ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP28

DEVELOPMENT OF SOFTWARE FOR ACCESSING THE VEPP-2000 COLLIDER FACILITY ARCHIVING SYSTEM

O. S. Shubina†, A.I.Senchenko, P.Yu. Shatunov
BINP SB RAS and Novosibirsk State University, Novosibirsk, Russia

Abstract

The VEPP-2000 is an electron-positron collider, that was commissioned at Budker Institute of Nuclear Physics. The VEPP-2000 acceleration complex consists of a few main subsystems: BEP booster ring and VEPP-2000 collider ring. Data from accelerator complex are recorded regularly with a frequency at 1 Hz. There is often a need to obtain already stored data for analysis or modeling. In addition, you must provide remote data access to optimize the workflow. The solution of this problem must be universal, and it must be easily adapted to various databases and installation modes. To solve the task, the software was developed based on the client-server architecture. The server part is responsible for processing data in automatic mode according to the developed algorithm. The client part allows to view the data in a user-friendly form. This article talks about the development of software, simplifying access to the VEPP-2000 archiving system, that is launched on the VEPP-2000 control panel.

INTRODUCTION

The VEPP 2000 is an electron-positron collider located at the Budker Institute of Nuclear Physics. The VEPP-2000 acceleration complex (Fig. 1) consists of two main subsystems: the BEP booster ring and the VEPP-2000 collider ring. Beam diagnostics system is based on 16 optical CCD cameras that register the synchrotron light from both ends of the bending magnets and provide full information about beam positions, intensities and profiles. In addition to optical BPMs [1], there are also 4 pick-up stations in the technical straight sections and the one current transformer working as an absolute current meter.

Over than 1200 control channels and 2400 monitoring channels and their joint usage impose rigid restriction on the control system [2]. The architecture of VEPP-2000 software is based on traditional three-layer structure. The important application in the middleware layer is an elaborately designed Log Server [3]. Its purpose is to store all necessary automation system data to the storage.

Every day various accelerator-based experiments are carried out. Sometimes researchers need to obtain the past data for the analysis and calculations. Moreover, it will be useful to have remote access to this data.

Thus, there is the problem of access to the data stored on the hard disk. Also, the solution of this problem have to be universal, and easily adapted to various databases and installation modes.

THE GENERAL SCHEME OF THE APPLICATION

To solve the task, the software has been developed based on the client-server architecture (Fig. 2). HTTP has been chosen as a transport layer protocol, since it is the most common method of data transmission. And also, the REST approach has formed the basis of this application.

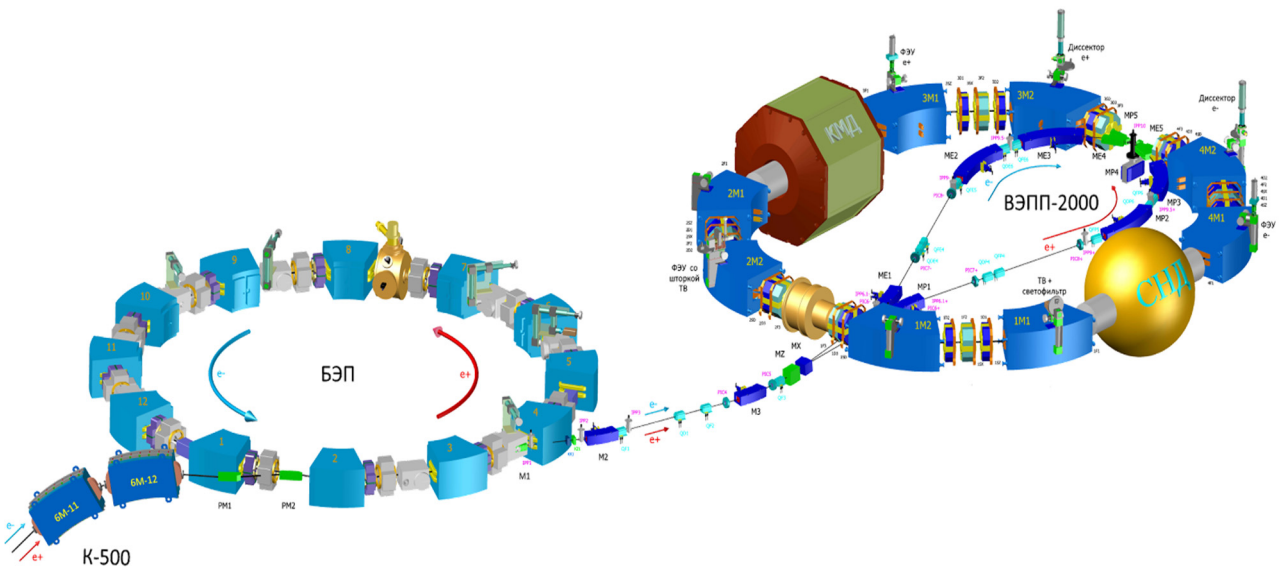

Figure 1: VEPP-2000.

