Haaga-Helia
University of Applied Sciences

Case study:

Real user monitoring for internal web application

Linh Pham

Thesis

Degree Programme in BIT

2020

Haaga-Helia
University of Applied Sciences

Abstract

11.5.2020

Author(s) Linh Pham	
Degree programme Business Information Technology	
Report/thesis title Real user monitoring for internal web application	**Number of pages and appendix pages** **37 + 3**

This thesis was started from the request of a mobile game company in Helsinki, who has been developing a web application for their internal use for a few years. At the time right before conducting this thesis, the company was in need of a tool to monitor the application's performance and users in real-time.

The goal of the thesis was to find and implement a monitoring tool for the company's internal web application. The tool must be able to meet the company's requirement of providing data about the application's daily use, API calls, errors, and other difficulties that users may encounter.

The search for a suitable monitoring tool begun in March 2020. After Datadog's Real User Monitoring tool was chosen and configured, the data of the application that was collected in two weeks from 07 April to 22 April was analyzed for discussion in this thesis. The thesis was completed in May 2020.

From the results of the two-week monitoring period, the application showed good performance generally, but there were some abnormal behaviors in a few days. Besides, Datadog's Real User Monitoring tool proved to be helpful in monitoring the application's performance, users and in detecting anomalies.

Keywords
Web monitoring, real user monitoring, end user experience monitoring

Table of contents

Terms and Abbreviations

API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
FCP	First Contentful Paint
RUM	Real User Monitoring
XHR	XMLHttpRequest

1 Introduction

This project topic was requested by a mid-size mobile game company located in Helsinki. The company has been developing a web application for its internal use since October 2017. The application, from now on, will be mentioned as G-App in this research, is used by game teams, game testers and live operators of the company for viewing data from servers or managing content and player data in their games.

G-App has constantly been developed, maintained, and upgraded with new features assisting the work of game teams. However, the team that develops G-App does not know how G-App is actually being used, as there was never any data collected from users. The team does not know if G-App is useful for game teams, what users do on G-App, or how many frequent users there are on G-App every day. If there are any errors from G-App, the development team is often put in a passive situation where they only know if game teams report the problems. Here comes the need for a monitoring tool that not only can test the usability of G-App but also can monitor G-App's health and availability.

Usability testing before an application goes live is essential, but not sufficient. Users may be constrained by many other factors that prelaunch usability testing cannot anticipate (such as their environments, their browsers, their network connections). Therefore, they can behave differently compared to their behaviors in prelaunch usability tests (Croll & Power, 2009). Using a monitoring tool for G-App can help to track users' performance after deployment, the development team can learn about their users' behaviors and make needed improvements. Furthermore, a monitoring tool can also help the development team be more active in detecting problems and solving issues. In the end, not only the development team but also game teams can benefit from this project as well-maintained G-App can smooth their work and help to boost their work productivity.

1.1 The research objectives and research questions

The main objectives of this project are to find out the usability of G-App after deployment and to monitor G-App's performance. To do that, these following questions, which were also the information required by the company, need to be answered:

- How often is G-App being used daily?
- What do users do on G-App? What difficulties do users encounter?
- What kind of errors do users get when using G-App? Possible ways to log these errors for tracking purposes?
- How many API calls are made from G-App?

A suitable monitoring tool which is approved by the company will be configured within G-App, quantitative data will be recorded once G-App is deployed after that (such as measurements of load time, or counts of errors and page views). Descriptive analysis will be applied to analyze data. Results about possible problems and G-App’s performance will be concluded to serve the development team in their future work of development, support and, maintenance.

1.2 Out of scope

G-App is currently available in 3 environments: development, test, and production. Game teams and game testers use all three environments for different purposes. This research will only collect data from the development environment. Because development environment is deployed more often than the others, so it is suitable for the time limit of this project. Besides, usability testing before deployment will not be covered as G-App is already in use for a couple of years.

2 Theoretical background

This chapter starts with the basic knowledge about usability, then a brief introduction to functional and non-functional testing methods. Theoretical background chapter then continues with information about HTTP cookies. Before implementing a monitoring tool for G-App, it is necessary to understand what web monitoring is and know different methods of monitoring. Therefore, the definition about web monitoring is introduced next and followed with introduction to 2 popular monitoring methods: synthetic monitoring and real user monitoring.

2.1 Usability

This section is intended to present usability. The section starts off with an introduction about usability and its components, and then continues with the discussion about the importance of usability of intranets.

2.1.1 Definition

According to the Interaction Design Foundation (2020), the term “usability” was created to replace the outdated term “user friendliness” in the early 1990s, which refers to the ease of access, and ease of use of a product (in this section, the term “product” is used generally to refer to a website, a software, or an application). Since then, many different versions of usability definition have been proposed, which not only describe usability comprehensively but also associate usability with its measurable components. Hereby, usability definition given by International Organization for Standardization (ISO) (2018), Jeff Rubin & Dana Chisnell (2008) and Jakob Nielsen (1993) are presented in order, and the explanations of usability’s components are also mentioned after.

ISO defined usability in chapter 11 of its document about ergonomics of human-system interactions as “an extent to which a system, product, or service can be used by specific users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” (ISO 9241-11:2018). From this definition, it is understood that usability can be reached if the three components effectiveness, efficiency and satisfaction are fulfilled.

Jeff Rubin and Dana Chisnell (2008) associated usability with the absence of frustration in using a product, which means a user can do whatever he or she wants to do in expected ways without hindrance, hesitation, or questions. Also in this definition, attributes of usability were listed in more details: besides three components like in ISO’s definition (effectiveness, efficiency, and satisfaction), learnability and usefulness were taken into account.

Jakob Nielsen (1993) stated that usability is “the question of whether the system is good enough to satisfy all the needs and requirements of users and potential stakeholders, such as the users’ clients and manager”. He defined usability with five quality components: efficiency, satisfaction, learnability, memorability, and errors (Nielsen 1993; 2012).

Each definition above defines usability with different sets of components. The summary and comparison between usability’s components of the three definitions are highlighted in Table 1.

	ISO 9241-11:2018	**Rubin & Chisnell (2008)**	**Jakob Nielsen (1993; 2012)**
Usability	Efficiency	Efficiency	Efficiency
	Satisfaction	Satisfaction	Satisfaction
	Effectiveness	Effectiveness	Errors
		Learnability	Learnability
		Usefulness	Memorability

Table 1. Usability components by different definitions

Some usability’s components are all agreed by the three definitions, or two of them and there are also some unique components depending on definition author’s point of view. In this part, each component will be explained for further clarification. However, not all set of components will be explained but only the set of components defined by Jakob Nielsen because his definition is well-known, updated and in detail.

Efficiency means the quickness with which users can accomplish tasks once they are familiar with the product (Nielsen, 2012; Rubin & Chisnell, 2008). This measurement does not apply on novice users as every product takes time for users to learn how to use. The level of experience of users can be assessed by different methods such as certain amount of time using the product, learning curve measurement, or just simply by the confirmation of the user. Efficiency of a product can be evaluated by measuring the time it takes to execute a specific test task (Nielsen, 1993).

Satisfaction refers to the pleasure that users feel when using the product (Nielsen, 2012) or in other explanation, when the use of product meets user’s expectations and needs

(ISO 9241-11:2018). Satisfaction attribute can be measured by psychophysiological factors (such as heart rate, blood pressure), by asking users directly for their opinions or by conducting questionnaire survey. As psychophysiological approach can involve intimidating method, like taking blood sample, it is unsuitable for studies (Nielsen, 1993).

Errors concerns about the number of errors users made, the severity of these errors and the possibility of recovery from these errors (Nielsen, 2012). Errors include either minor errors which users can correct immediately and catastrophic errors which can destroy user's work. Catastrophic errors should be taken care of separately with more effort in order to eliminate them or reduce their frequency (Nielsen, 1993).

Learnability is defined by the ease of using the product in the first time (Nielsen, 2012) This is the most fundamental component of usability. Most products must be easy to learn as user do not spend time to learn the complete product but only part of it before starting to use it. Learnability can be measured simple method, like picking up a novice user for the product and observe if he or she is able to complete specified tasks successfully (Nielsen, 1993).

Memorability refers to the ability to reestablish proficiency in using the product for infrequent users (Nielsen, 2012). Rubin & Chisnell (2008) also refer to this attribute in their definition of usability, but as a part of learnability. Memorability attribute is assessed by testing performance of infrequent users, or alternatively, by conducting a memory test which means asking users questions about the product after they finish their test session (Nielsen, 1993).

2.1.2 Importance of intranet usability

In a research conducted by Nigel Bevan (2005), there are five benefits that usability can contribute to a business: development savings, e-commerce sales increment, product sales increment, usage benefits for employees, and finally, support and maintenance costs reduction. While e-commerce and product sales gain are advantages that only commerce websites can benefit from, the others are potential for all other type of websites, including intranet.

Time and cost of development process can be saved by early definition of user goals and usability objectives, as well as by early identification and resolution of usability issues (Bevan, 2005). For an intranet, user goals and usability objectives should be easy to identify because the company knows exactly who the users are and what environments the users are using (Nielsen, 2012).

Usability is also beneficial for employees who are the users of the intranet. Good usability not only helps employee learn and perform tasks faster, but also reduce the possibility of errors made by users (Bevan, 2005). Improvements on usability boosts employee productivity dramatically for intranet with bad usability (Nielsen, 2012). Even though the usability improvement for good usability intranet is not as huge, but research found that it still contributes to the increase in productivity and return on investment (Nielsen, 2007).

Usability helps reduces the costs and time for intranet support and maintenance (Bevan, 2005). When user's expectation and needs for an intranet are met, the users are trained faster with less support needed.

2.2 Web-based testing methods

Testing has been a standard practice in software development since a long time ago. The first definition about testing was created in the 1950s with a simple description "what programmers did to find bugs in their programs" (Lewis, 2017). Looking for problems has always been the fundamental of testing, the more issues found the better (Kaner, Falk & Nguyen, 1999). However, as software testing matures, its definition has also developed. Testing is not all about finding bugs anymore, but also about checking whether the product meets its specification and whether it fits company's purposes. Checking if a product satisfies its specification is a condition to determine its quality and readiness to be used. Checking if a product fits company's purposes means demonstrating that it is good enough for users to carry out their tasks (Graham, Veenendaal, Evans & Black, 2008).

In order to execute a web application, not only the application itself but also its running environment (the whole infrastructure of software, hardware and middleware components) are needed. Therefore, testing a web application should be considered from two points of view: the application's non-functional requirements (its running environment) and functional requirements (the application) (Lucca & Fasolino, 2006)

Non-functional test

Non-functional test is targeted for verifying the quality characteristics of a web application. (Desikan & Ramesh, 2006). This type of test concerns about how well or how fast the application is performing (Graham, Veenendaal, Evans & Black, 2008). Lucca and Fasolino (2006) listed different kinds of non-functional test for web application as followed:

- Performance test
  The objective is to verify system performances (such as response time, service availability). This test is important as long response time or service unavailability can exhaust web users.
- Load test
  This test is used to measure the time needed for the application to perform specified tasks under certain conditions.
- Stress test
  The goal of stress test is to evaluate application responses during activity peaks. The application will be observed to check if it crashes and how well it can recover from such conditions.
- Compatibility test
  Compatibility test deals with the possible failures of the application in different web server platforms, different web browsers, different release versions or their configurations.
- Usability test
  The aim of usability test is to find out to what extend the application is easy to use and user-friendly.
- Accessibility test
  Accessibility test is considered a type of usability test, in which the access to content of the application will be tested in a particular condition of client side.
- Security test
  The aim of security test is to verify if the application defenses effectively against security issue (such as such the access of unauthorized users or improper use the resources of the application).

Functional test

In functional tests, an application's functionality and features are tested (Desikan & Ramesh, 2006). To simply put, functional test means testing what the application does. Functional test can be carried out by two methods: requirement-based or business-process-based. Requirement-based test uses functional specifications of the application as the basis for the test. On the other hand, business-process-based test describes the daily working scenario involving the application (Graham, Veenendaal, Evans & Black, 2008).

The normal software development life cycle includes five phases: requirements, design, development, test and operation. Software testing can be done at any phase of the process, not necessarily in the test phase (Kaner, Falk & Nguyen, 1999). The earlier a failure is detected, the cheaper it is to fix (Figure 1) (Graham, Veenendaal, Evans & Black, 2008). A study conducted by Martin & McClure (1983) pointed out the cost of failure in requirements, design, development phases are respectively 3%, 3% and 5% of the development cost. On the other hand, the cost of failure in test and operation phases are 15% and 67%.

