Introduction

EHRServer is a generic, minimal, open source, openEHR clinical data storage with a REST API for committing and querying openEHR data. EHRServer was designed and developed by Pablo Pazos[1] at CaboLabs Healthcare Informatics. It also has a Web Based UI to manage EHRs and for visual query creation using openEHR Archetypes and Operational Templates.

EHRServer on GitHub: https://github.com/ppazos/cabolabs-ehrserver

Test instance in the cloud: https://cabolabs-ehrserver.rhcloud.com/ehr-0.1/

Installing EHRServer locally

You can download EHRServer from here: https://github.com/ppazos/cabolabs-ehrserver/archive/master.zip

And run it locally just by installing Grails v2.4.4: http://www.grails.org/download.html

[1] https://www.linkedin.com/in/pablopazosgutierrez

Using EHRServer

Before committing any data, you need an Operational Template (OPT) that specifies the structure, semantics, constraints and terminology of your clinical documents. You can find OPT samples in our GitHub repo[2].

There are two ways of adding an OPT into EHRServer:

1. Copy the OPT file into the "ehrserver/opts" folder. When EHRServer is started, it will automatically load all the OPTs from that folder.
2. Upload the OPT file using the Administration User Interface[3]. Just go to Templates > Upload Templates.

OPTs should comply with this XSD to be accepted by EHRServer:

https://github.com/ppazos/cabolabs-ehrserver/blob/master/xsd/OperationalTemplate.xsd

[2] https://github.com/ppazos/cabolabs-ehrserver/tree/master/opts
[3] http://cabolabs-ehrserver.rhcloud.com/ehr-0.1/

Commit Service

Before committing any data, you have to create a patient and his/her EHR. All committed clinical documents should be associated with an EHR.

That can be done using the Administration User Interface:

People > New Person (create the person with the patient role), then Create EHR for the new Person.

To commit data to EHRServer, use the commit REST service:

```
_baseurl_/rest/commit(String ehrId, Array xmlVersions, String auditSystemId, String auditCommitter)
```

Where:

- _baseurl_ is where EHRServer is running, e.g. http://localhost:8090/ehr/rest/commit
- ehrId is the UID of the EHR where the clinical documents are committed to
- xmlVersions is a set of clinical documents (called COMPOSITIONS in openEHR) in XML format, that are contained in a VERSION object that contains version control and audit information.
- auditSystemId is the UID of the API client
- auditCommitter is the name of the person responsible of the commit (we will add an id parameter to identify that person)

The XML VERSIONs should comply with this XSD:
https://github.com/ppazos/cabolabs-ehrserver/blob/master/xsd/Version.xsd

Sample VERSIONs in XML can be found here
https://github.com/ppazos/cabolabs-emrapp/tree/master/committed

To test the commit service without writing your own client, you can use one of these apps:

- https://github.com/ppazos/cabolabs-emrapp
- https://github.com/ppazos/EHRCommitter

Also check the unit test code:

- https://github.com/ppazos/cabolabs-ehrserver/blob/master/test/unit/ehr/RestControllerTests.groovy

Querying

From the Administration User Interface you can create any number of queries. EHRServer supports two types of queries: to retrieve clinical documents (openEHR COMPOSITIONs) using criteria over data (e.g. retrieve all the documents with systolic BP > 140), or to retrieve data values (e.g. systolic BP, diastolic BP and body temperature). Each query is executed in the context of one EHR. In the future, we'll remove this constraint to run queries over sets of EHRs (e.g. to do statistical analysis, or to query family EHRs).

Show Query

UID	1d00aef6-1609-4567-9ca9-1af457a910d0
Name	dsgs
Group	path
Format	xml
Type	datavalue

Select

archetypeId	path
openEHR-EHR-COMPOSITION.signos.v1	/content[at0006]/data[at0007]/events[at0002]/data[at0003]/items[at0004]/value
openEHR-EHR-COMPOSITION.signos.v1	/content[at0006]/data[at0007]/events[at0002]/data[at0003]/items[at0005]/value
openEHR-EHR-COMPOSITION.signos.v1	/content[at0008]/data[at0009]/events[at0010]/data[at0011]/items[at0012]/value

Fig. 1: Query for data, grouped by path, format as XML by default.

The examples below are shown after committing two records to the EHRServer, for the same EHR.

You can also test your query using the Administration User Interface. If your query retrieves data values, and you select JSON as an output format, numeric data will be charted for you.

