHighestConsump(totalHourlyPowerConsumption);  highestPower.setDate(finalUsage.getDate().toString());  highestPower.setHourusage(finalUsage.getHourusage());      return highestPower;  }  //CompareHouseConsumption  public Electricityusage CompareHouseConsumption(Electricityusage u1, Electricityusage u2)  {  if ((u1.getAirconditioner() + u1.getFridge() + u1.getWashingmachine()) > (u2.getAirconditioner() + u2.getFridge() + u2.getWashingmachine()))  {  return u1;  }  else  {  return u2;  }  }  //5a: Response: {{"resid":1, "fridge":7.2,"aircon":0.0,"washingmachine":2.4}  @GET  @Path("findDailyPowerUsagePerAppliances5a/{resid}/{date}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public Object findDailyPowerUsagePerAppliances5a(@PathParam("resid") Integer resid, @PathParam("date") Date date) {  TypedQuery<Object[]> query = em.createQuery("Select e.resid.resid, e.date, e.airconditioner, e.fridge, e.washingmachine from Electricityusage e where e.resid.resid = :resid and e.date = :date", Object[].class);  query.setParameter("resid", resid);  query.setParameter("date", date);  List<Object[]> queryList = query.getResultList();  //JsonArrayBuilder arrayUsage = Json.createArrayBuilder();  JsonObject findHourUsage = Json.createObjectBuilder().build();  double finalAirConditionUsage = 0;  double finalFridge = 0;  double finalWashingMachine = 0;  for (Object[] row : queryList) {  double tempAirCondition = (double) row[2];  finalAirConditionUsage = tempAirCondition + finalAirConditionUsage;  double tempFridge = (double) row[3];  finalFridge = tempFridge + finalFridge;  double tempWasingMachine = (double) row[4];  finalWashingMachine = tempWasingMachine + finalWashingMachine;  JsonObject findUsageMachine = Json.createObjectBuilder()  .add("resid", (int) row[0])  .add("airconditioner", finalAirConditionUsage)  .add("fridge", finalFridge)  .add("washingmachine", finalWashingMachine)  .build();  //arrayUsage.add(findUsageMachine);  findHourUsage = findUsageMachine;  }  return findHourUsage;  } |
| A new Java Class | package SmartER;  import static SmartER.Electricityusage\_.date;  import static SmartER.Electricityusage\_.hourusage;  import static SmartER.Resident\_.resid;  import java.util.Date;  /\*\*  \*  \* @author levuanhuyen  \*/  public class highestTotalHourlyPowerConsumption {  // public void setResid(Integer resid) {  // throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  // }  private int resid;  private Double highestConsump;  private String date;  private int hourusage;      public highestTotalHourlyPowerConsumption() {  }      public highestTotalHourlyPowerConsumption(int resid,double highestConsump, String date, int hourusage ) {  }    public String getDate() {  return date;  }  public void setDate(String date) {  this.date = date;  }  public int getHourusage() {  return hourusage;  }  public void setHourusage(Integer hourusage) {  this.hourusage = hourusage;  }  public Double getHighestConsump() {  return highestConsump;  }  public void setHighestConsump(double highestConsump) {  this.highestConsump = highestConsump;  }  public int getResid() {  return resid;  }  public void setResid(int resid) {  this.resid = resid;  }  public void setResid(Integer resid) {  throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  }  } |

Screenshot on web service

1. Task 5
   1. Daily Usage of Appliances

* Input: resident id, and a date.
* Output: one record (JSON object) that includes the total daily power usage for each appliance for that date. Example: (the sum of hourly usage (0- 23) of each appliance for the date entered)