† olgashubina2011@gmail.com.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP28

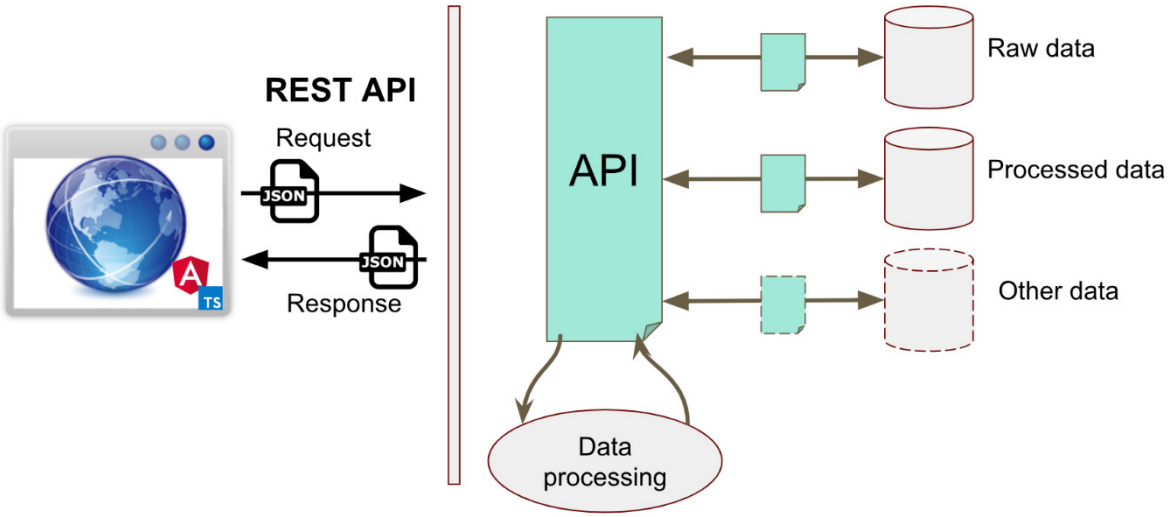

Figure 2: General scheme.

The interaction between the client and the server is carried out using standard HTTP methods. When the user enters all necessary data, the client part of the application generates a special request and passes it as a JSON format message, which has the following structure:

- t1 - the beginning of the requested time interval;
- t2 - the end of the requested time interval;
- channels - the list of channels;
- fields - the method of data processing;
- level - the level of compression.

Next, the server processes the request and sends the response also in JSON format. Then the client works on the received data and provides it to the user. Below we will consider in more detail how the client and server parts of the software are arranged.

THE SERVER PART OF THE APPLICATION

To create the server part, the Flask web framework was used in the Python programming language using the Jinja2 template engine. This framework was chosen because it is designed to create small web applications that have basic capabilities, which are sufficient for our system.

The server part (Fig. 3) provides a single-entry point for obtaining the necessary data via the URL identifier. Its main task is to handle requests coming from the client and to issue data that satisfy the query parameters. Rest approach allows to hide from the client the implementation of specific procedures describing the structure of data storage, as well as procedures for interacting with it.

The server has several main components. The first component is responsible for provision of data from the database. Since it was necessary to create the most universal tool for interacting with data, it was decided to implement a unified software interface for accessing data from various sources. It allows to operate data through small interfaces that are implemented for a specific storage source.

The VEPP-2000 archiving system stores all the necessary data in the object-relational database of PostgreSQL. For example, for a month, the amount of data stored is about 900 million records, which is equivalent to 62 GB of disk usage. Thus, there is the problem of convenient representation, a large amount of data for quick viewing. Therefore, the component responsible for processing the data according to the developed algorithms was created. Since, amount of raw data is growing, there is a need to average new data regularly. To solve this problem, the Celery [4] task manager was used.

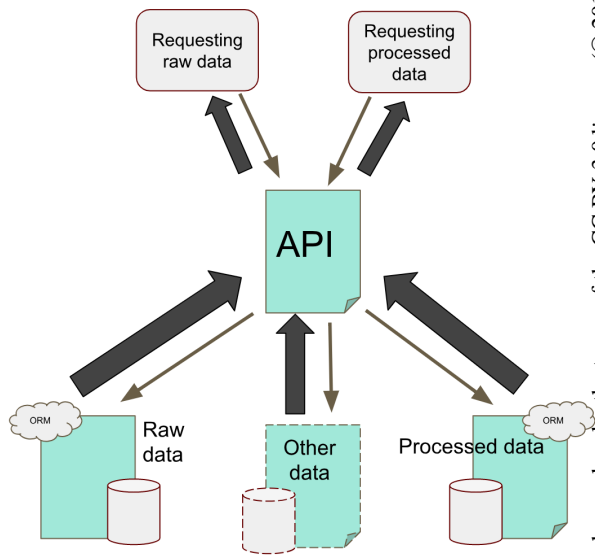

Figure 3: Server part scheme.