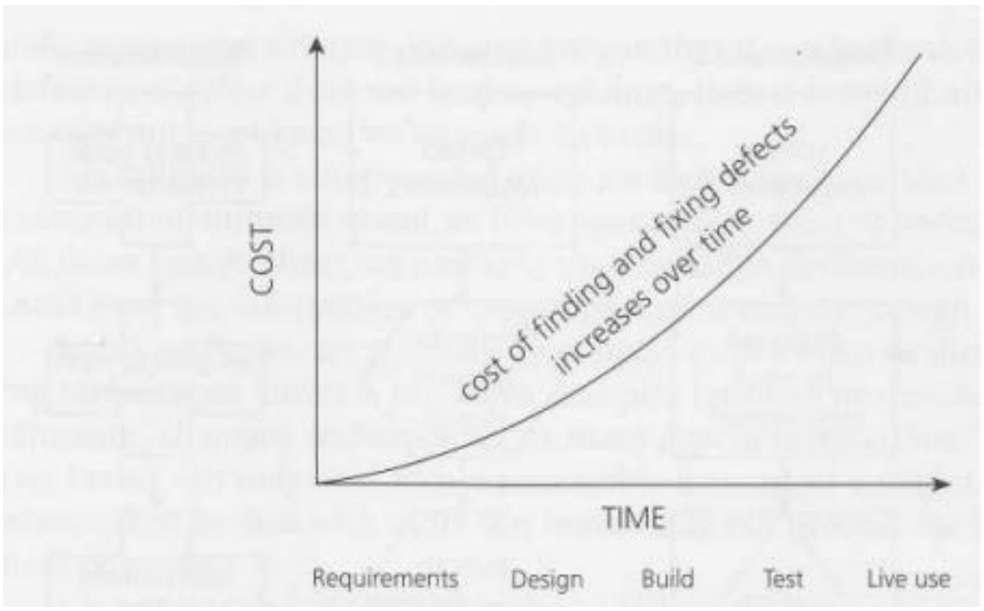


Figure 1. Cost of defects (Graham, Veenendaal, Evans & Black, 2008)

2.3 Cookies

The Hypertext Transfer Protocol (HTTP) is designed for stateless operation, that means originally, each request sent from the client to the server is treated independently of the previous one despites of the same user. The statelessness is useful for building web browsers and servers, but not for web applications (Kristol, 2001). For example, a user adds some items to an online shopping basket, he/she needs some time to think whether he/she would buy it so he/she closes the web. Next day, he/she decides to purchase those items and comes back to the web, the shopping basket is empty because the server does not know that is the same customer from yesterday, so it does not return yesterday items in his/her shopping basket. To solve this problem, cookies, also known as HTTP cookies, were invented by Netsacpe in order to identify users and allow persistent sessions (Aggarwal et al, 2002). Cookies are generated and modified by the server, stored by the client and transferred between server and client in each interaction (Tirtea, Castelluccia & Ikonomou, 2011). Besides storing data of the same user, cookies also help to track which pages users visit to better understand how users navigate throughthe site (Kristol, 2011).

2.3.1 Types of cookies

Cookies can be categorized from different perspectives. If looking at cookies through life span perspective, cookies are divided into two groups: session cookie and persistent cookie. In a different way, if cookies are considered by its generator, cookies are classified into first-party and third-party groups (Tirtea, Castelluccia & Ikonomou, 2011).

Session cookies

Session cookies are also known as non-persistent cookies or temporary cookies. Session cookies is expired once the user closes the browser or the sessions time out (Aggarwal et al, 2002). Session cookies are usually stored in cache memory (Tirtea, Castelluccia & Ikonomou, 2011).

Persistent cookies

Persistent cookies are stored on disk. They live longer than session cookies and survive browser closes or even computer restarts. However, they still have an expiration date (Aggarwal et al, 2002).

First-party cookies

First-party cookies are created by the site user is visiting. They are connected to the webserver of the URL of that page (Tirtea, Castelluccia & Ikonomou, 2011).

Third-party cookies

Third-party cookies are not created by the site users are visiting, but by the site linked to the current site as third-party content. Third-party cookies are widely used in advertising industry (Tirtea, Castelluccia & Ikonomou, 2011).

2.3.2 How cookies work

The process of how cookies work can be broken down to 3 steps (Tirtea, Castelluccia & Ikonomou, 2011):

1. The client sends first request to the server.
2. In response for this request, the server include a piece of arbitrary information in the response header, which is known as cookie. This arbitrary information can be anything (such as a user identifier, a database key) for the server to know where it left off (Kristol, 2001). The cookie is stored in the client.

3. The next time the client sends any request, it will include cookies in its request header. The server relies on the cookie to save the state and return the correct content (Kristol, 2001).

2.4 Web monitoring

Monitoring is to the collection of tools and processes used to ensure the availability and health of the IT systems and services (Dixon, 2017) by surveilling their existence, extend of state changes, and dataflow (Ligus, 2012). The purpose of monitoring is to detect faults and be the groundwork for eliminating those defects (Ligus, 2012). Web monitoring mentioned in this research refers to monitoring front-end parts of any web application, such as its static assets (images, CSS, Javascript), or API calls (Julian, 2017).

A monitoring tool should identify when resources of the application become unavailable or when the application performs poorly (Dixon, 2017). Unavailability of resources leads to downtime of the application, which is a dread for any system or application. Downtimes or slow page loads can reduce the effectiveness of even the most usable website (Croll & Power, 2009) and they are usually the cause for losses of revenue (Ligus, 2012). For example, one minute downtime of Amazon back in 2013 resulted in approximately 67000 USD loss in revenue of the company (Clay, 2013). Monitoring tools can help to check the availability of the website by detecting either hard errors (like abruptly terminated connection) and soft errors (like broken navigational paths) (Croll & Power, 2009). Moreover, monitoring tools also assist in looking for any anomalous behaviors in the performance of application which cannot be found in the previous development phases (Ligus, 2012).

2.4.1 Categories

Molyneaux categorized web monitoring in one of his books (2014) into 2 groups: active monitoring and passive monitoring. Active monitoring is synthetic monitoring, which is prescheduled, repetitive automated tests. On the other hand, passive monitoring is also known as real-user monitoring, (RUM) which watches users traffic on the website.

Mastin had a different opinion about the categorization of web monitoring in his research about RUM (2016). According to Mastin (2016), RUM is about actions that users initiate, while synthetic monitoring is about actions that computers generate. RUM and synthetic monitoring can either be active or passive. He defined active monitoring as a controlled experiment where all actions are planned for monitoring. Passive monitoring, in contrast, is data recorded from users’ input without any pre-created actions. As a result of this categorization, there are four groups of web monitoring (Mastin, 2016):

- *Active RUM*: users’ actions that trigger data to be sent. This method helps to test any features that could lead to an issue before it happens by getting data initiated from users’ actions.
- *Passive RUM*: users’ traffic, behaviors, performance is logged and tracked. This method makes it possible to detect problems in real-time by showing what is happening on the website.
- *Active synthetic monitoring*: tests that are generated from a device located in multiple network points. This method accommodates regular, systematic testing that uses an active outbound monitor.
- *Passive synthetic monitoring*: tests that are sent out from fixed locations at fixed intervals. This method can be used in implementing regular, systematic testing using human/environmental elements as a trigger.

2.5 Synthetic monitoring

Synthetic monitoring includes tests that simulate users’ traffic on a website to measure experience on that web application. From the web monitoring tool point of view, synthetic monitoring shows what the website’s performance is like for end-users, and it usually involves a complete user interface (UI) or browser session (Lever, 2016).

A web application includes many components: DNS, routers, CDNs, load balancers, third-party components, servers, database (Figure 2) (Croll & Power, 2009).

- DNS (Domain Name System) is what resolves hosts or mail server to IP address (Steen & Tanenbaum, 2017).
- Routers connect networks and move packets from one network to another (Moore, 2017)
- CDN (Content Delivery Network) acts as an infrastructure for distributing and replicating documents of multiple sites across the Internet (Steen & Tanenbaum, 2017).
- Load balancers handle load balancing process which distributes the traffic among available servers to improve both resource ultilization and response time (Moharana, 2013).
- A server is a process of implementing a specific service, such as a file system service (Steen & Tanenbaum, 2017).
- Database is a scheme of grouping a set of integrated data files together (Kahate, 2004).

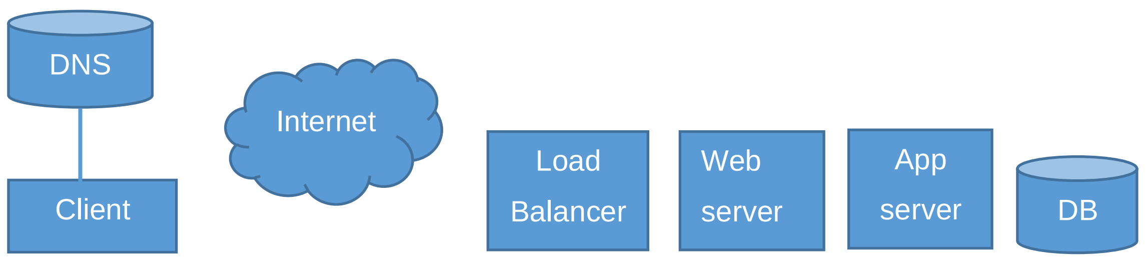


Figure 2. Web infrastructure components (Croll & Power, 2009)

Synthetic monitoring from the web application point of view is monitoring components on the left of the Internet component (Figure 2) (Croll & Power, 2009). It is called external synthetic monitoring. Synthetic monitoring on components on the right of the Internet component is also called internal monitoring, which will not be included in the scope of this research.

There are different tests for different components, but they all deliver data about the health of those components and help to determine issues. (Croll & Power, 2009)

a) *Testing DNS*
   DNS test is included in synthetic testing and is shown in the test's result. DNS test watches the response time for DNS lookups, DNS resolution of every site involved in building a page and also check if DNS lookups return the correct IP addresses.

b) *Testing network connectivity*
   Network connectivity can be tested by a ping, which is a test where a single packet is sent to get a response back. The ping includes the number of bytes received, the address of the host, and a sequence number for tracking any lost packets. A ping ends with a summary of the results. Generally, the purposes of ping tests are to check connectivity, packet loss, and latency for any kind of application. Ping test is easy to run and a basic test for checking Internet reachability. Synthetic monitoring tools can help to create ping tests at regular intervals.

c) *Testing website service*
   Testing DNS and network is not enough. A reachable website does not ensure the right content. Synthetic monitoring tool can help to send test by asking a content from the server. A handled request can be confirmed by checking the response code sent back in the test and also the total response time.

2.5.1 How it works

Croll and Power showed in their book (2009) how synthetic monitoring works. It is a simple 6-step process that is visualized in Figure 3.

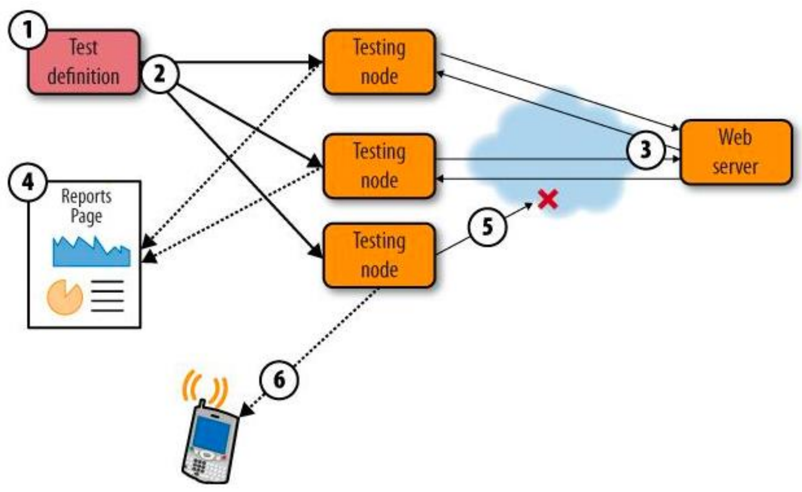


Figure 3. The basic steps in synthetic monitoring (Croll & Power, 2009)

In the beginning, all needed tests are configured, and then the service sends out the tests to different nodes. Those nodes run tests at regular intervals and record their performances. The testing results are saved in reports. If an error occurs, the node can capture it and even make a copy of that error, and then an alert will be sent to the company’s device, notifying about that error.

2.5.2 Advantages and disadvantages

Synthetic monitoring has either pros and cons. Understanding the advantages and disadvantages of synthetic monitoring benefits companies in making decisions and also in implementing the works.

Advantages

Synthetic monitoring agents can be expanded to many locations (Mastin, 2016). This advantage is possible if the service is of large synthetic monitoring providers as they have data centers with multiple IP providers. The tests can include many networks from all the available locations. This advantage can be beneficial for e-commerce or global websites as development teams want to check how the websites behave for different parts of the world.

Synthetic monitoring can provide regular monitoring independent of real users (Mastin, 2016). As synthetic monitoring is a stimulation of users' traffic on the website, it can help to catch problems at its regular intervals without real users on the site despite time of the day. (Lever, 2016).