Restful method:

|  |  |
| --- | --- |
|  |  |
| Restful Method on ElectricityusageFacadeRest.java | @GET  @Path("findDailyPowerUsagePerAppliances5a/{resid}/{date}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public Object findDailyPowerUsagePerAppliances5a(@PathParam("resid") Integer resid, @PathParam("date") Date date) {  TypedQuery<Object[]> query = em.createQuery("Select e.resid.resid, e.date, e.airconditioner, e.fridge, e.washingmachine from Electricityusage e where e.resid.resid = :resid and e.date = :date", Object[].class);  query.setParameter("resid", resid);  query.setParameter("date", date);  List<Object[]> queryList = query.getResultList();  JsonObject findHourUsage = Json.createObjectBuilder().build();  double finalAirConditionUsage = 0;  double finalFridge = 0;  double finalWashingMachine = 0;  for (Object[] row : queryList) {  double tempAirCondition = (double) row[2];  finalAirConditionUsage = tempAirCondition + finalAirConditionUsage;  double tempFridge = (double) row[3];  finalFridge = tempFridge + finalFridge;  double tempWasingMachine = (double) row[4];  finalWashingMachine = tempWasingMachine + finalWashingMachine;  JsonObject findUsageMachine = Json.createObjectBuilder()  .add("resid", (int) row[0])  .add("airconditioner", finalAirConditionUsage)  .add("fridge", finalFridge)  .add("washingmachine", finalWashingMachine)  .build();  findHourUsage = findUsageMachine;  }  return findHourUsage;  } |
| Return a record | {"resid":1,"airconditioner":60.0,"fridge":14.3,"washingmachine":22.600000000000005} |

Screenshot on web service:

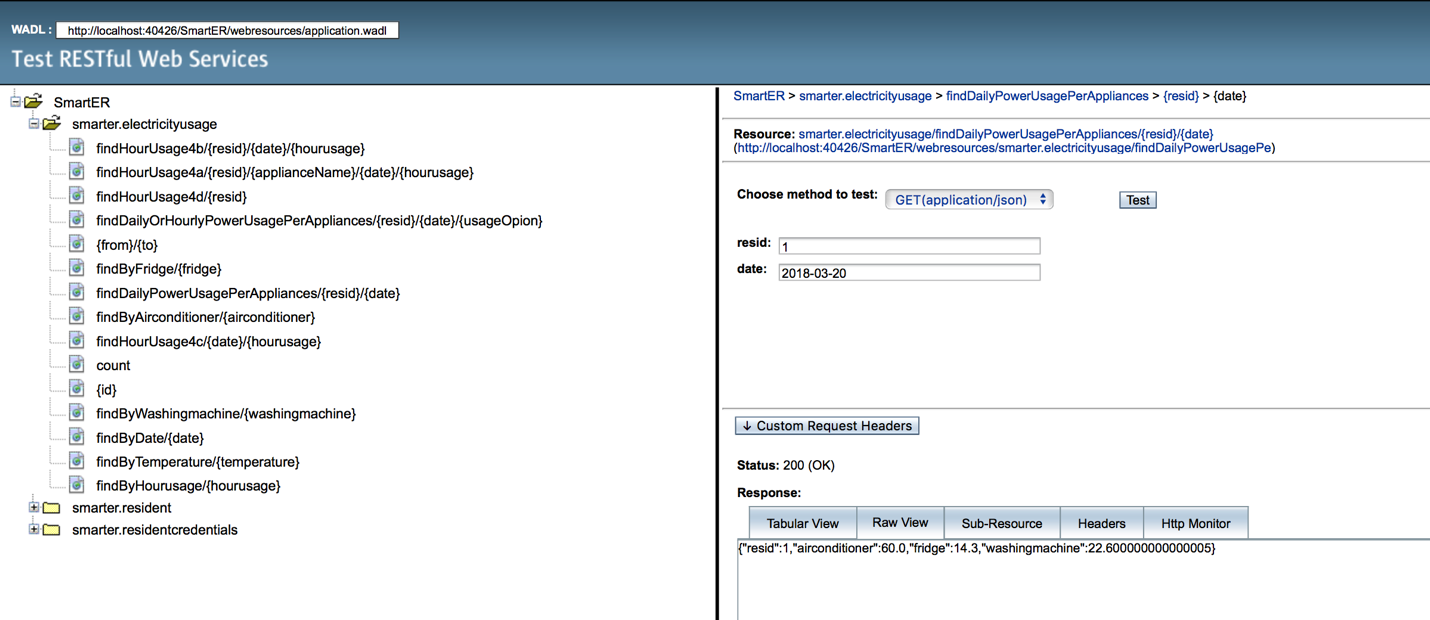


Figure 5.1: A record of the total daily power usage for each appliance for that date