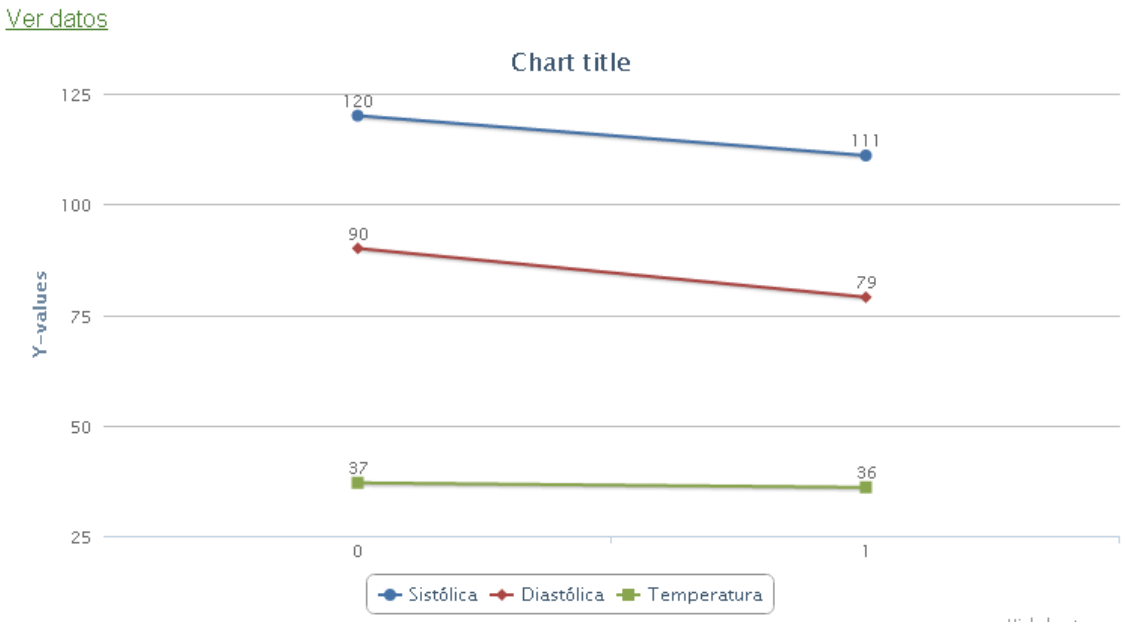

Fig. 2: Automatic data charting for JSON data results

And you can display the returned data, JSON in this case:


```
{
  "/content[at0006]/data[at0007]/events[at0002]/data[at0003]/items[at0004]/value": {
    "type": "DvQuantity",
    "name": "Sistólica",
    "serie": [
      {
        "magnitude": 120,
        "units": "mm[Hg]",
        "date": "2013-09-07T20:16:09Z"
      },
      {
        "magnitude": 111,
        "units": "mm[Hg]",
        "date": "2013-09-07T20:29:31Z"
      }
    ]
  },
  "/content[at0006]/data[at0007]/events[at0002]/data[at0003]/items[at0005]/value": {
    "type": "DvQuantity",
    "name": "Diastólica",
    "serie": [
      {
        "magnitude": 90,
        "units": "mm[Hg]",
        "date": "2013-09-07T20:16:09Z"
      },
      {
        "magnitude": 79,
        "units": "mm[Hg]",
        "date": "2013-09-07T20:29:31Z"
      }
    ]
```


Fig. 3: data retrieved for query

Executing queries using the REST services:

```
        _baseurl_/rest/query(String queryUid, String ehrId, String format,
                             boolean retrieveData, boolean showUI, String group)
```

This is a real query execution for the previously defined query, with two COMPOSITIONS committed for the same EHR:

http://cabolabs-ehrserver.rhcloud.com/ehr-0.1/rest/query?queryUid=c265110b-df2e-4b65-b20f-094d92cb5b4f&ehrId=11111111-1111-1111-1111-111111111111