Synthetic monitoring helps to test the website or the application before deployment (Mastin, 2016). Because synthetic monitoring does not require real users to perform the tests, it can be used to test the website or the application when they need to be tested right before going live.

Disadvantages

Synthetic monitoring does not reflect real users' experience on the website (Croll & Power, 2009). As synthetic monitoring does not include real users, synthetic monitoring cannot stimulate precisely the diversity of network connections, bandwidth, browser, or desktop. As a result, synthetic monitoring can deliver unrealistic results or only a subset of true conditions of the application (Molyneaux, 2014).

Synthetic monitoring may not test every page and every navigational path (Mastin, 2016). The real users may access some part on the website that synthetic monitoring does not test or go to a path that is not anticipated. If there are any errors in those places, they will not be recorded; hence, the development team may not be aware of those errors.

2.6 Real user monitoring

RUM is an abbreviation for Real User Monitoring, which is a method to collect measurements generated from actual users on a website or an application (Mastin, 2016). RUM is used to check a site's performance and availability from real users' perspective, so the quantity of the results is dependent on the number of users on the site.

The data captured by RUM tools can be both from the server-side or client-side (Croll & Power, 2009). From the server-side perspective, the RUM tool gathers information of individual HTML objects, organizes them into pages, visits, and groups them together under some common characteristics. From the client-side point of view, the RUM tool records key events of page loads in the browsers such as the start of a page or the time all data are retrieved from servers. Usually, RUM vendors offer a hybrid solution that includes both server-side and client-side monitoring for a complete perspective on end-user experience.

Many metrics can be recorded from a web page, the browsers expose those metric via Navigation Timing API which is a specification recommended by the World Wide Web Consortium (W3C) (Julian, 2017). However, not every RUM tool provides all of those metrics, so tool research should be conducted in order to help companies find the most suitable RUM tool for their business. In general, metrics collected by RUM can be categorized into performance metrics, headers and DOM information, error conditions, page content, and external metadata (Croll & Power, 2009).

a) Performance metrics

Some metrics under this categorization are request time (the time for a client to sends the server a request), first byte (the time for the server to respond with a status code and response object), redirect time, network time (the time to deliver an object to the client), page render time, end-to-end time (the total time to request, receive and display a page), etc.

b) Headers and DOM information

All HTTP objects contain information about the request and response in the headers sent between a client and a web server. RUM tools can help to collect metrics of both requests and responses. Some request metrics are browser type, URL, parameters, referrer (the referring URL that triggers the browser's request), and cookies. And response metrics can be recorded are object size, HTTP code, last-modified date, object type, response content, cachability information, cookies.

c) Error conditions

RUM tools help to detect errors from low-level network error to bad HTTP status code or specific content on a page.

d) Page content

Some RUM tools extract additional information about page content and associate it with a view or a visit. Even metadata, like the total amount of time spent on the current page, can be recorded. Collecting page contents of all views/all visits assists in detecting regular problems.

e) External metadata

Information about visitors' IP address and municipality, country, carrier in the session log can also be captured. This information helps the development team get insights about their users' origins. However, for an internal website of a company which is located at a specific location, the data about users' origins cannot be as helpful as all of the users are under one roof.

2.6.1 How it works

Croll and Power (2009) broke down the RUM process into 6 specific steps:

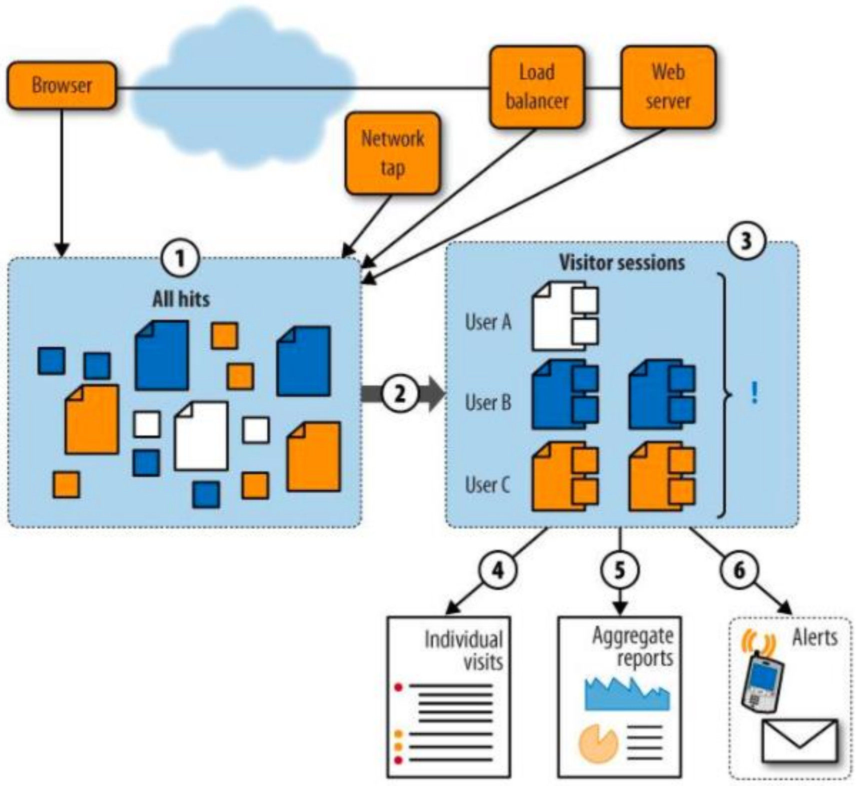


Figure 4. The basic steps in RUM solutions (Croll & Power, 2009)

1. *Capture*

   The monitoring tool starts to capture pages and objects from many resources like Javascript on a browser or server log files, etc.
2. *Sessionization*

   The tool collects data and organizes it into a record of the pages and components along with timing information of every single visit through session cookie.
3. *Problem detection*

   The recorded data is examined to find any interesting happenings of errors, slowness, and many more possible problems.
4. *Individual visit reporting*

   Every individual visit can be reviewed as they are summarized after being captured.
5. *Reporting and segmentation*

Here comes the aggregate data like the availability or performance of a particular page.

6. *Alerting*

   Any urgent issues that the RUM tool detected can trigger alerts.

2.6.2 RUM collecting methods

As mentioned above, there are 2 approaches to collect users' experience by RUM tools, which are server-side and client-side. Server-side data collecting methods include server logging, reverse proxies, and passive analyzers (inline RUM). On the other hand, client-side data collecting methods which can be named are desktop agent and Javascript instrumentation in the browsers (Croll & Power, 2009).

Server logging

This method provides basic information about user performance, such as what data was requested, the time at which data was requested, the time at which a response was delivered, or when the server finished sending the response object. This method is needed to detect issues or frequent incidents; however, it is mostly beneficial when combined with a client-side monitoring method.

Reverse proxies

Reverse proxies have not been a popular collecting method. As reverse proxies are located between the web server and the client, it introduces an additional point of failure, which is hard to diagnose. However, its location gives reverse proxies a different advantage from the other method that it can measure the health and performance of a network from both ends of a connection.

Passive analyzer (inline RUM)

This method uses a device (also called a tap) or a spare spot on a network switch to duplicate every packet that goes through it. This approach is powerful as it can still detect problems in the cases where the server is down, or the browser does not load. Nevertheless, inline RUM cannot provide an ultimate view on users' experience as it is located between a server and a client (like reverse proxies), not on a browser itself.

Desktop agents

Agents are installable software applications on client desktops. Agents can watch every user's behaviors on not only web application, but also on other applications. Agents can even know how healthy is the network and how much resources of the CPU are being used because they have access to the client’s operating system. As might be expected, agents are probably cannot be used for commercial purposes.

Javascript

Real user monitoring by Javascript is implemented by putting a snippet of code to web pages that need to be monitored. When the users browse web pages, they download a monitoring script that runs on their client. Javascript can get all the information that the browser knows about the session or the user. However, Javascript is page-specific, which means it records what happens on each page separately. When the user leaves a page, a record on that page ends. This feature helps to promote security and privacy for users as what users do outside those Javascript-tagged pages are out of sight.

2.6.3 Advantages and disadvantages

Similar to synthetic monitoring, RUM also has its advantages and disadvantages. Knowing what RUM can do and its limitation is beneficial in choosing the right monitoring tool for websites or web applications.

Advantages

All of the measurements are taken from real users (Mastin, 2016). This advantage ensures the accuracy in collected data as the data reflects precisely what the users experience on the website from their network, their browser to their behaviors.

A large number of diverse data can be collected from different locations (Mastin, 2016). This characteristic is extremely beneficial for global websites as it would help to understand user’s behaviors from different regions in the world. The more users the website has, the more data the monitoring tool can collect.

RUM can provide real-time alerts of errors that users are experiencing (Mastin, 2016). As some RUM tools have the ability to log errors of real-time users, the errors are more reliable as real users are running the tests.

Disadvantages

RUM cannot be used before deployment (Mastin, 2016), because at that time, there are no users on the website. Synthetic monitoring is better in that case because it does not need real users to create traffic. However, RUM can still be used if the website is deployed to a specific number of users for testing before really going live.

RUM cannot collect enough data in non-peak hours (Mastin, 2016). Normally, websites have many traffics during the day, but traffics drop at night time. This issue could lead to insufficient data at night, which prevents RUM from detecting problems if there are any.

2.6.4 RUM tool example – Google Analytics

Google Analytics is a free analytics tool developed by Google. Google Analytics is a type of RUM (Julian, 2017). In order to setup Google Analytics within an application, a Javascript tag is pasted into every web page that users' interactions need to be measured (Google, 2020).

According to Google (2020), Google Analytics collects data based on 3 sources:

- *The HTTP request of the user*: each request carries information about the request and the computer sending this request, such as the hostname, the browser type, referrer and language.
- *Browser/system information*: The DOM of browsers can provide more details about browser and system information, like Java and Flash support and screen resolution.
- *First-party cookie*: Google Analytics sets and reads first-party cookies on user's browser in order to get information about user session and ad campaign on every page request.

Google Analytics gathered data and categorized them into 5 main types of report (Google, 2020):

- *Audience report*: This report gives the insights into the characteristics of users on the site. The data include information about demographic (age, gender), geo (language, location), behavior (new or returning, frequency and recency, engagement) and many more.
- *Advertising report*: Google Analytics allows to integrate Google Ads in order to measure post-click performance metrics of users who clicked through the ads to come to the website. Moreover, there are collection of advertising features supporting digital marketing activities of business.
- *Acquisition report*: This report provides data about how website acquires users, what are users' behaviors on the site after acquisition and their conversion patterns.
- *Behavior report*: This report delivers information of site speed, site search, events, behavior flow and so on. Especially in the behavior flow report, the paths user

travels on the website are visualized to understand what content engages the users and identify potential content issues.

- *Conversions report*: Goals and E-commerce tracking of website are set before conversions report gathers data to evaluate how activities on website are conversed to the success of business, such as newsletter signups or purchases.

3 Implementation plan

3.1 Method

Considering between the two methods of web monitoring, RUM appears to be a more suitable monitoring method to the objectives of this research, and also to the company’s requirements.

The first and also most important reason is that the company wants to learn about users of G-App. They want to know how end-users use G-App and what their behaviors are like. This reason is compatible with the essence of RUM: monitoring real users. RUM makes it possible to capture data reflecting exactly real user experience while synthetic monitoring just produces user stimulations.

Secondly, as mentioned in part 2.3.3, RUM can report real-time errors that users are encountering. Even though synthetic monitoring can also detect possible errors, it does that at regular intervals, so the data is not real-time and not generated from real users. As the intention of the company is to find out difficulties and errors that end-users have, RUM is a better choice in this case.

Thirdly, real user behaviors can be unpredictable. They can go to some paths of the website and perform some actions that are not anticipated. If using synthetic monitoring, user experience in those paths is not recorded. However, with RUM, user behaviors will be recorded as long as users are on the website, regardless of which path of the website they visit.

Last but not least, G-App has gone live for a few years. Though it is continuously maintained and upgraded with new features, all of the existing features have not been tested thoroughly. Both synthetic monitoring and RUM can be used for testing a deployed web application. However, as RUM is a better choice for all other reasons above, it is more convenient for using RUM in this case as well.

3.2 Tool

After agreeing on using RUM to monitor G-App, research about available RUM tools in the market has to be conducted in order to find the most suitable tool for the company. The company is open to all options.

The first option to consider is Google Analytics as it is a popular free tool with a big support community. However, when looking at the metrics Google Analytics provides (part 2.6.4), Google Analytics appears to be an unsuitable tool for monitoring internal websites. Google Analytics focuses on metrics that are beneficial to marketing and advertising purposes. Three out of five reports that Google Analytics creates (advertising, acquisition and conversion reports) are unnecessary for G-App. Because G-App is used by the employees of the company, there is no needs to acquire new users, measure conversion rate or do advertising. More than that, some metrics in other reports also are not useful for G-App also. One example is the bounce rate metric in behaviour flow report, which means the percentage of single-page session on the site. For internal website in general, and G-App for specific, users can visit only one page of the website because that is all needed for their job, not because they lose the interest on the website.

