Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Integration of standard datasources with interactive data visualization solutions

Bachelor's Thesis

Author
Márton Orova

Advisor
dr. Zoltán Szatmári
Attila Simon

November 22, 2019

Contents

Kivonat **i**

Abstract **ii**

1 Introduction **1**

1.1 Problem definition 1

1.2 Motivation 2

1.3 Goals 2

2 Background **3**

2.1 Datasource 3

2.2 Grafana 3

2.2.1 Data Source 3

2.2.2 Panel 4

2.2.3 Query Editor 4

2.2.4 Dashboard 4

2.2.5 Interactivity in general 4

2.2.6 Interactivity in Grafana 5

2.2.6.1 Time Range Controls 5

2.2.6.2 Variables and Templating 5

2.2.6.3 Basic statistic values in legend 6

2.2.6.4 Data links 6

2.2.6.5 Creating custom interactivity 6

2.3 Relevant RapidMiner components 7

2.4 JSON 7

2.5 REST API 7

3 Design **8**

3.1 Architecture 8

3.2 Components 8

3.2.1 Extended SimpleJSON datasource plugin . . . 9
3.2.2 Proxy . . . 9
3.2.3 RapidMiner components . . . 9
3.2.4 JSON backend . . . 9
3.2.5 Python data source . . . 9
3.2.6 Weather API . . . 10
3.2.7 MySQL database . . . 10

4 Implementation **11**
4.1 Gateway . . . 11
4.2 Pros . . . 11
4.3 Cons . . . 11

5 Evaluation **12**

6 Future work **13**

7 Related works **14**

8 Summary **15**

Acknowledgements **16**

Bibliography **17**

HALLGATÓI NYILATKOZAT

Alulírott *Orova Márton*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2019. november 22.

Orova Márton
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végző hallgatók által elkészítendő szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon LaTeX alapú, a *TeXLive* TeX-implementációval és a PDF-LaTeX fordítóval működőképes.

Abstract

This document is a LaTeX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* TeX implementation, and it requires the PDF-LaTeX compiler.

Chapter 1

Introduction

A bevezető tartalmazza a diplomaterv-kiírás elemzését, történelmi előzményeit, a feladat indokoltságát (a motiváció leírását), az eddigi megoldásokat, és ennek tükrében a hallgató megoldásának összefoglalását. A bevezető szokás szerint a diplomaterv felépítésével záródik, azaz annak rövid leírásával, hogy melyik fejezet mivel foglalkozik.

Nowadays, the question of storing, processing and displaying the data is becoming more and more important throughout every industry. The time, when the collected data was only useful for computers and specialists, passed. Today, the need for showing data in an easily understandable form is significant. It is no wonder, people through the whole hierarchy of a company would like to be well-informed about the results and the ongoing processes. In addition, it is getting highly valuable to be able to display vast amount of data in a way that even outsiders can comprehend.

Because of this trend, many technologies attempting to solve these problems have appeared, creating a wide variety of tools which organizations can use.

In enterprise-grade environments, the use of complex systems - so called data-pipelines - are becoming increasingly common. These tools provide an integrated solution for transforming and querying data coming from data sources built with different technologies. With the help of these data-pipelines, it is possible to collect many types of data, no matter the format or the frequency.

All these things for one reason, to prepare the data for machine or human decision-making.

1.1 Problem definition

Every organization needs to manage the sometimes cumbersome task of managing data. It is common, that this data does not come from one place, but from multiple sources, which can present various problems, especially if at some point, merging all the available data is necessary. It can happen for example, if the organization wants to visualization of the data to be in one place.

The most fundamental one is maybe the issue of the data format and the way of accessing the data. There are already numerous industrial standards available, however, using these together can cause difficulties.

In addition, each of these heterogeneous components need to be configured in different ways, with varying set of parameters. This can produce a heavy overhead for the data-specialists.

Thus, we can establish the notion, that an universal tool for integrating different types of data sources is highly desired.

