International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 07 | July 2022 www.irjet.net p-ISSN: 2395-0072

Design and Monitoring Performance of Digital Properties

Ankit Kumar Singh[1], B. M. Sagar[2]

[1,2]*Department of Information Science and Engineering, R. V. College of Engineering, Karnataka, Bangalore, India.*

Abstract - *Digital Property is a very wide term which considers all forms of electronic information ranging from a simple file stored inside a personal computer, images stored in hosted cloud storage systems, codebases stored on GitHub, etc. Businesses need digital properties for innumerable reasons, most important ones being – to maintain a strong social media presence, to provide insights on what it does and finally, to create valuable assets for themselves and boost their bottom-line. Ranging from API's to fully fledged web-based applications, all come under the category of Digital Properties. Naturally, due to many reasons, these digital properties may not work as intended by the owner and hence, monitoring the performance of these becomes an equally valuable task. This paper defines what a complete end to end digital property solution looks like using Django and tries to deliver the importance of such systems in modern day businesses. In this paper, by digital properties, the focus will be on two major properties – APIs and Web Applications.*

Key Words: **Digital Properties, Software Development Life Cycle, DevOps, Agile, Virtualization, API, Web Applications, Application Performance Monitoring.**

1. INTRODUCTION

A strong presence on internet has rather become necessary for all kinds of businesses and organizations. Not only at the top of the hierarchy, but even for individual teams and divisions, digital properties help in providing new and exciting ways to solve problems. These solutions in turn help the organization to boost their performance from grass root levels.

Although digital properties are necessary for the organizations, their design is a much-complicated task. There are large number of factors that are taken into consideration while designing such applications. Starting from the concurrency of the applications to support multiple users at the same time, to providing utmost sense of the security to the user, everything turns out to be a daunting task. Now a days, a wide variety of technical stacks are available which perform the job well. Large frameworks are being built to provide ease of development to the developers and at the same time reduce the time to deliver the applications. Apart from being extremely easy to use and with appropriate document and support, these frameworks are Open Source, i.e., one can view the whole codebase and can inspect, modify, and enhance it as per their needs.

After designing and creating the appropriate software, the question of its feasibility arises. If a certain software isn't feasible, rather than adding value to the organization, it will decrease their productivity and performance. Hence, monitoring the performance of the developed tools and softwares becomes an equally important task.

This paper focuses on design and creation of a web-based application using Django as the software stack and using the concept of Application Performance Monitoring to monitor the performance of the latter, along with all the APIs integrated within, hence providing a complete end to end solution with the help of concepts of Software Development Life Cycle and thereby following the appropriate steps.

2. RELATED WORK

Web development is not something which the world is unknown too. There exists numerous research, books, and data on web about web itself. They help in throughout development of projects and have the programmers and researchers to know about new advancements and mistakes of the past. Django is one such stack which has become popular in recent past for developing web applications in quick time. Django is credited with reshaping the web development through Python. "This actually takes input from the user converts it into some valid query and then fires it onto the database. It also takes input from the database after the query is fired and renders the result onto the screen." [1]

Most of the surveys done for this paper revolve around Application Programming Interfaces (APIs) – How they work? How can one monitor their performance? What are the criteria for measuring their performance? How to connect an API with an existing frontend? and so on.

In [2], everything starting from what an API is to how to monitor their performance – has been explained in brief. It also lists the various types of API's – SOAP, REST and JSON as well as explains the importance of monitoring API's and how one can achieve that. Finally, it gives insights on how Splunk's Observability Tool can be used for application performance monitoring.

[3] gives insights on how modern Application Performance Monitoring Tools (APMs) work. Being one of the best APM provider itself, DataDog documentation does its best to explain the modern agent-based tools wherein an agent is a software program that collects event metrics from the host that it is integrated in and sends the data to the monitoring tool which is then used to analyze and represent that data in much aesthetic ways. Such APMs can be integrated with most of the modern-day frameworks like NodeJS, Django, Java, etc.

Even Browser JavaScript has an in-built library for monitoring performance of web-based applications, Performance-API. [4] explains how this library can be used to analyze some of the metrics used in evaluating the performance of web applications. It also lists advantages of using Performance API and other related libraries which can be used in complement to latter. It mainly lists properties like High Resolution Time, Resource Timing, Paint Timing, etc. which can be used to monitor performance of a webpage. [5] is the official MDN documentation for the Performance API, listing all the classes and functions it comprises and brief details for the same.

On digging deeper, another APM, [6] gives not only observability, but also provides traces – the collected telemetry data. It provides a seamless flow between metrics and traces. If anything, suspicious is found in a web app transaction, one can click that transaction and view what's going behind the scenes and find a fix for the issue. It also provides application of advanced filters to trace data. Finally, it supports various intuitive representations like Gantt Charts to visualize the collected data efficiently.