As the company is using the Datadog monitoring tool at the moment for monitoring their servers' performance, Datadog is the second option to considered. The company suggested to prioritize checking Datadog before any other tools to know if Datadog offers the RUM tool fitted the company's purposes.

In December 2019, Datadog announced its new features in the Dash conference, which include its RUM tool. The tool was promised to collect, visualize, and analyze data about user interactions, help developers to troubleshoot user-facing issues and, make decisions based on user data. In other words, Datadog's RUM tool helps to understand the health and performance of the company's services and how they align with the company's business objectives (Lentz, 2019).

Datadog's RUM tool is a hybrid web monitoring tool as it has the characteristic of both active and passive RUM tool (part 2.1.1). Some metrics that can be collected by Datadog are listed in the below table, along with their purposes.

Metrics	**Definition**	**Purpose**
Session	A user journey on application which includes: • Multiple pageviews • Associated telemetry Expires after 15 minutes of inactivity Maximum duration is 4 hours	Know what users do on the application Know how often the application is being used

View	A user visit on a page (Resource, error, long task, and user action are attached to each view)	Know total views and what happens on the view
Resource	Trace XHRs/Fetch requests to its corresponding back-end	Trace API calls from the application
Error	All errors appear on users' console	Track errors
Long task	Tasks that take more than 50ms to complete	Know what tasks are long tasks, and theirs causes
User action	A specific action that its interaction needs to be monitored	Understand user interaction on the application

Table 1. Metrics collected by Datadog (Datadog documentation, 2020)

From Table 1, it is indisputable that metrics provided by Datadog's RUM tool can fulfill the company's purpose. Besides these metrics, there are performance metrics such as page load time or pageviews will also collected by Datadog. However, specific monitoring metrics assisting this research's purposes will be chosen later in the implementation phase. As the company is already collaborating with Datadog, they decided to use Datadog's RUM tool for monitoring G-App.

3.3 Implementation phases

There are 4 phases in the implementation plan of RUM tool in G-App. The four phases are listed in order as followed:

1. *Setting up RUM for G-App*
   In the first phase of implementation, a unique identification including application id and client token for each environment of G-App must be created in Datadog application. Even though the scope of this research is monitoring only the development environment, identifications for test and production environments are also created for further plans of the company. After that, RUM code snippet will be place in G-App's source code. This phase is considered completed and successful if the Datadog dashboard starts to display data collected from G-App.

2. *Collecting data*
   The next phase in the project is collecting data from the real users of G-App. The data of 15 latest days will be collected for further analysis because Datadog keeps data at full granularity for 15 days (Datadog documentation, 2020).

3. *Data analysis and summary*

   The summary about G-App, such as number of views, number of errors, number of long tasks, etc., will be stated in the report. After that, the collection of data will be analyzed in order to understand the application's performance.

4. *Documentation*

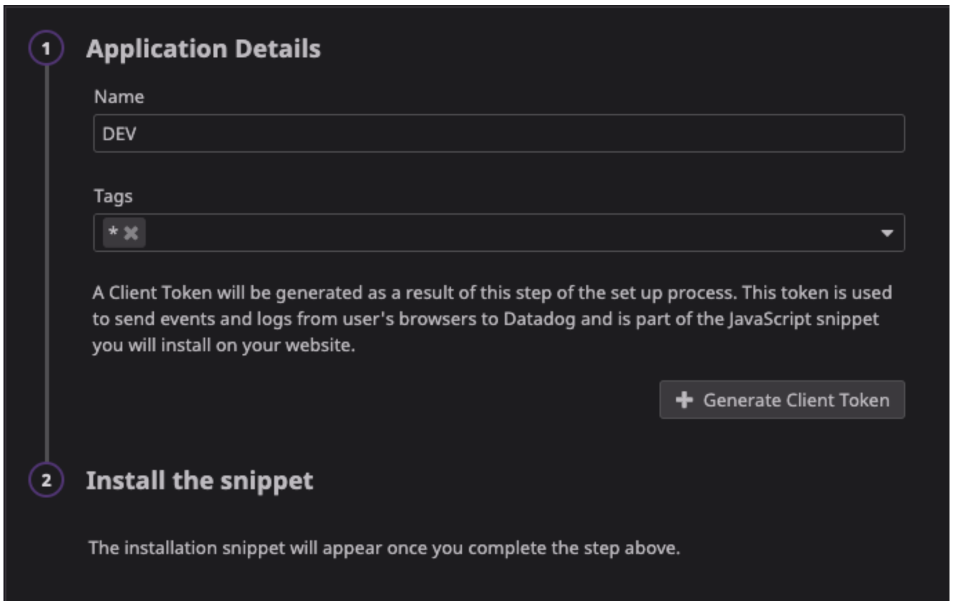
   As requested by the company, documentation about the RUM tool on G-App will be written after all the above phases are completed. This documentation aims to help the other developers in the team understand RUM setup and the data collected by RUM.

4 Implementation

4.1 RUM setup

Datadog’s RUM tool is installed in the application by pasting a Javascript code snippet into the application’s source code, which includes unique identifiers (application Id, client token) of the application. The unique identifier is generated in Datadog’s web application, where collected data can also be found later.

As mentioned in part 1.2, G-App is available in 3 environments (development, test, and production). Each environment has different usage purposes and consumes different API resources. Therefore, each environment is monitored separately for better performance tracking, error solving, and different unique identifiers are required for each of them.



1 Application Details

Name

DEV

Tags

*

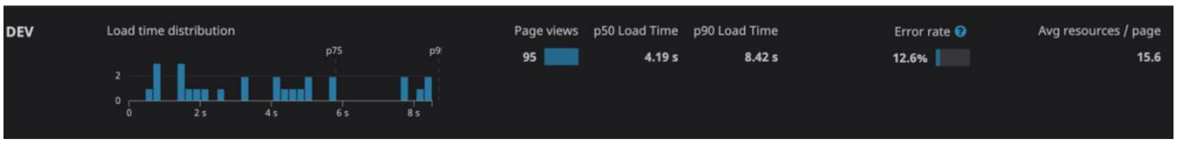
A Client Token will be generated as a result of this step of the set up process. This token is used to send events and logs from user's browsers to Datadog and is part of the JavaScript snippet you will install on your website.

2 Install the snippet

The installation snippet will appear once you complete the step above.

Figure 5. Datadog application interface for creating unique identifier

After generating a unique identifier for each environment, the code snippet should be put into every HTML page that needs to be monitored. However, G-App is a React application that contains only one main HTML file, so the code snippet is put into that *index.htm* file (Appendix 1). Datadog web application starts to display collected data from the development environment (Figure 6), so the setup was successful.



DEV

Page views	p50 Load Time	p90 Load Time	Error rate	Avg resources / page
95	4.19 s	8.42 s	12.6%	15.6

Figure 6. Datadog RUM application displaying collected data of development environment

4.2 Collecting data

Of all the data collected bu8 Datadog RUM tool, data of some specific metrics will be observed thoroughly, because they are helpful in answering research questions (part 1.2). Below is the list of research questions and the metrics support the answers to those questions.

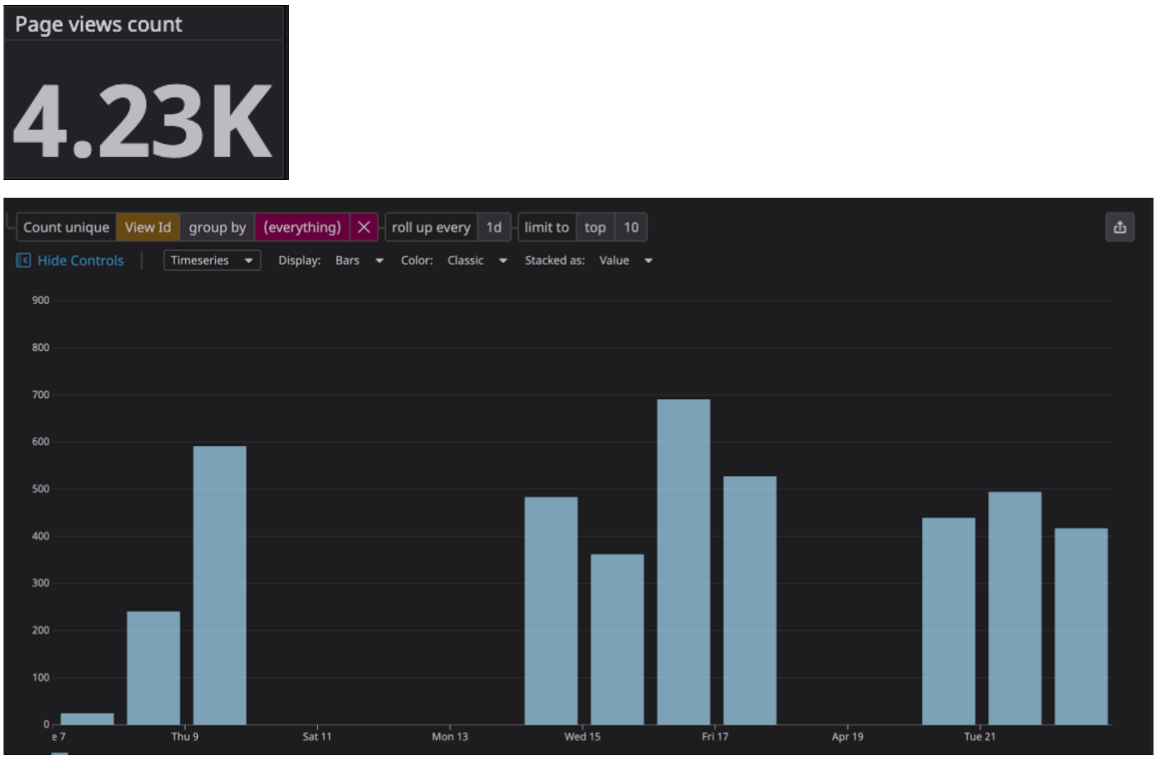
- How often is G-App being used daily?
    - *Page views*: a page view represents a user visit on a page, so from the number of page views, the development team can know how much traffic G-App gets every day.
- What do users do on G-App? What difficulties do users encounter?
    - *Session*: A session includes all the pages that user visits in chronological order, so it helps to visualize what users do on G-App.
    - *Load time*: The total time it takes for a page to load entirely, including all of its resources (images, scripts, CSS). Slow load time reduces the effectiveness of the website, and as a result, its usability.
    - *First contentful paint (FCP)*: This metric represents the time from navigation to the time when the browser renders the first bit of content from DOM. This is an important metric for monitoring a website's performance as it provides feedback for users if the page is actually loading (Google, 2020). Slow FCP is considered a difficulty for users when using G-App.
    - *Long task*: Task that takes more than 50ms to complete is considered long. Even though not all long tasks are caused by the front-end of the application (because the problem may lie in the network or backend), it is worth monitoring the frequency and know the causes for better support and maintenance.
- What kind of errors do users get when using G-App? Possible ways to log these errors for tracking purposes?
    - *Error*: Errors' information associating with each view will help the development team in troubleshooting the issues. Besides, the total number of errors and summary about error kinds help to provide the general view on the website performance.
- How many API calls are made from G-App?
    - *Resource – XHR*: Resources fetched in each view is recorded. In order to know API requests and their responses, XHR data will be collected. XHR means XMLHttpRequest, which is an API in the form of an object, used to

request data from servers. XHR can be used to retrieve any type of data (such as plain text or JSON), not only just XML.

4.3 Data analysis and summary

The data are taken from 9.00 AM 07 April 2020 to 5.00 PM 22 April 2020. During this period, there were public holidays (Easter) from 10 April to 13 April.