1.2 Motivation

- one visualization tool (Grafana) for multiple datasources in on place (one consistent way of visualizing data)
- open-source development
- integration task -> get familiar with many new technologies
- Grafana: de facto open-source visualization tool

1.3 Goals

- creating a data-gateway for accessing and visualizing data
- using different datasources (different technologies, data formats)
- connecting the gateway to Grafana (industry standard for opensource data visualization)
- presenting the main datasources and their features
  - relational databases
  - time-series databases
  - key-value stores
- discovering available options for interactivity in Grafana
- design a data-gateway for connecting (two-way, duplex) different types of datasources to grafana
- implement a POC data-gateway for connecting a Python based and a RapidMiner based datasource
- present the advantages and disadvantages of the created gateway

There are a couple objectives to achieve during this thesis project.

first of all, an investigation is needed about the mainly available data sources and their varying features. This is necessary to get into context and to establish further decisions concerning this work.

Chapter 2

Background

2.1 Datasource

- data formats
  - JSON
  - XML
  - other??
- data sources and their features, use-cases, pros/cons?
  - relational db
  - timeseries
  - key-value
  - document
- with examples!

2.2 Grafana

Grafana is one of the most popular open source analytics and monitoring solution that can be connected to the majority of the main data sources out-of-the-box. It allows its users to query, visualize and alert on the collected metrics.

Altough Grafana has got plenty of useful feature, only those relevant to the scope of the thesis project will be briefly explained here

2.2.1 Data Source

Grafana supports many different storage backends (Data Source). Each Data Source has a specific Query Editor (see later) that is customized for the features and capabilities that the particular Data Source exposes. Of course, this leads to the fact that the query language and capacity of each Data Source are obviously very different.

Grafana mainly favors time series data (e.g. from Prometheus or InfluxDB), but it can work with other types of data source (e.g. relational databases (MySQL, PostgreSQL, MSSQL), logging and document databases (Loki, Elasticsearch)).

2.2.2 Panel

The Panel is the basic visualization building block in Grafana. Each Panel provides a so called Query Editor (dependent on the Data Source selected in the panel) that allows the user to create data source specific queries (e.g. an SQL query for a MySQL data source) in order to extract the desired metrics as precisely as possible.

There are multiple built-in Panel types available in Grafana, however, custom panels made by the open source community are also accessible. Probably the most widely used Panel types are the Graph, Table, Singlestat and Gauge.

— TODO insert example picture —

2.2.3 Query Editor

The Query Editor exposes the capabilities of the Data Source and allows the user to query the metrics that it contains. The queries created in the Query Editor of a panel determine what data will be displayed on the panel.

— TODO insert example picture

2.2.4 Dashboard

The Dashboard is where all the previously mentioned building blocks come together. Dashboards can be thought of as an organized set of Panels.

We can use the Dashboard to visualize different metrics in one, easily manageable place. This is quite useful, when many aspects must be taken into account in order to be able to thoroughly understand our currently inspected data set.

— TODO insert example picture

2.2.5 Interactivity in general

When working with and visualizing massive amount of data is a major part of one's profession, it can be quite convenient if the tools can provide the ability to allow some user-interaction. Meaning that the user do not have to work with static diagrams or rewrite complex configurations so that he or she is capable of further analyzing the collected data.

Commonly available interactivity options can be the followings:

- *statistic indicators* - easily accessible statistic information (e.g. average value, spread-diagram) about individual objects shown on the diagram (this is usually achieved by mouse-hovering)
- *local interactions* - changing the outlook of a diagram without affecting other displayed objects (e.g. reordering the bars on a bar-chart, rescaling the axis on a graph or other diagram-specific modifications)
- *selection and linked highlighting* - when we have multiple diagrams displaying different views of the same data set, selecting a subset of the data on one diagram (e.g. a bar on a bar-chart) highlights that particular part of the data on another diagrama (e.g. a set of points on a scatter-plot)

- *linked analysis*- for example, selecting a subset of the data triggers a reactive analysis, creating a statistic model (regression, scatter-plot, spread-diagram) using that specific data set

2.2.6 Interactivity in Grafana

While it is hard to find a tool, which possesses all the before-mentioned interactivity capabilities, Grafana provides certain features to enable efficient user-interaction for the good of solid understanding of the data.

2.2.6.1 Time Range Controls

Grafana provides numerous ways to interactively manage the time ranges of the data being visualized.

On the Dashboard-level, the 'Current time range' selector can be used to change the dashboard time. Doing this refreshes the whole dashboard with each panel on it. Those panels, which display time-dependent information, will only show data-points which timestamps are in the newly set time range.