3. METHODOLOGY

The methodology adopted to carry out the paper can be divided into 5 major steps. These are explained below.

3.1 Using Django Architecture for developing the application

The digital properties in highlight, i.e., the web apps and the API's, make use of Django as a framework for both frontend and backend. Django is a high-level Python web framework that allows development in quick time and is now popularly used to develop secure and maintainable websites.

Django simplifies the work for software developers and allows a softer learning curve as compared to other frameworks. Django mainly uses the following two concepts:

- Rapid Development: Developers can do more than one iteration at a time without starting the whole schedule from scratch
- Don't Repeat Yourself (DRY) concept: Developers can reuse existing code and focus on the unique one.

Django offers great compatibility with PostgresQL [7] and provides ease of development. The fact that Python is a comparatively slow language is compensated by the quick development time it offers due to availability of vast number of libraries. The Model-View-Template Architecture of Django also allows users to handle the databases much more efficiently.

Another reason for using Django is the security it provides. It provides protection against most common types of attacks in the form of middlewares [8]. And most importantly, Django is good for Search Engine Optimization which is very important for big businesses and organizations.

3.2 Utilizing crons to perform backend jobs

Crons are readily available on Linux based distributions for scheduling tasks that can be executed in future. These execute the assigned job periodically, for e.g., Execution of garbage collection script daily at midnight or to perform automatic backup every week and so on.

For most of the crons, there are three components, these are:

- Script to be executed,
- Command that executed the abovementioned script, and
- The action or output of the script.

Cron jobs in this project will be used to make changes in the database (perform CRUD operations) and to retrieve data from external API's. This section of the paper will be referred to as 'backend scripts' for the rest of the paper.

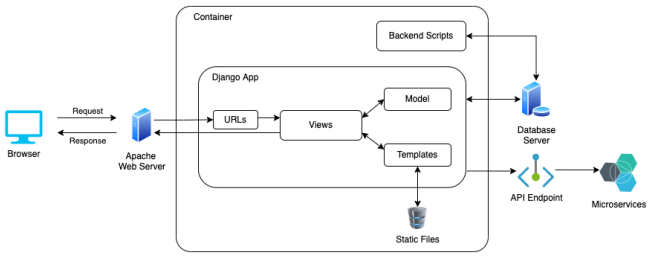

Fig 1. Proposed architecture for the web application.

3.3 Usage of external libraries to develop the frontend

HTML, CSS and JavaScript are the main constituent of the frontend part of the web application. Django also provides support via Django Template language.

Apart from that, several 3rd-party packages have also been used. These packages offer a wide range of features that are required for the application. DataTables is one such framework. It is used to display large amounts of data quickly in tabular format, at the same time offering functionalities like:

- Pagination
- Responsive
- Customized Search

The most important part, DataTables being open-source library. Another framework, SemanticUI, which is very similar to Bootstrap, has also been used. It offers human-friendly HTML which is used to build responsive and clean layouts. Classes are pre-defined, and the code is minimal which further helps in debugging.

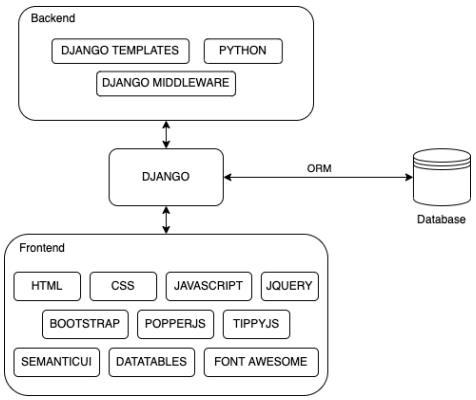

Fig 2. Frontend Tech-Stack

Another library that is being use is Popperjs which is yet another famous library for integrating tooltips in a web application. It provides inbuilt support for positioning tooltips and other effects which prevent overflow and clipping of the data.

3.4 Integrating an elastic agent in the application

After the application is up and running, hosted on a Linux based cloud hosted environment, monitoring performance of the application begins.

To achieve the objective, another software stack has been used, the Elasticsearch-Logstash-Kibana (ELK) stack. In recent past, the ELK stack has become very popular in the field of monitoring and observability tools development. Also, the code is open source, so users can experiment with it and develop applications as per their own convenience.

To use Elastic APM, an agent is installed inside the web application which takes note of all the processes happening inside it. This agent is then connected to the Kibana dashboards where it passes the data collected for Kibana to make sense of it. Logstash is used to get the application logs and send them to Kibana so that the user can get helpful visualizations out of those logs. Kibana is basically a dashboarding tool provided by team Elastic. It receives data in the form of indexes and provides different meaningful visualizations for the user to make sense of it.