4.3.1 Pageviews


Page views count

4.23K

Count unique View Id group by (everything) roll up every 1d limit to top 10

Hide Controls | Timeseries | Display: Bars | Color: Classic | Stacked as: Value

Figure 7. Total number of page views and page view distribution every day

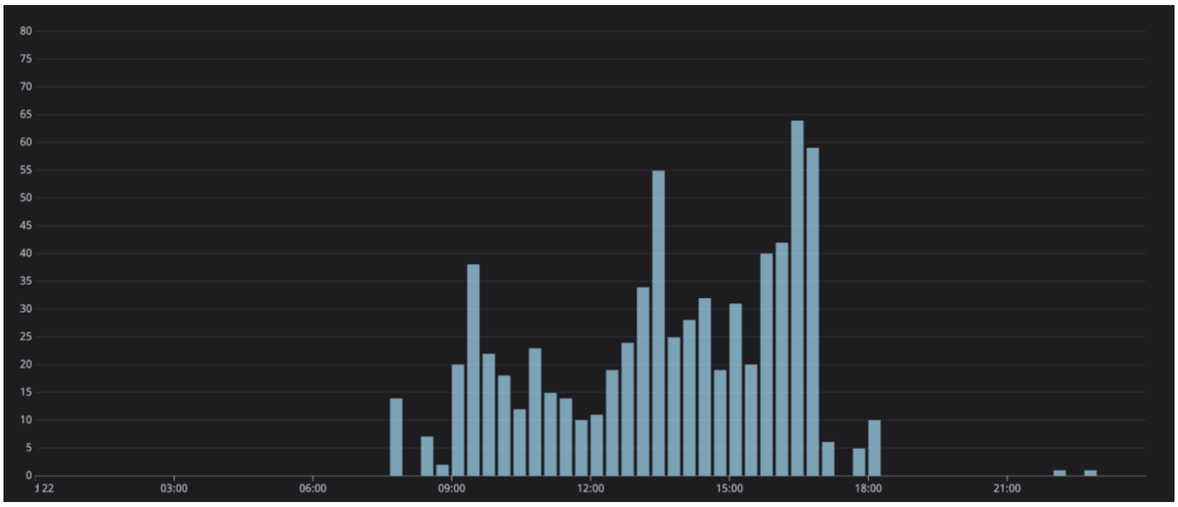
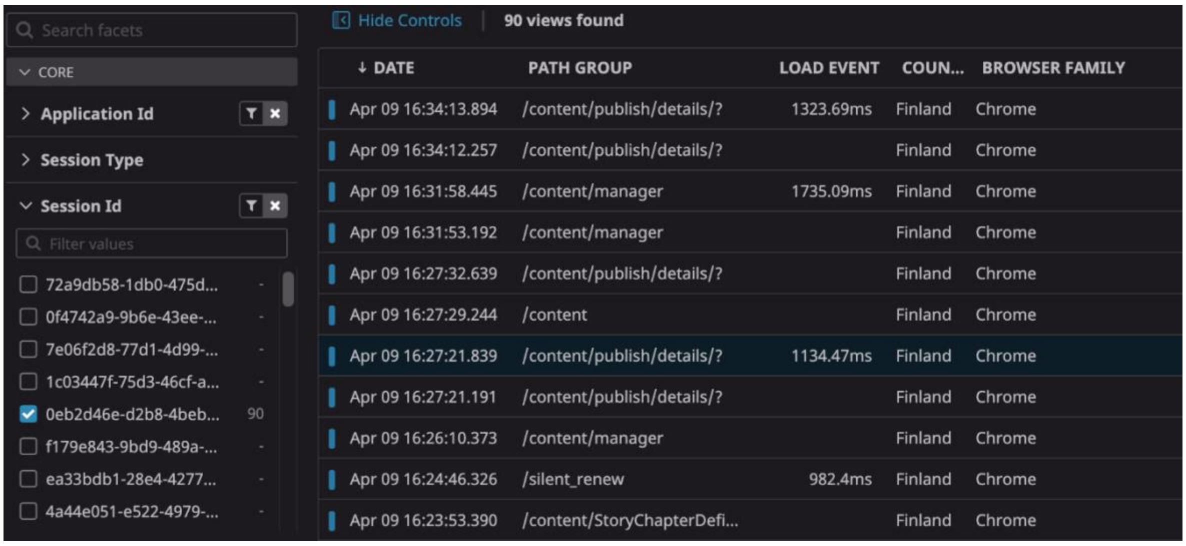


Figure 8. Number of page views distribution on a specific day (22 April 2020)

The total page views of G-App during the monitoring period is approximately 4230 page views. The total page views of each day are presented as bar graph in a Figure 7. The vertical axis shows the count of page views, while the horizontal axis represents the timeline. The majority of page views per day falls between 400 to 600 views, the highest page views recorded was on 16 April, which was almost 700 views. However, there were exceptionally low traffic days, such as on 07 April, when traffic was about below 50 page views. There was absolutely no traffic from 10 April to 13 April, 18 April and 19 April.

Figure 8 is an example of the number of page views distribution on a day (22 April). Most of the data was collected from around 8 AM to 7 PM. There was little to no data during the other time of the day.

4.3.2 Session

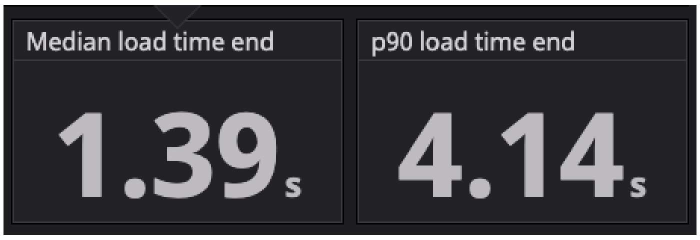


↓ DATE	PATH GROUP	LOAD EVENT	COUN...	BROWSER FAMILY
Apr 09 16:34:13.894	/content/publish/details/?	1323.69ms	Finland	Chrome
Apr 09 16:34:12.257	/content/publish/details/?		Finland	Chrome
Apr 09 16:31:58.445	/content/manager	1735.09ms	Finland	Chrome
Apr 09 16:31:53.192	/content/manager		Finland	Chrome
Apr 09 16:27:32.639	/content/publish/details/?		Finland	Chrome
Apr 09 16:27:29.244	/content		Finland	Chrome
Apr 09 16:27:21.839	/content/publish/details/?	1134.47ms	Finland	Chrome
Apr 09 16:27:21.191	/content/publish/details/?		Finland	Chrome
Apr 09 16:26:10.373	/content/manager		Finland	Chrome
Apr 09 16:24:46.326	/silent_renew	982.4ms	Finland	Chrome
Apr 09 16:23:53.390	/content/StoryChapterDefi...		Finland	Chrome

Figure 9. Details about a journey of user on G-App

Figure 9 shows the detail of a particular session. In the left column, there is a list of session Ids, from which a session can be filtered for detailed examination. Next to a session Id is the number of page views in that session. The details of that session, including every page the user visited, are presented as a list on the right. For example, a session in Figure 9 consists of 90 page views. The last page this user viewed is on the path “/content/publish/details/?” on 09 April 16:34. The details of this view, including errors, resources, long tasks that happened here, can be retrieved by clicking on the row (part 4.3.6 and part 4.3.7).

4.3.3 Load time


Median load time end

1.39s

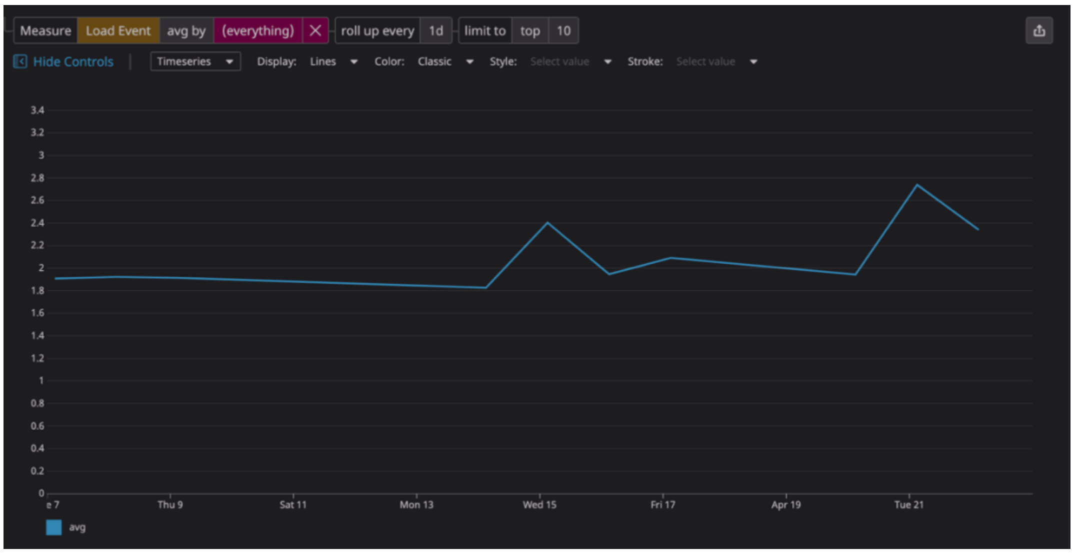
p90 load time end

4.14s

Figure 10. Median load time and p90 load time

Median (p50) and p90 are short for percentiles 50 and percentiles 90. Percentiles mean the distribution of input data. Percentiles show how many values in the set are smaller than the value of that percentile (Ligus, 2012).

During the monitoring period, the median load time (p50) of G-App was recorded at 1.39 seconds (Figure 10). This means 50% of the total load time cases are under 1.39 seconds. The p90 load time of G-App was 4.14 seconds. It is understood that 90% of the total load time cases are faster than 4.14 seconds, or in other words, the slowest load time (10% of total load time cases) was longer than 4.14 seconds.

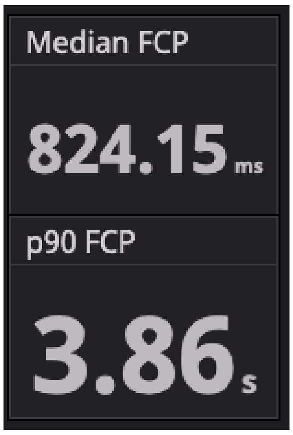

Measure Load Event avg by (everything) roll up every 1d limit to top 10

Display: Lines Color: Classic Style: Select value Stroke: Select value

Figure 11. Average load time daily

The graph in Figure 11 illustrates the average load time of G-App every day during the monitoring period. The average load time was quite stable in the first week (07 April to 14 April) from 1.8 seconds to 2 seconds. There was an increase in average load time on 15 April. From 15 April, the average load time was relatively steady but still higher than the previous week. The highest average load time was recorded on 21 April at 2.8 seconds.

4.3.4 First contentful paint


Median FCP

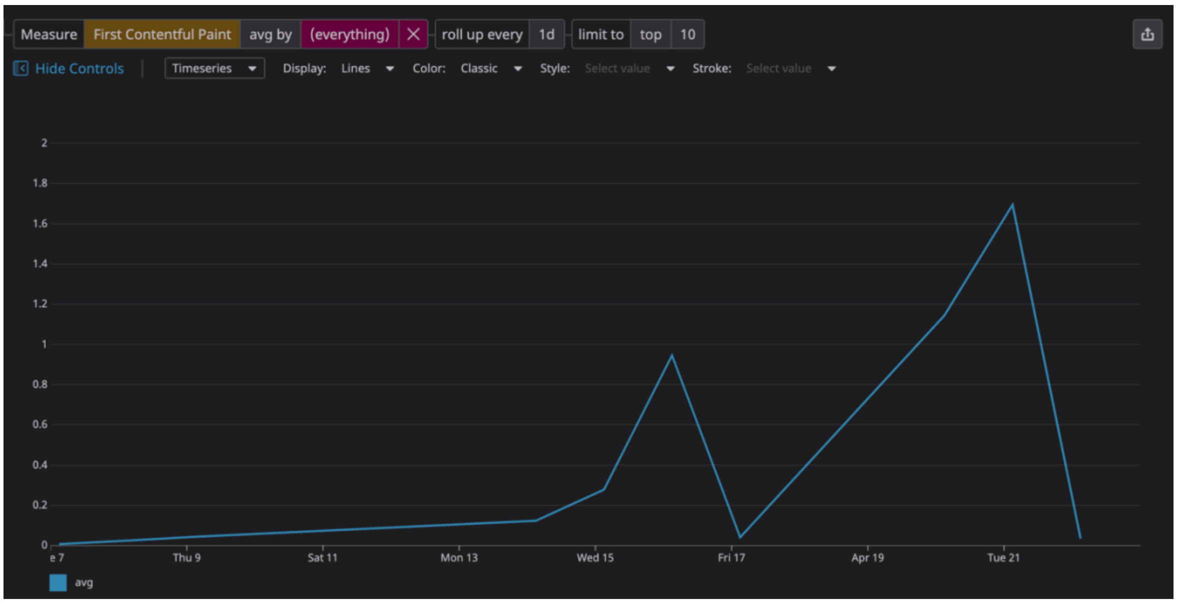
824.15 ms

p90 FCP

3.86 s

Figure 12. Median FCP and p90 FCP

Median FCP was 824.14 milliseconds, which means 50% of first content was rendered within 824.14 milliseconds. The slowest time (10% of the collected data) of FCP was longer than 3.86 seconds, as the p90 FCP metric indicated.