In the VEPP-2000 control system, currently, there are two types of channels: the first accepts text values, the second accepts a floating-point number. Depending on the type of data received, two data processing scenarios were developed. For numerical values, this is a finding for a certain period of time:

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP28

- arithmetic mean;
- the minimum value;
- the maximum value;
- the median value;
- root mean square deviation.

And for text:

- the left boundary value;
- right boundary value;
- the most significant value;
- central importance.

The last method processes data with both textual and numerical values. Data is processed by the methods described above at three different intervals. Thus, we obtain a group of tables consisting of two processing methods for three levels of compression.

THE CLIENT PART OF THE APPLICATION

The next stage of our work was the creation of the client part. The graphical interface (Fig. 4) was developed on the basis of the Angular5 [5] framework using the TypeScript language.

The main tasks of the GUI are the providing a convenient viewing of channels with various details, and the providing an interface to access data. For the convenience, the channel list was organized as a drop-down tree, similar to a hierarchical file system. For example, the path to the VEPP / Currents / FZ channel will look like this:

- VEPP
  - Currents
    - VEPP / Currents / FZ

Since only one selected value is drawn when displaying the processed data graph, for viewing convenience, when hovering to a specific point, the system displays the remaining values calculated for the given time.

Also, for a more comfortable viewing of the resulting graphs it is possible to scale the graphics and switch between different levels of compression. And also, it provides a convenient navigation on the graphic of mouse movement.

Since data from channels that have textual values cannot be represented in a graph similar to a graph with numerical values, it was customary to visualize information with text values using a table. This method is a kind of graphical interpretation of the tables stored in the database. Also, for more convenience, in the form of a table, it's possible to display data with numerical values.

Despite the fact that the graphical interface was faced with the task of providing fast and comfortable viewing of data, sometimes there might be a need to get raw data for a long period. In this case, the application provides a system for generating a script in Python. It has the same interface as for receiving data, only thus, the user is given a script displayed, which, also, is available for download. This script, when it is launched, connects to the server and uploads the data to a file located on the computer from which it is running.

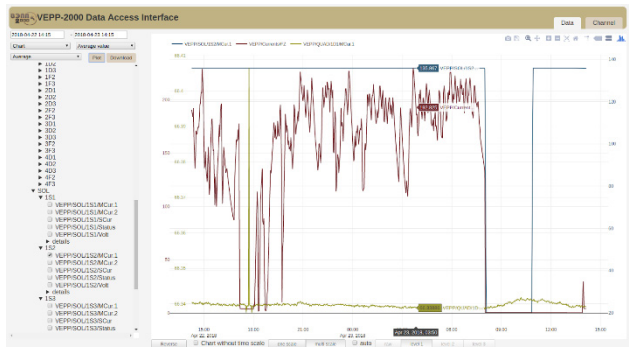

Figure 4: Web GUI.

CONCLUSION

As a result of this work, the software, that provides remote access to the data of the archiving system of the VEPP-2000 accelerator complex, has been developed. Using the REST approach provided a single point of access to data from various sources. As a result, a sufficiently flexible system was obtained in which the client's implementation does not depend on the implementation of the server, which allows, if it is necessary, to make minor or even significant changes to one of the parts without affecting the other at all.

For the convenience of viewing large amounts of data, the algorithms for processing them were implemented. According to the conducted tests, these algorithms allowed to increase the speed of data acquisition on about 90 times. Since the raw data will be regularly replenished, a task manager has been created that performs automatic data processing, according to the developed algorithms. Also, the approach, that allows to avoid the problems of reconnection, has been implemented.

REFERENCES

[1] Yu. Shatunov *et al.*, "Project of a New ElectronPositron Collider VEPP-2000", in *Proc. EPAC'00*, Vienna, Austria, p. 439.

[2] A. Senchenko *et al.*, "VEPP2000 Collider Control System", in *Proc. PCaPAC'12*, Kolkata, India, Dec. 2012, paper FRCB04.

[3] A. Senchenko, D. Berkaev, "VEPP2000 Logging System", in *Proc. PCaPAC'12*, Kolkata, India, Dec. 2012, paper WEPD14.

[4] Celery official site, `http://www.celeryproject.org`

[5] Angular5 official site, `https://angular.io`