## Hourly/Daily Usage

* Input: a resident id, a date and a view variable (‘hourly or daily’).
* Output:
  + If the ‘hourly’ value is entered, the list will return the total hourly usage data and the temperature value per hour for that particular date (the list will include 24 records).
  + If the ‘daily’ value is entered, return a record of the total usage of 24 hours, and an average temperature value (an average of 24 temperature values) for that date

Restful Method:

|  |  |
| --- | --- |
| Restful Method on ElectricityusageFacadeRest.java | @GET  @Path("findDailyOrHourlyPowerUsagePerAppliances5b/{resid}/{date}/{usageOpion}")  @Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})  public Object findDailyOrHourlyPowerUsagePerAppliances5b(@PathParam("resid") Integer resid, @PathParam("date") Date date, @PathParam("usageOpion") String usageOption) {  TypedQuery<Object[]> query = em.createQuery("Select e.resid.resid, e.date, e.temperature, e.airconditioner, e.fridge, e.washingmachine, e.hourusage from Electricityusage e where e.resid.resid = :resid and e.date = :date", Object[].class);  query.setParameter("resid", resid);  query.setParameter("date", date);  List<Object[]> queryList = query.getResultList();  JsonArrayBuilder arrayUsage = Json.createArrayBuilder();  JsonObject findHourUsage = Json.createObjectBuilder().build();  double finalDailyUsage = 0;  double finalDailyTemp = 0;  for (Object[] row : queryList) {  if (usageOption.contains("hourly")) {  JsonObject listHourlyUsage = Json.createObjectBuilder()  .add("resid", (int) row[0])  .add("totalUsage", (double) row[3] + (double) row[4] + (double) row[5])  .add("temperature", (double) row[2])  .add("date", row[1].toString())  .add("hour", (int) row[6])  .build();  arrayUsage.add(listHourlyUsage);  }  if (usageOption.contains("daily")) {  double tempDailyUsage = (double) row[3] + (double) row[4] + (double) row[5];  finalDailyUsage = tempDailyUsage + finalDailyUsage;  double tempDailyTemperature = (double) row[2];  finalDailyTemp = tempDailyTemperature + finalDailyTemp;  JsonObject totalDailyUsage = Json.createObjectBuilder()  .add("resid", (int) row[0])  .add("usage", (double) finalDailyUsage)  .add("temperature", (double) (finalDailyTemp / 24))  .build();  findHourUsage = totalDailyUsage;  }  }  if (usageOption.contains("daily")) {  return findHourUsage;  }  JsonArray arrayBuilder = arrayUsage.build();  return arrayBuilder;  } |
| Variable View: Hourly | [{"resid":1,"totalUsage":3.9,"temperature":14.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":0},  {"resid":1,"totalUsage":3.9,"temperature":13.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":1},  {"resid":1,"totalUsage":3.9,"temperature":15.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":2},  {"resid":1,"totalUsage":3.9,"temperature":15.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":3},  {"resid":1,"totalUsage":4.0,"temperature":15.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":4},  {"resid":1,"totalUsage":4.3,"temperature":17.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":5},  ….  {"resid":1,"totalUsage":4.0,"temperature":19.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":20},  {"resid":1,"totalUsage":3.0,"temperature":17.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":21},  {"resid":1,"totalUsage":2.7,"temperature":16.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":22},  {"resid":1,"totalUsage":2.6,"temperature":16.0,"date":"Tue Mar 20 00:00:00 AEDT 2018","hour":23}] |
| Variable View: Daily | {"resid":1,"usage":96.9,"temperature":19.375} |