Result:

```
<map>
  <entry key="openEHR-EHR-
OBSERVATION.blood_pressure.v1/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value">
    <entry key="type">DV_QUANTITY</entry>
    <entry key="name">Sistólica</entry>
    <entry key="serie">
      <map>
        <entry key="magnitude">36346.0</entry>
        <entry key="units">mm[Hg]</entry>
        <entry key="date">2015-05-22 01:48:31.0 EDT</entry>
      </map>
      <map>
        <entry key="magnitude">36346.0</entry>
        <entry key="units">mm[Hg]</entry>
        <entry key="date">2015-05-22 01:52:28.0 EDT</entry>
      </map>
      <map>
        <entry key="magnitude">120.0</entry>
        <entry key="units">mm[Hg]</entry>
        <entry key="date">2015-05-22 01:54:59.0 EDT</entry>
      </map>
    </entry>
  </entry>
  <entry key="openEHR-EHR-
OBSERVATION.blood_pressure.v1/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value">
    <entry key="type">DV_QUANTITY</entry>
    <entry key="name">Diastólica</entry>
    <entry key="serie">
      <map>
        <entry key="magnitude">34534.0</entry>
        <entry key="units">mm[Hg]</entry>
        <entry key="date">2015-05-22 01:48:31.0 EDT</entry>
      </map>
      <map>
        <entry key="magnitude">34534.0</entry>
        <entry key="units">mm[Hg]</entry>
        <entry key="date">2015-05-22 01:52:28.0 EDT</entry>
      </map>
      <map>
        <entry key="magnitude">90.0</entry>
        <entry key="units">mm[Hg]</entry>
        <entry key="date">2015-05-22 01:54:59.0 EDT</entry>
      </map>
    </entry>
  </entry>
  <entry key="openEHR-EHR-
OBSERVATION.body_weight.v1/data[at0002]/events[at0003]/data[at0001]/items[at0004]/value">
    <entry key="type">DV_QUANTITY</entry>
```

```
    <entry key="name">Peso</entry>
    <entry key="serie"/>
  </entry>
  <entry key="openEHR-EHR-
OBSERVATION.pulse.v1/data[at0002]/events[at0003]/data[at0001]/items[at0004]/value">
    <entry key="type">DV_QUANTITY</entry>
    <entry key="name">Frecuencia</entry>
    <entry key="serie"/>
  </entry>
</map>
```

Now specifying format=json

http://cabolabs-ehrserver.rhcloud.com/ehr-0.1/rest/query?queryUid=c265110b-df2e-4b65-b20f-094d92cb5b4f&ehrId=11111111-1111-1111-1111-111111111111&format=json

Result:

```
{
    "openEHR-EHR-
OBSERVATION.blood_pressure.v1/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value": {
        "type": "DV_QUANTITY",
        "name": "Sistólica",
        "serie": [
            {
                "magnitude": 36346,
                "units": "mm[Hg]",
                "date": "2015-05-22T05:48:31Z"
            },
            {
                "magnitude": 36346,
                "units": "mm[Hg]",
                "date": "2015-05-22T05:52:28Z"
            },
            {
                "magnitude": 120,
                "units": "mm[Hg]",
                "date": "2015-05-22T05:54:59Z"
            }
        ]
    },
    "openEHR-EHR-
OBSERVATION.blood_pressure.v1/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value": {
        "type": "DV_QUANTITY",
        "name": "Diastólica",
        "serie": [
            {
                "magnitude": 34534,
                "units": "mm[Hg]",
                "date": "2015-05-22T05:48:31Z"
            },
            {
                "magnitude": 34534,
                "units": "mm[Hg]",
                "date": "2015-05-22T05:52:28Z"
            },
            {
                "magnitude": 90,
                "units": "mm[Hg]",
                "date": "2015-05-22T05:54:59Z"
            }
        ]
    },
    "openEHR-EHR-
OBSERVATION.body_weight.v1/data[at0002]/events[at0003]/data[at0001]/items[at0004]/value": {
```

```
        "type": "DV_QUANTITY",
        "name": "Peso",
        "serie": []
    },
    "openEHR-EHR-OBSERVATION.pulse.v1/data[at0002]/events[at0003]/data[at0001]/items[at0004]/value":
{
        "type": "DV_QUANTITY",
        "name": "Frecuencia",
        "serie": []
    }
}
```

Query Result Grouping

For queries that retrieve data values, there are 3 ways of grouping the results:

- **Ungrouped**: just a set of data values without any grouping.
- **Composition**: group data values that belongs to the same composition. We call this "row grouper".
- **Path**: group data values with the same path. We call this "column grouped".

Uses

Grouping data value results by composition simplifies displaying data as tables, while using the path grouper simplifies charting because it groups data with the same semantics that is easily mapped to time series, because results also includes a timestamp.

The mix of REST querying services and grouping, make the creation of clinical user interfaces, like clinical dashboards, clinical history and clinical summaries, a very simple task.