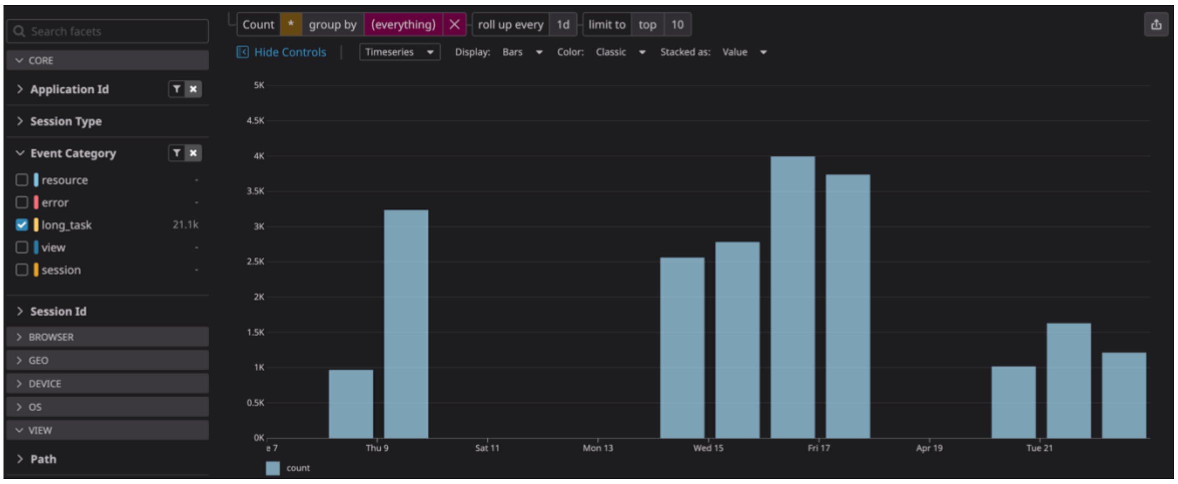

Measure First Contentful Paint avg by (everything) roll up every 1d limit to top 10

Display: Lines Color: Classic Style: Select value Stroke: Select value

Figure 13. Average FCP daily

Figure 13 shows the average FCP time every day during the monitoring period. Average FCP fluctuated from time to time. The longest average time recorded was almost 1.8 seconds; the fastest rendering was approximately 0.2 seconds, which was on 14 April.

4.3.5 Long tasks


Count * group by (everything) roll up every 1d limit to top 10

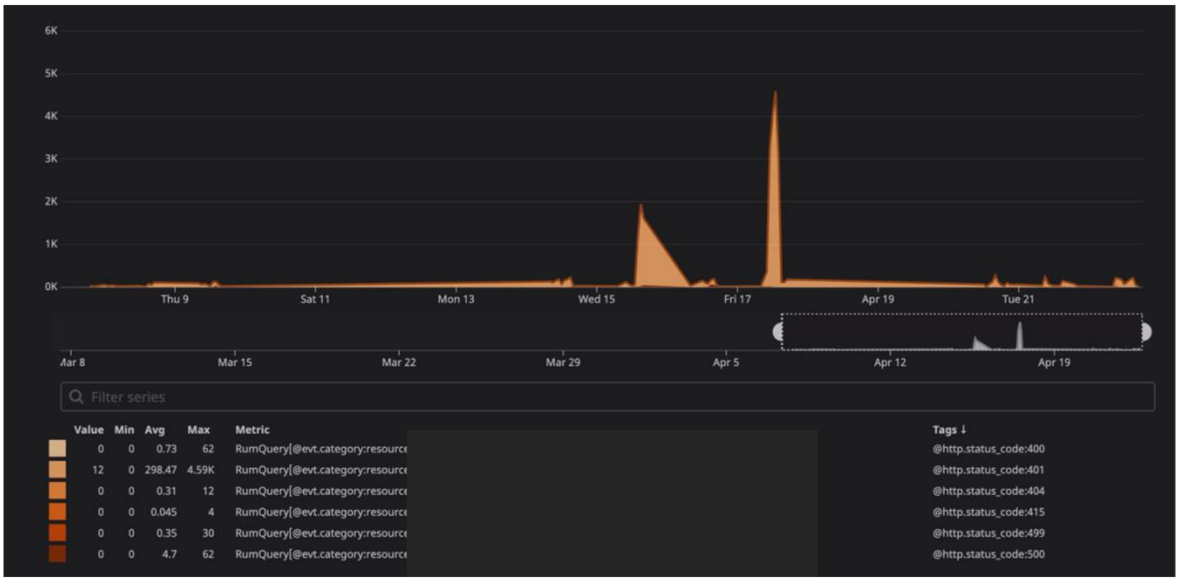
Timeseries Display: Bars Color: Classic Stacked as: Value

resource	-
error	-
✓ long_task	21.1k
view	-
session	-

Figure 14. Number of long tasks daily

Long tasks are tasks taking more than 50ms to complete. The total number of long tasks is pointed out on the left column with the value of 21.100 tasks (Figure 14). The bar graph on the right shows long task distribution daily. On 09 April and the week after (14 April to 18 April), there was a significant number of long tasks, which ranged between 2.500 tasks and 4.000 tasks. In the last week of monitoring (20 April to 22 April), long task count was around 1.600 or less.

4.3.6 Errors



	Value	Min	Avg	Max	Metric	Tags ↓
🟧	0	0	0.73	62	RumQuery[@evt.category:resource	@http.status_code:400
🟧	12	0	298.47	4.59K	RumQuery[@evt.category:resource	@http.status_code:401
🟧	0	0	0.31	12	RumQuery[@evt.category:resource	@http.status_code:404
🟧	0	0	0.045	4	RumQuery[@evt.category:resource	@http.status_code:415
🟧	0	0	0.35	30	RumQuery[@evt.category:resource	@http.status_code:499
🟫	0	0	4.7	62	RumQuery[@evt.category:resource	@http.status_code:500

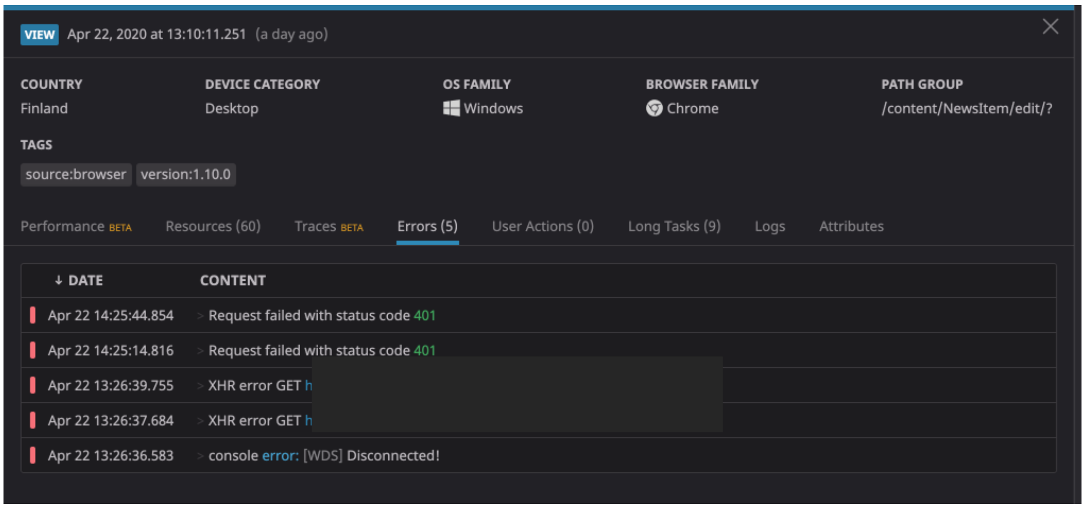
Figure 15. Error count grouped by HTTP status code

The graph in Figure 15 is about the number of errors that happened in G-App during this period. The summary under the graph shows that errors were grouped into five groups by their HTTP response code:

- *400 – Bad request*: The server could not understand the request because of invalid syntax.
- *401 – Unauthorized*: The client is not authenticated to get the response of the request from the server. The client must authenticate themselves first.
- *404 – Not found*: The server cannot find the requested resources. This means the URL is not recognized, or the endpoint is valid but the resource does not exist.
- *415 – Unsupported media type*: The media type requested by the client is not supported by the server, so the server rejects the request.
- *499 – Client close request*: The client closes the connection while the request is being processed.
- *500 – Internal server error*: The server encountered a situation where it does not know how to handle.

  (MDN web docs, 2020)

The errors of G-App occurred from the client side for the most part (status code starts with number 4). The error that happened the most is 401, which was at the highest of 4.590 times on the late 17 April, and at 298 times averagely every day. The frequency of 401 is significantly higher than other types of error. The highest frequency of other errors was only 62 times, which were of status code 400 and 500.



VIEW Apr 22, 2020 at 13:10:11.251 (a day ago)

COUNTRY	DEVICE CATEGORY	OS FAMILY	BROWSER FAMILY	PATH GROUP
Finland	Desktop	Windows	Chrome	/content/NewsItem/edit/?

TAGS
source:browser version:1.10.0

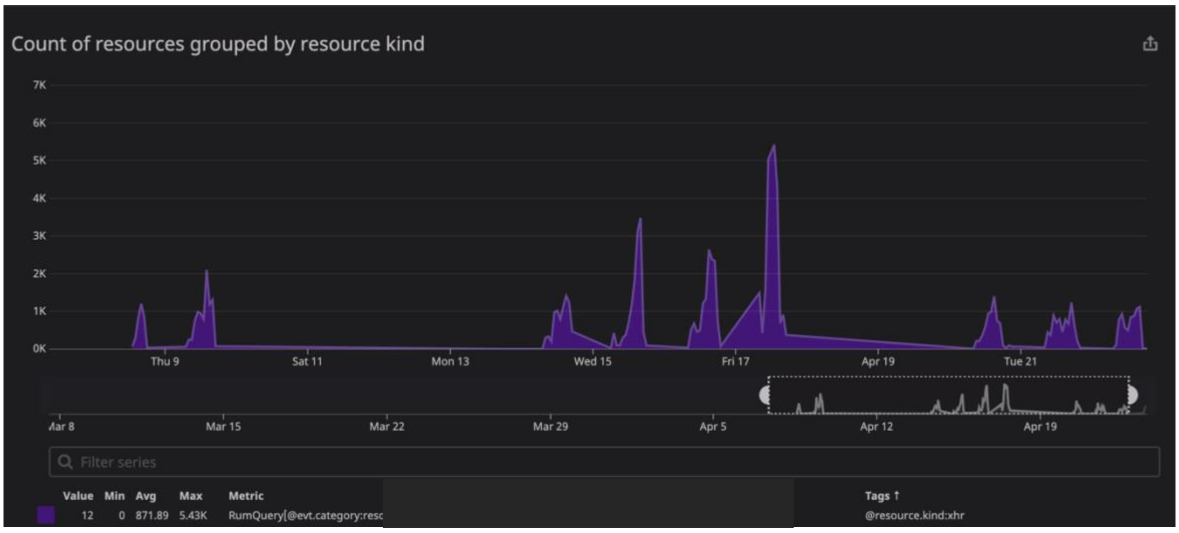
Performance BETA | Resources (60) | Traces BETA | Errors (5) | User Actions (0) | Long Tasks (9) | Logs | Attributes

↓ DATE	CONTENT
Apr 22 14:25:44.854	Request failed with status code 401
Apr 22 14:25:14.816	Request failed with status code 401
Apr 22 13:26:39.755	XHR error GET h[redacted]
Apr 22 13:26:37.684	XHR error GET h[redacted]
Apr 22 13:26:36.583	console error: [WDS] Disconnected!

Figure 16. Errors of a specific page view

Figure 16 is a detail of a particular view. From the top, the time this page was viewed is shown. Below is the information related to this view. Most importantly, the path group is displayed at the end of the row. Errors that arose in this view are listed below. For example, from Figure 16, in the path “/content/NewsItem/edit/?”, there were 5 errors in total, 2 of them were errors with status code 401.