Screenshot on web service:

1. Option A: hourly

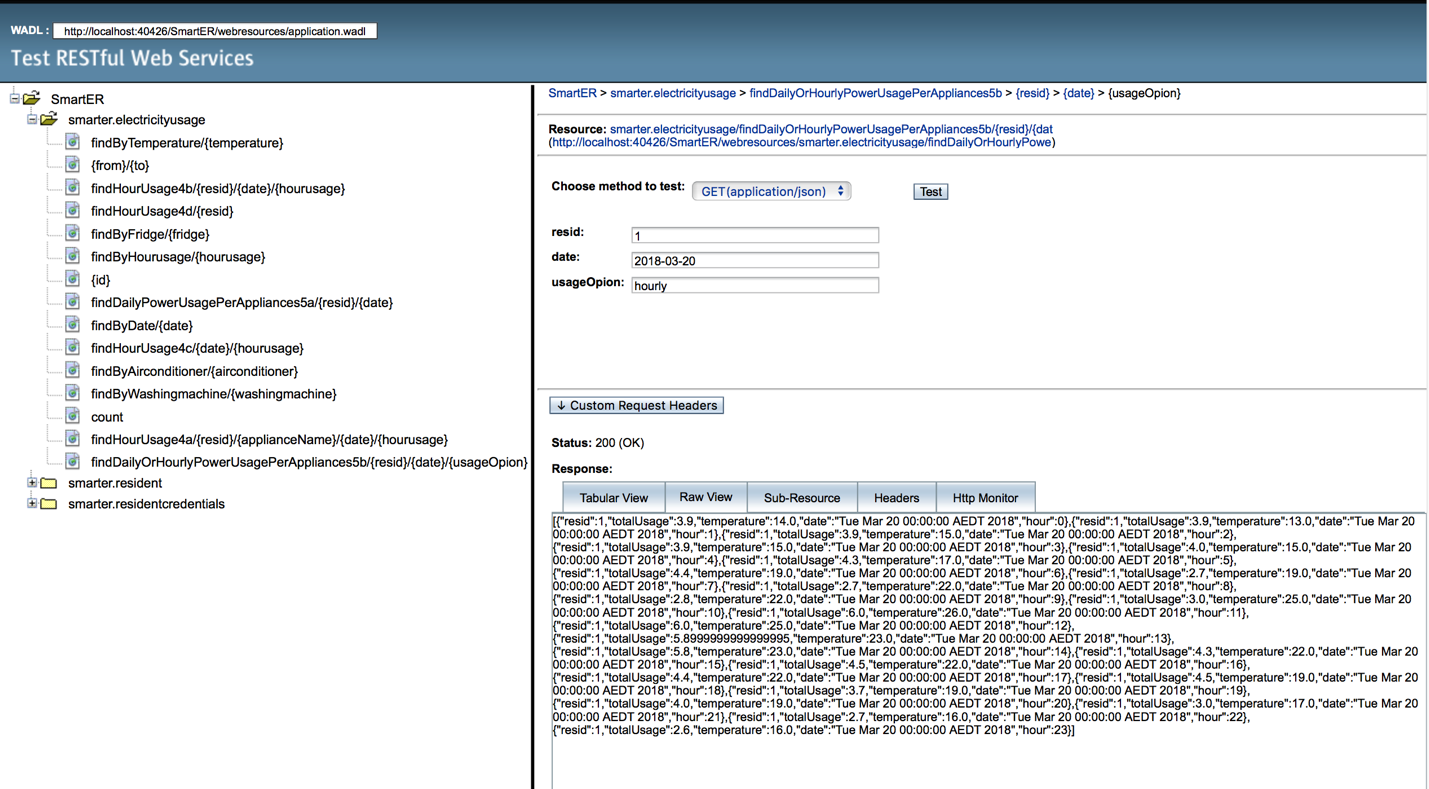


Figure 5.2.1: A list of 24 records showing the total hourly usage data and the temperature value per hour for that particular date