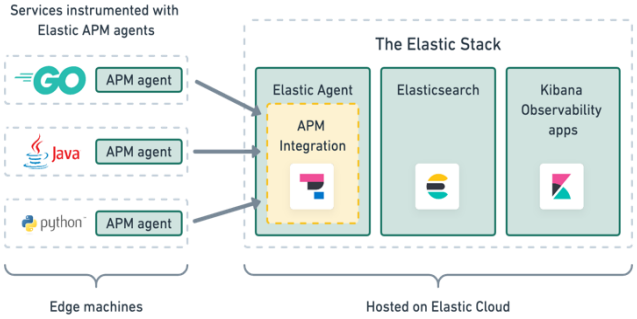

Fig 3. Elastic APM - Workflow

As a part of Elastic APM, one can also visualize system metrics, system health, etc. using different agents (referred to as Beats by Elastic). Elastic APM supports a wide variety of backend systems via which, one can design their web application.

Elastic supports a wide variety of integrations in the form of agents which cover almost all the systems. From monitoring the health of the host itself to monitoring the performance of the application, Elastic provides support for each aspect of monitoring that one might think of.

3.5 Monitoring the performance on Kibana

After setting up Kibana and connecting it with the elasticsearch agent, performance of the web application is monitored. Whenever a user opens the web application, the processes which take place are displayed as transactions on the Kibana dashboard. In depth details of a transactions are provided by Kibana.

Kibana also provides graphical representation of various metrics like latency, availability, success/error transactions, etc. to monitor the application. Using all these metrics, performance of a web application can be evaluated with much ease. The filtering and comparison capabilities provided by Kibana are also very much customizable as per user's needs.

Custom dashboards can also be created in Kibana using various available fields and creating visualization [9] upon them thereon. These visualizations can be built on top of the pre-defined ones, and hence provide information which is specific to user's needs.

4. ADVANTAGES

This paper provides insights on how one can create digital properties in the form of web applications and API's and provide complete observability and monitoring solutions for the same. This results in the creation of an end-to-end solution without needing any other input from and 3rd party as such.

Some of the most important advantages behind development of such applications are:

- Creation of Digital property allows the developer to experience various aspects of Software Development Life Cycle.
- These applications provide Businesses and Organizations with complete solutions which further boost their productivity.
- Exposure to various available technologies to design and create software systems.
- Provide monitoring and observability to the user, hence a complete solution.
- Based on the performance, one can evaluate where the performance of the digital property is lagging and take appropriate steps.

In this era of modern advances, users interact with digital properties for various content and services. Ensuring a high level of user experience and delivering seamless performance independent of user's geographical proximity has become the need of the hour. This is where performance monitoring plays its part. [10] is another such application which evaluates the performance of web applications and presents a set of metrics which can further help companies fine-tune their services and deliver services efficiently.

5. CONCLUSION

A fully fledged web-based application along with the complete flow of its development cycle has been explained in this paper. It provides insights on how Businesses and Organizations can use the Agile approach of Software Development Life Cycle to develop end to end complete solutions and enhance their efficiency.

The paper also provides exposure to Observability tools and the importance of Observability in modern software infrastructure, and the way observability is helping improve and optimize the efficiency of web-based applications.

REFERENCES

[1] Adrian Holovaty and Jacob K. Moss, "The Definitive Guide to Django: Web Development Done Right", pp. 17.

[2] API's in Action, [online]

Available: https://www.splunk.com/en_us/pdfs/resources/e-book/apis-in-action.pdf

[3] Network Performance Monitoring, [online]

Available: https://www.datadoghq.com/product/network-monitoring/network-performance-monitoring/

[4] How to practically use Performance API to measure performance, [online] Available: https://blog.logrocket.com/how-to-practically-use-performance-api-to-measure-performance/

[5] Performance API, [online] Available: https://developer.mozilla.org/en-US/docs/Web/API/Performance_API

[6] SigNoz Overview, [online]

Available: https://signoz.io/docs/userguide/overview/

[7] Why Django is so impressive for developing with PostgreSQL and Python, [online] Available: https://www.enterprisedb.com/postgres-tutorials/why-django-so-impressive-developing-postgresql-and-python

[8] Django Middleware, [online] Available: https://docs.djangoproject.com/en/4.0/ref/middleware/

[9] M. Kumar J., S. Dubey, B. Balaji, D. Rao and D. Rao, "Data Visualization on GitHub Repository Parameters Using Elastic Search and Kibana," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), 2018, pp. 554-558, doi: 10.1109/ICOEI.2018.8553755.

[10] L. Dane and D. Gurkan, "NetForager: Geographically-Distributed Network Performance Monitoring of Web Applications," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), 2020, pp. 0142-0149, doi: 10.1109/CCWC47524.2020.9031171.