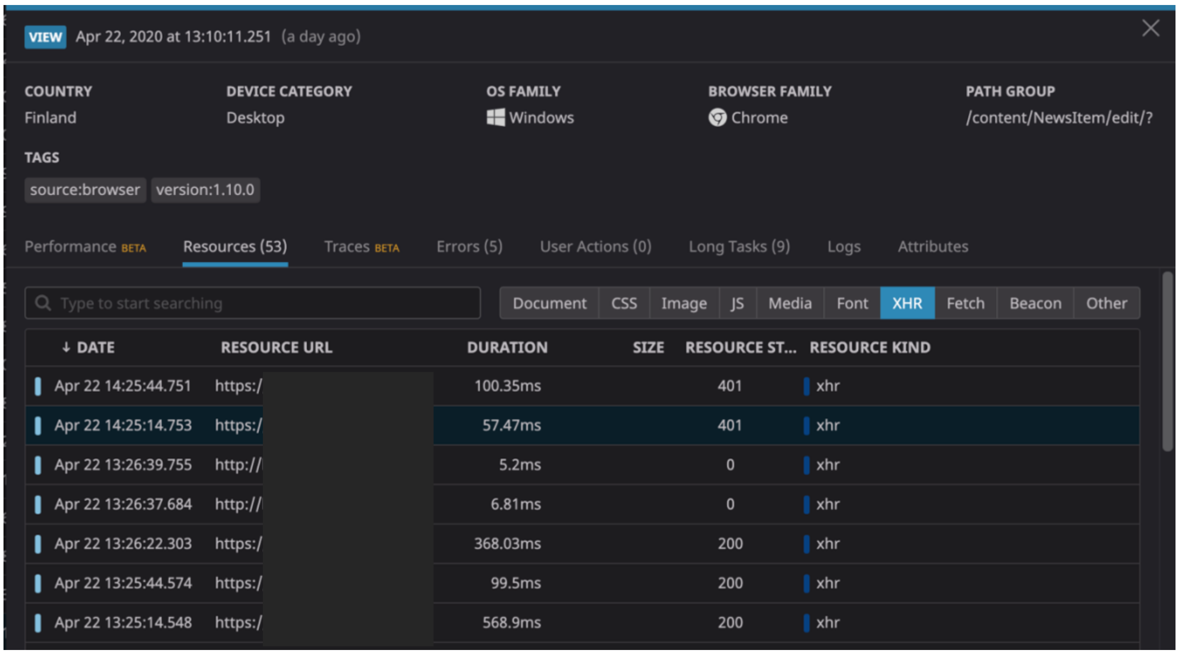
4.3.7 Resource – XHR



	Value	Min	Avg	Max	Metric	Tags ↑
🟪	12	0	871.89	5.43K	RumQuery[@evt.category:resc [redacted]	@resource.kind:xhr

Figure 17. Number of XHR resource daily

Figure 17 demonstrates the number of XHR calls daily. The highest number of XHR calls was on 17 April. In the week from April 14 to 18 April, the number of XHR resources was remarkably higher compared to the other days.



VIEW Apr 22, 2020 at 13:10:11.251 (a day ago)

COUNTRY	DEVICE CATEGORY	OS FAMILY	BROWSER FAMILY	PATH GROUP
Finland	Desktop	Windows	Chrome	/content/NewsItem/edit/?

TAGS

source:browser version:1.10.0

↓ DATE	RESOURCE URL	DURATION	SIZE	RESOURCE ST...	RESOURCE KIND
Apr 22 14:25:44.751	https:/ [redacted]	100.35ms		401	xhr
Apr 22 14:25:14.753	https:/ [redacted]	57.47ms		401	xhr
Apr 22 13:26:39.755	http:// [redacted]	5.2ms		0	xhr
Apr 22 13:26:37.684	http:// [redacted]	6.81ms		0	xhr
Apr 22 13:26:22.303	https:/ [redacted]	368.03ms		200	xhr
Apr 22 13:25:44.574	https:/ [redacted]	99.5ms		200	xhr
Apr 22 13:25:14.548	https:/ [redacted]	568.9ms		200	xhr

Figure 18. XHR resources from a specific page view

Figure 18 is also a detail of a particular view (like Figure 16). The general information about this view is the same in Figure 16. However, Figure 18 focuses on checking the resources of this view. The total number of resources of this view is 53, including all types of resources (such as document, CSS, Javascript). XHR resource can be selected to view separately. In Figure 18, a list of XHR calls is presented with the date, resources URL, duration of the request, and also the resource response status code. It is clear that the first two XHR responses were errors with 401 code (unauthorized), while at the end of the list, there are some successful requests (response code 200).

4.4 Writing documentation

Documentation (Appendix 2) is written in order to share the knowledge and the work with other colleagues. The documentation is not about what the author found out about G-App's performance during the time conducting this research, but rather a general guide for using RUM tool. The documentation aims to help colleagues to retrieve needed data, which support troubleshooting and monitoring purposes in the future. RUM's setup is introduced thoroughly by Datadog, so it will not be included in the documentation.

The documentation starts with a brief introduction about RUM applications and dashboards of each environment of G-App. Next, the documentation helps readers get familiar with all the possible metrics collected by RUM and their attributes. Then it goes to the details of how to retrieve the required data.

5 Discussion

The main purpose of this chapter is to discuss about the monitoring results of G-App in chapter 4: what can be learnt about G-App’s performance from those collected metrics and weather Datadog RUM is a helpful, suitable tool to the company’s purposes.

During the monitoring period for this project, the data was missing from 10 to 13 April (Easter holidays), and on 18, 19 April (weekend) (Figure 7). Moreover, data was mostly recorded from 8 AM to 8 PM every weekday (Figure 8), rarely any data was collected during another time of the day. The reason behind those days and time of missing data is because G-App is an internal web application. G-App is only used when employees work. Besides, RUM tool only gathers data when there are actual users using the application, so data from G-App will only be available in working days on working hours.

Datadog RUM tool, just like Google Analytics, uses cookies to organize user activities into sessions. The difference between the two tools is how they count one session. As for Datadog RUM, a session expires after 15 minutes of inactivity or after 4 hours. On the other hand, a session of Google Analytics expires after 30 minutes of inactivity or at midnight. As Google Analytics aims to support marketing on the website of businesses, they also count sessions based on campaigns. If a user comes to the website from two different campaigns, Google Analytics records two different sessions (Google, 2020).

Comparing to Google Analytics, the session list of Datadog (Figure 9) is more dynamic, as it is possible to browse details of different data (such as errors, resources, long tasks) on each view in a session. This feature would be beneficial in tracking and troubleshooting problems. For example, the development team can check what XHR requests were sent in a specific view and their response codes, or what errors belong to those requests. With Google Analytics, a session list just simply includes a series of different URL paths that users visited with general information about their device or visited time; no technical data is provided.

A research conducted by Brian Dean’s team at Backlinko in 2019 found out that the average load time for desktop applications is 10.3 seconds (Dean, 2019). That was the result from analyzing 5 million pages. However, that number is still much slower than the load time that Google suggests. In their RAIL model for web development, Google recommends that web pages should deliver content and become interactive in under 5 seconds (Google, 2020). Comparing the average load time of G-App with this standard, G-App performs efficiently at averagely 1.39 seconds (Figure 10). Apparently, even percentile 90 load time of G-App (4.14 seconds, Figure 10) is still faster than the speed recommended by Google.

Philip Walton proposed on Google Developers site (2020) that a good First Contentful Paint score is within 1 second at percentile 75. At percentile 50, G-App's FCP time is 0.8 seconds (Figure 12), which is faster than the standard, but the percentile 90 FCP time is fairly slow (at 3.86 seconds, Figure 12). However, as the recommended FCP time is for percentile 75, it is hard to make a comparison here. If the development team wants to keep the FCP time of G-App up with the recommendation by Philip Walton, the percentile 75 FCP time should be calculated first. Even though Datadog RUM dashboard does not provide a specific number like in Figure 12 for percentile 75 of FCP, it produces a graph of percentile 75 FCP in a specific period. That graph can be useful in tracking weather G-App's percentile 75 FCP time meets the recommendation.

The result in part 4.3.5 shows that there is an excessively high number of long tasks that happened in G-App, averagely 2100 tasks per working day. With tasks that need more than 50ms to complete, they make the users experience perceptible delays. Moreover, the connection between action and reaction of users may be broken because of waiting for a long time (Google, 2020). The primary cause of long tasks is large scripts (Osmani, 2019), but there can be many reasons such as slow responses from a server, network connection, etc. In this situation, it really depends on how the development team perceives those long tasks. If they think Google's recommendation on task duration (within 50ms) is suitable for an internal web application like G-App, a thorough investigation about causes of these long tasks should be conducted in order to make improvements. Datadog RUM tool supports in finding out the duration of the tasks and in which paths of G-App did those tasks happen.

There are some anomalies in the error graph of G-App (Figure 15). An unusually high number of errors happened in two consecutive days (16 and 17 April), and the majority of them is with 401 response code – unauthorized error. Looking at Figure 16 – a graph of XHR resources called from G-App, there are also some anomalies in 16 and 17 April. It seems like there was a spam of resource requests to the server, and they are rejected because of unauthorized users. The reasons behind those anomalies are not discussed in this research, as they could be related to other components of the system which are under company's confidentiality. However, the importance is from the monitoring results, Datadog RUM tool proves to be useful in detecting anomalous behaviors of the application. It helps the development team be aware of the issues. Moreover, the Datadog's detailed view of errors (Figure 16) can be beneficial to the development team in troubleshooting those problems. As the tool helps to identify errors immediately when they happen, it is expected to reduce the cost of fixing those errors for the company.

Overall, from the results, Datadog RUM tool shows that it suits the company’s purposes. It is able to provide necessary data about the application’s performance. It can also capture the behaviors of users or any errors in the application. Most importantly, Datadog RUM tool shows detailed data, which is valuable for tracing and solving problems. Utilizing Datadog RUM for G-App is undoubtedly advantageous for the work of G-App’s development team.

6 Conclusion

This thesis was started from the need of a monitoring tool for a web application of the company, known as G-App. The objective of this thesis is implementing a monitoring tool in order to find out about usability of G-App and observe its daily performance. The mentioned tool should at least be able to provide data about G-App's daily use, API calls, errors or any possible difficulties users may encounter.

A search about suitable monitoring tool was conducted. Google Analytics was the first approach. However, as Google Analytics aims to support marketing purposes, it is unfitted for an internal web application like G-App. Later, Datadog RUM tool was chosen as it offers ideal metrics, and in addition, other Datadog products are already used by the company. After decision was made, the tool was configured in G-App. Because of the time limit of this thesis, the data was taken from a 2-week monitoring period.

The recorded data was analyzed by descriptive analysis method. The results are able to reflect G-App's performance through metrics like load time, First Contentful Pain time, page views, XHR resources or errors. The difficulty users may experience was indicated by long tasks data. During the monitoring period, G-App shows good performance with fast load time and fast FCP time. Datadog RUM tool was also successful in pointing out anomalies of the application. A high number of XHR calls and errors with response code 401 in two consecutive days was discovered.

Finally, the work of implementing a monitoring tool for G-App is completed. The next phase depends on the development team to utilize the tool to support their daily work.

References

Aggarwal, A. & Sayer, M. & Reddy, S. & Gourly, D. & Totty, B. 2002. HTTP: The definitive guide. Published by O’Reilly Media, Inc.

Bevan, N. 2005. Cost benefits evidence and case studies. URL: https://www.semanticscholar.org/paper/Cost-benefits-evidence-and-case-studies-Bevan/897b02057eb3778da0350bf5ad20be341da2977a. Accessed: 13 April 2020.

Catchpoint, 2018. A primer on real user monitoring – Monitoring what matters (ebook).

Clay, K. 2013. Amazon.com goes down, loses $66,240 per minute. URL: https://www.forbes.com/sites/kellyclay/2013/08/19/amazon-com-goes-down-loses-66240-per-minute/ - 690d6546495c. Accessed: 16 April 2020.

Croll, A. & Power, S. 2009. Complete web monitoring. Published by O’Reilly Media, Inc.

Datadog documentation, 2020. Real user monitoring. URL: https://docs.datadoghq.com/real_user_monitoring/. Accessed: 09 April 2020.

Desikan, S. & Ramesh, G. 2006. Software testing: principles and practices. Published by Dorling Kindersley Pvt. Ltd.

Dixon, J. 2017. Monitoring with Graphite. Published by O’Reilly Media, Inc.

Google, 2020. First contentful paint. URL: https://developers.google.com/web/tools/lighthouse/audits/first-contentful-paint. Accessed: 22 April 2020.

Google, 2020. How a web session is defined in Google Analytics. URL: https://support.google.com/analytics/answer/2731565. Accessed: 04 May 2020.

Google, 2020. Measure performance with the RAIL model. URL: https://developers.google.com/web/fundamentals/performance/rail. Accesed: 05 May 2020.

Google, 2020. Set up the Analytics tag. URL: https://support.google.com/analytics/answer/1008080?hl=en&ref_topic=1008079. Accessed: 28 April 2020.

Google, 2020. Tracking code overview. URL: https://developers.google.com/analytics/resources/concepts/gaConceptsTrackingOvervie w. Accessed: 28 April 2020.

Graham, D. & Veenendaal, E. & Evans, I. & Black, R. 2008. Foundations of software testing, ISTQB certification, revised edition. Published by Cengage Learning EMEA.

Interaction Design Foundation, 2020. Literature on usability. URL: https://www.interaction-design.org/literature/topics/usability. Accessed: 13 April 2020.

International Organization for Standardization, 2018. Part 11: Usability: Definitions and concepts. URL: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en. Accessed: 14 April 2020.

Julian, M. 2017. Practical monitoring. Published by O'Reilly Media, Inc.

Kahate, A. 2004. Introduction to database management systems. Published by Pearson India.
Kaner, C. & Falk, J. & Nguyen, H. Q. 1999. Testing computer software, 2nd edition. Published by Wiley Publishing, Inc.