— TODO insert example picture

If there is a Graph Panel on the dashboard, there is another possibility to change the Dashboard-level time range. We can select a time range on the Graph Panel with the cursor. This method is quite effective in case, when we would like to swiftly zoom into our data to review more refined details.

— TODO insert example picture

2.2.6.2 Variables and Templating

Variables in Grafana allow for more interactive and dynamic dashboards. We can use variables in metric queries and in panel titles.

Their role is similar to that of 'conventional' variables in programming languages. Instead of hard-coding things into the program, we can use them to create algorithms, that operate on a more abstract level, resulting in more effortlessly maintainable softwares.

In Grafana, we can use the variables to write generalized metric queries. For example, if we have multiple servers and we have metric data from all of them, we could store the name or address of the servers in a variable, rather than defining numerous queries, which only differ in the server name. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdowns make it easy to change the value of each available variable, thus to adjust the data being displayed in the dashboard.

— TODO insert example picture of variable dropdown

There are different types of variable that can be used to make dashboards more dynamic.

- *Query* - This variable type allows for writing a data source query that usually returns a list of metric names. For example, a query that returns a list of server names, sensor ids, or data centers.

- *Interval* - This variable can represent time spans. We can use them, when we need to aggregate data-points by time or date.
- *Data source* - This type makes it possible to quickly change the data source for an entire Dashboard. This feature can be quite suitable, when we have multiple instances of a data source in for example different environments.
- *Custom* - This variable can be used to manually define the value options for the given variable.
- *Constant* - This variable can be handy for example to easily declare metric path prefixes.
- *Text box* - This variable type displays as a free text input field with an optional default value.
- *Ad hoc filters* - It is a quite special kind of variable that only works with some data sources (e.g. InfluxDB, Elasticsearch). It allows the user to add key/value filters that will automatically be added to all metric queries the use the specified data source.

2.2.6.3 Basic statistic values in legend

Statistic indicators were already mentioned above, where we discussed interactivity in general. In Grafana, we can display some basic statistic values (minimum, maximum, average, total) in the Graph Panel.

— TODO insert example picture of this legend values

2.2.6.4 Data links

Data links provide a way to add dynamic links to the visualization. These links can link to either other dashboards or to an external URL. when using data links, additional built-in variables become available that enable creating dynamic links.

These variables are:

- *___series_name* - the name of the series or the table
- *___value_time* - the timestamp of the point that is clicked on (in millisecond epoch)
- *___url_time_range* - The current time range
- *___all_variables* - Adds all current variables and their current values to the URL

2.2.6.5 Creating custom interactivity

As Grafana is a fully open source project, it is possible for the community, to develop custom objects in Grafana or to extend built-in ones. This way, it is achievable to create custom interactivity features according to the needs of the user.

As a part of my thesis project, I successfully extended the built-in Graph Panel with some time-dependency related interactivity that is explained in detail in section ..... .

2.3 Relevant RapidMiner components

2.4 JSON

- JavaScript Object Notation
- lightweight format for storing and transporting data (w3school)
- often used when data is sent from a server to a web page (w3school)
- ”self-describing” and easy to understand
- syntax rules (need good source, currently w3school)
  - data is in name/value pairs
  - data is separated by commas
  - curly braces hold objects
  - square brackets hold arrays
- example

2.5 REST API

Chapter 3

Design

— TODO —

- extended API for SimpleJSON datasource

  querying available paramaters for a webservice

3.1 Architecture

The designed architecture of the project can be observed in the figure 3.1. There are three data sources connected to Grafana through an extended version of the official SimpleJson data source plugin. Each data source access the data it provides in different ways; through an outside service, from a database or from memory.