1. Option B: daily

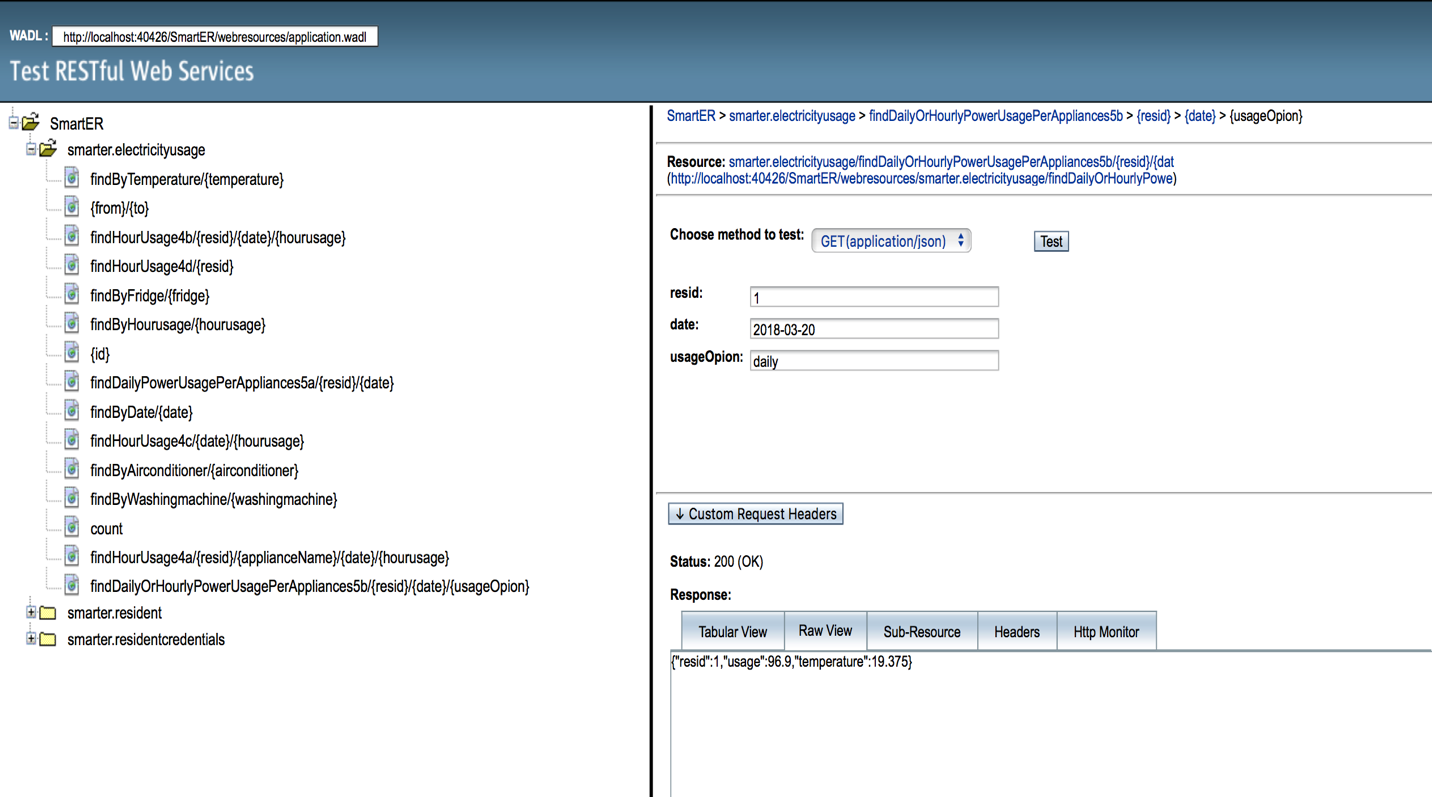


Figure 5.2.2: A record of the total usage of 24 hours, and an average temperature value (an average of 24 temperature values) for that date