Kristol, D. M. 2001. HTTP cookies: Standards, Privacy, and Politics. Bell Labs, Lucent Technologies.

Lentz, D. M. 2019. Introducing Datadog real user monitoring. URL: https://www.datadoghq.com/blog/real-user-monitoring-with-datadog/. Accessed: 09 April 2020.

Lever, C. 2016. Synthetic, RUM and a brief history of APM. URL: https://rigor.com/blog/synthetic-rum-history-of-apm. Accessed: 03 April 2020.

Lewis, W. E. 2017. Software testing and continuous quality improvement, 3rd edtion. Published by Auerbach Publications.

Ligus, S. 2012. Effective monitoring and alerting. Published by O’Reilly Media, Inc.

Lucca, G. A. D. & Fasolino, A. R. 2006. Testing web-based application: the state of art and future trends. URL: https://www.researchgate.net/profile/Giuseppe_Di_Lucca/publication/4174551_Testing_Web-based_applications_the_state_of_the_art_and_future_trends/links/00b4952305b93d334e000000.pdf. Accessed: 15 April 2020.

Martin, J. & McClure, C. L. 1983. Software maintenance: The problem and its solutions. Published by Prentice-Hall.

Mastin, P. 2016. Real user measurements. Published by O’Reilly Media, Inc.

MDN web docs, 2020. HTTP response status code. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status. Accessed: 23 April 2020.

Moharana, S. S 2013. Analysis of load balancers in Cloud computing. URL: https://www.researchgate.net/publication/269302960_Load_Balancer_as_a_Service_in_Cloud_Computing. Accessed: 17 April 2020

Molyneaux, I. 2014. The art of application performance testing, 2nd edition. Published by O’Reilly Media, Inc.

Moore, F. 2013. Routers (Network routing). URL: https://www.researchgate.net/publication/317057644_Routers_Network_Routing. Accessed: 17 April 2020.

Nielsen, J. 1993. Usability engineering. Published by Academic Press.

Nielsen, J. 2007. Intranet usability shows huge advances. URL: https://www.nngroup.com/articles/intranet-usability-huge-advances/. Accessed: 14 April 2020

Nielsen, J. 2020. Intranet users stuck at low productivity. URL: https://www.nngroup.com/articles/intranet-users-stuck-low-productivity/. Accessed: 14 April 2020.

Osmani, A. 2019. Are long JavaScript tasks delaying your Time to Interactive? URL: https://web.dev/long-tasks-devtools/. Accessed: 06 May 2020.

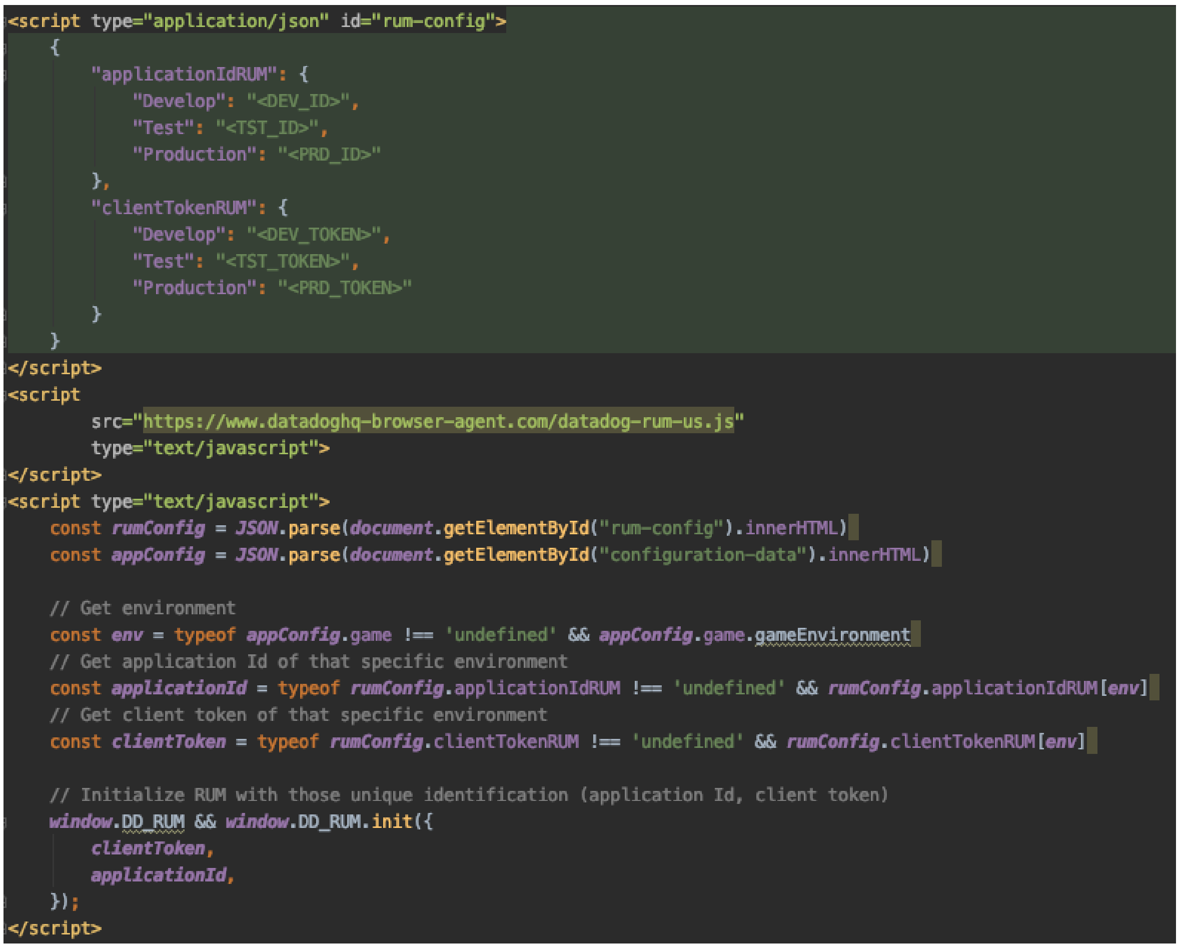
Rubin, J. & Chisnell, D. 2008. Handbook of usability testing, 2nd edition. Published by Wiley Publising, Inc.

Steen, M. & Tanenbaum, A. S. 2017. Distributed systems, 3rd edition. Published by Pearson Education, Inc.

Tirtea, R. & Castelluccia, C. & Ikonomou, D. 2011. Bittersweet cookies: Some security and privacy considerations. Copyright by the European Union Agency for Security. URL: https://www.enisa.europa.eu/publications/copy_of_cookies. Accessed: 27 April 2020.

Appendices

Appendix 1. RUM setup code



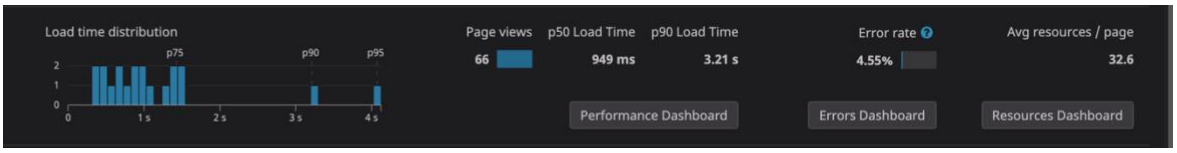
```
<script type="application/json" id="rum-config">
    {
        "applicationIdRUM": {
            "Develop": "<DEV_ID>",
            "Test": "<TST_ID>",
            "Production": "<PRD_ID>"
        },
        "clientTokenRUM": {
            "Develop": "<DEV_TOKEN>",
            "Test": "<TST_TOKEN>",
            "Production": "<PRD_TOKEN>"
        }
    }
</script>
<script
        src="https://www.datadoghq-browser-agent.com/datadog-rum-us.js"
        type="text/javascript">
</script>
<script type="text/javascript">
    const rumConfig = JSON.parse(document.getElementById("rum-config").innerHTML)
    const appConfig = JSON.parse(document.getElementById("configuration-data").innerHTML)

    // Get environment
    const env = typeof appConfig.game !== 'undefined' && appConfig.game.gameEnvironment
    // Get application Id of that specific environment
    const applicationId = typeof rumConfig.applicationIdRUM !== 'undefined' && rumConfig.applicationIdRUM[env]
    // Get client token of that specific environment
    const clientToken = typeof rumConfig.clientTokenRUM !== 'undefined' && rumConfig.clientTokenRUM[env]

    // Initialize RUM with those unique identification (application Id, client token)
    window.DD_RUM && window.DD_RUM.init({
        clientToken,
        applicationId,
    });
</script>
```

Appendix 2. Documentation

G-App’s RUM

Datadog Real User Monitoring (RUM) tool is used to monitor G-App’s performance and availability. Development, test and production environments are monitored separately. All RUM applications are listed here: <*LINK*>

There are 3 dashboards created for each environment: performance dashboard, errors dashboard and resources dashboard.



More information about each dashboard can be found at:
https://docs.datadoghq.com/real_user_monitoring/dashboards/

Metrics

Collected metrics are:

- *View*: a user’s visit on the application
- *Session*: collection of views
- *Resource*: a resource event (images, XHR/Fetch, CSS or JS scripts) and its information (name, loading duration, etc.)
- *Long task*: Task that blocks main thread for more than 50ms
- *Error*: errors emitted by the browsers

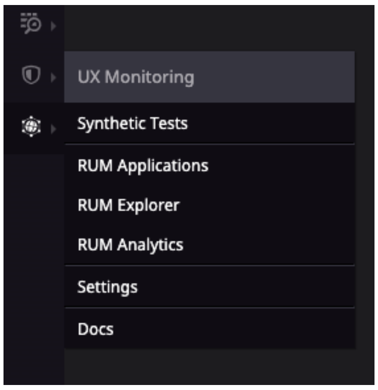
Every metric comes with many of its attributes (such as duration, URL, status code, etc.). More details about metrics and their attributes can be found here:
https://docs.datadoghq.com/real_user_monitoring/data_collected/

Retrieving data

<u>Tools:</u>

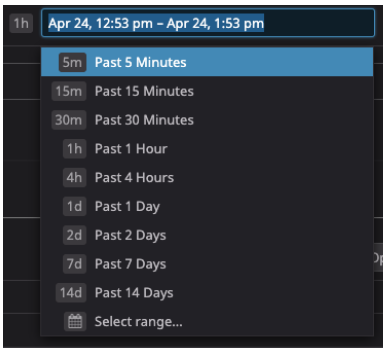
- *RUM Explorer* allows to explore all views and their related telemetries
- *RUM Analytics* display views data aggregation

Both RUM Explorer and RUM Analytics can be found on the left menu of Datadog web app (picture below).

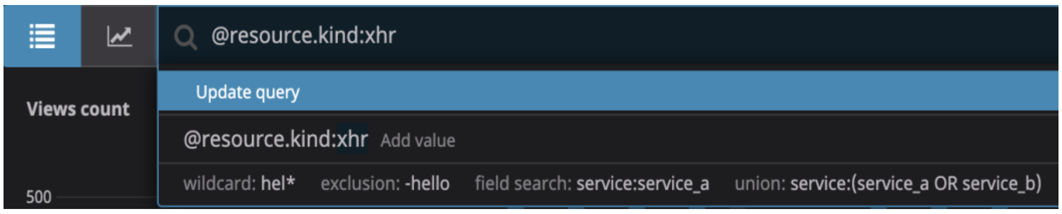


Implementation:

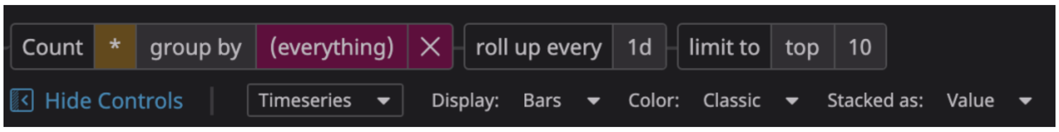
1. Choose a time range



2. Put in the query, choose either Explorer or Analytics view on the top left


@resource.kind:xhr
Views count
Update query
@resource.kind:xhr Add value
500
wildcard: hel*
exclusion: -hello
field search: service:service_a
union: service:(service_a OR service_b)

3. *(Analytics only)* Choose what measure to retrieve, time interval and so on. Detailed data can be shown by clicking on the bar/line of the graph => "View events"



Query instruction:

Queries used in Explorer and Analytics are Datadog's metrics' attribute name. More about those attributes, please check Datadog's docs in the ***"Metrics"*** section above.

Query examples:

- Retrieving all views with errors in development environment
  - ⇨ @application.id:xxx @view.measures.error_count:>0
- Retrieving all errors with code 401 in path group "/content"
  - ⇨ @view.url_details.path_group:\/content @http.status_code:401