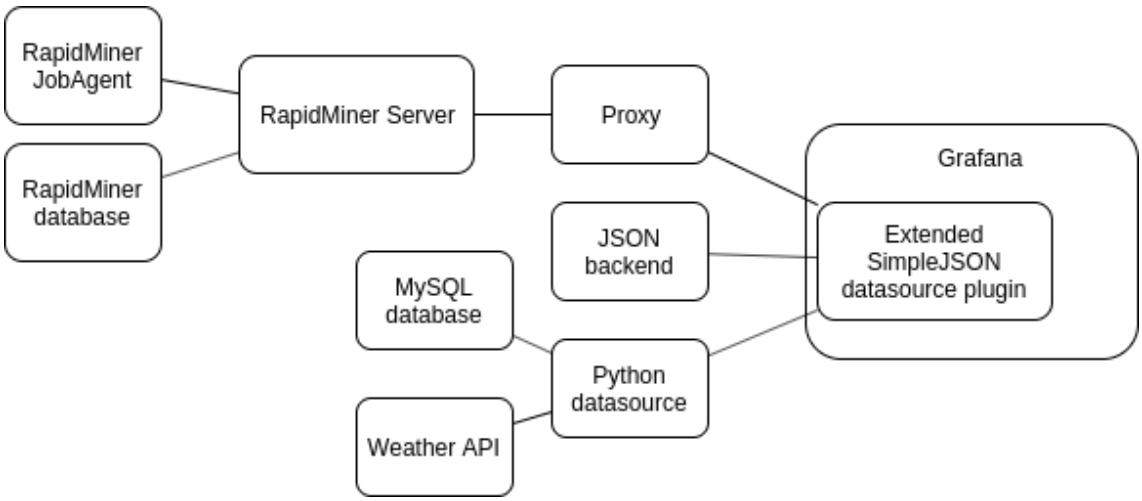

Figure 3.1: Architecture diagram

3.2 Components

In this section, each component is presented, focusing on their responsibilities.

3.2.1 Extended SimpleJSON datasource plugin

Grafana uses data source plugins to connect to different data storage backends. Each data source plugin exposes a Grafana specific interface which allows Grafana to communicate with each data source plugin the same way.

3.2.2 Proxy

The main responsibility of this component is to translate between the RapidMiner Web service which the RapidMiner Server exposes and the SimpleJSON datasource plugin. The problem is that RapidMiner Server exposes its result in JSON, but is in another format that the SimpleJSON datasource plugin accepts.

3.2.3 RapidMiner components

The RapidMiner Server acts as a data source for Grafana with the help of the proxy component. In addition to its numerous features, with RapidMiner Server we can store and expose data science processes, like model building, predictions, data ETL operations, etc. Long-running jobs are executed externally via Job Agents. The RapidMiner Server stores its data in an external database. The Server can expose certain processes that can be called via REST API requests. These are the so called Web services which we use to get the results of the RapidMiner processes and transfer them to Grafana.

3.2.4 JSON backend

This component is a example backend implementation for the SimpleJSON data source plugin. It serves as a base for creating other backends for the data source plugin. It also further expresses the general usability of the SimpleJSON plugin.

3.2.5 Python data source

Although Grafana has a built-in plugin to communicate with a MySQL database, there exists some use-cases, when having a custom component between the data source and the visualization tool is feasible.

With an extra component in the middle, we have extra control over the data which travels from the data storage backend (in our case, MySQL) to the visualization platform. This means that we do not have to rely solely on the capabilities of the database, which can lead to simpler queries, smaller communication overhead with the database.

Having a custom middleware makes it possible to implement the business logic in a separate component and only display business-relevant information with the visualization tool.

It also enables us to aggregate information from different backends and provide only one kind of interface towards visualization which can result in better maintainability.

In this project I demonstrate a proof-of-concept use-case, when this extra components reads information from a database and also from an external API service. I believe, that this is a quite common problem, to have some data in one's storage, but for a better business competence, the usage of external services is also necessary.

3.2.6 Weather API

This component represents the previously mentioned external service. Its responsibility is to expose a service that can be utilized by other softwares in order to acquire additional business-relevant information.

3.2.7 MySQL database

The Python datasource uses this database component to read some sample data from it.

- Grafana
- proxy/gateway
- python-datasource
  - python-datasource
  - mysql
  - weather-api
- RapidMiner stack
  - rapidminer-server
  - job-agent
  - database
- Grafana datasource plugin (extended API - parameters)
  - extended API for asking for available parameters
  - extended GUI, that dynamically lists available parameters
- Grafana extended panel plugin

Chapter 4

Implementation

- docker-compose
- communication flow between components
- interfaces API endpoints

4.1 Gateway

4.2 Pros

4.3 Cons

Chapter 5

Evaluation

Chapter 6

Future work

- Integrate proxy and Grafana RapidMiner datasource into one

Chapter 7

Related works

interactive-piechart-panel (github/eastcirclek) see notebook

Chapter 8

Summary

Acknowledgements

Ez nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

Bibliography