FEDERAL UNIVERSITY OF ITAJUBÁ - UNIFEI
MASTERS DEGREE PROGRAM IN
COMPUTER SCIENCE AND TECHNOLOGY

Footstep: An Approach to Track Web Usage in Real Time.

Wesley Goulart Siqueira

Itajubá, 13th March 2019

FEDERAL UNIVERSITY OF ITAJUBÁ - UNIFEI
MASTERS DEGREE PROGRAM IN
COMPUTER SCIENCE AND TECHNOLOGY

Wesley Goulart Siqueira

Footstep: An Approach to Track Web Usage in Real Time.

Dissertation submitted to Masters Degree Program in Computer Science and Technology as part of requirements to obtain the Master (M.Sc.) in Computer Science and Technology title.

Concentration area: Computing Systems

Orientador: Prof. Dr. Laércio Augusto Baldochi Júnior

13th March 2019
Itajubá

FEDERAL UNIVERSITY OF ITAJUBÁ - UNIFEI
MASTERS DEGREE PROGRAM IN
COMPUTER SCIENCE AND TECHNOLOGY

Footstep: An Approach to Track Web Usage in Real Time.

Wesley Goulart Siqueira

Dissertation approved by examining board in February 25, 2019, granting the author the title of **Master (M.Sc.) in Computer Science and Technology.**

Itajubá
2019

Wesley Goulart Siqueira

Footstep: An Approach to Track Web Usage in Real Time

Dissertation submitted to Masters Degree Program in Computer Science and Technology as part of requirements to obtain the Master (M.Sc.) in Computer Science and Technology title.

Prof. Dr. Laércio Augusto Baldochi Júnior
Orientador

Prof. Dr. Ethan Vincent Munson

Prof. Dr. Bruno Guazzelli Batista

Itajubá
13th March 2019

Agradecimentos

Agradeço e dedico este trabalho aos meus pais, Gloria e Venicio, dos quais tive todo amor e apoio durante toda minha vida.

Não menos importante, agradeço e dedico este trabalho também à minha amada esposa, Luciana, pela compreensão, paciência e total apoio nessa jornada. Dedico também ao meu recém chegado e tão amado filho, Diego.

Agradeço também aos meus irmãos, Glauber, Warley e Wallace por completarem o meu alicerce.

Em seguida, agradeço ao meu orientador Prof. Dr. Laércio Augusto Baldochi Júnior, pela paciênca e confiança em mim depositadas.

E a quem mais que possa ter contribuído no desenvolvimento deste trabalho, meus sinceros agradecimentos.

"True knowledge exists in knowing that you know nothing."
(Socrates)

Abstract

Understanding the user behavior is paramount for the success of any website. Existing approaches for understanding the user behavior in web applications exploit server web logs. These logs contain details on how each user browsed, for instance, an online store. Although server logs are useful to provide important insights concerning the user behavior, they do not provide fine grained information about the actions they perform in the user interface. To tackle this problem, this work proposes Footstep, an all-in-one system that provides facilities for collecting, processing and analyzing logs gathered at client-side. Footstep allows tracking users as they browse web pages, collecting the entire DOM-tree information associated to each triggered event. Footstep also provides a graph-based data model which facilitates the extraction of knowledge from collected logs. Finally, an analytics tool is provided to display information regarding pages and elements audience, the navigation flow between pages and elements, as well as conversion rates on pages and elements.

Key-words: User behavior, web analytics, graph theory.

Resumo

Compreender o comportamento do usuário é primordial para o sucesso de um *website*. As abordagens existentes que exploram análise do comportamento dos usuários utilizam os *logs* de servidor. Esses logs possuem detalhes sobre o que cada usuário acessou, como, por exemplo, uma loja *online*. Embora os logs sejam úteis ao fornecer uma boa percepção a respeito do comportamento dos usuários, eles não fornecem informações detalhadas sobre as ações realizadas pelos mesmos nas páginas acessadas. A fim de atacar este problema, este trabalho propõe o Footstep, um sistema completo que fornece coleta, processamento e análise dos eventos disparados pelos usuários, no nível dos elementos das páginas. Footstep fornece rastreamento dos usuários à medida em que navegam pelas páginas, coletando informações sobre a estrutura DOM dos elementos associados a cada evento. Footstep fornece também um modelo de dados baseado em grafos que facilita a extração de informação útil dos eventos coletados. Finalmente, uma ferramenta analítica é fornecida para visualizar as informações a respeito das interações sobre as páginas e elementos, como audiência, fluxos de navegação e taxas de conversão das páginas e elementos.

Palavras-chave: Comportamento do usuário, análise web, teoria dos grafos.

List of Figures

Acronyms

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
COP	Container, Object and Page
CSV	Comma Separated Values
DOM	Document Object Model
ETL	Extract, Transform and Load
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISP	Internet Service Provider
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
OLAP	Online Analytics Processing
OS	Operating System
URL	Uniform Resource Locator
XML	Extensible Markup Language
WUM	Web Usage Mining

Contents

1 Introduction

Over the past years, efforts were made by both industry and academia to understand and analyze the user behavior in web applications. In a highly competitive market, web companies need to understand the behavior of their users in order to improve the user experience. This need has lead to the development of analytics tools, capable of tracking users' events, measure audience, categorize pages, plot heat maps of pages, record the users' navigation, among other features. As different tools provide different features, it is common that companies resort to more than one tool in order to acquire the information they need.

The user experience in websites is important to please and retain visitors. In e-business, it is paramount to provide pleasant and concise navigation and highly desirable serving the right content at the right time for the users, as they can easily move to another online retailer if their necessities are not satisfied (SCHAFER; KONSTAN; RIEDL, 2001). Therefore, the success or failure of a web application is due to its potential to attract and retain visitors. Thus, studies to understand and predict user's behavior in Web applications has become increasingly important (GÜNDÜZ; ÖZSU, 2003).

In today's connected world, e-commerce companies offers the opportunity of browsing endless product catalogs, comparing prices, enjoying deals, creating wish lists, from anywhere, in a fast and easy way. As a result, e-commerce sales have grown much faster than store sales over the past years in almost every product category. In order to improve sales, business analysts try to identify the reasons that motivate customers to purchase, or not, a given product (MOE; FADER, 2004; LIU et al., 2016).

Some of existing analytics tools rely on server logs in order to track the users' actions in websites. Server logs present information regarding the resources accessed by users such as pages, images and other files. Almost all web applications already implement server logging for reasons such as monitoring, health of services, debugging, and so on. On the other hand, server logs present a low level of detail, the identification of the user session is not trivial, and it is very hard to extract useful information about the user experience without any code instrumentation to enrich the logs.

Web logs may also be collected on the client side. Gathered in the user's browser, client logs provide more details about the user interactions, such as clicks, the movement of the cursor, the use of the scroll bar and the keyboard. Moreover, by exploiting client web logs it is possible to associate events to page elements, making it possible to spot where a given interaction occurred, which allows tracking the user more precisely.

The volume of client log data is significantly larger than the volume of server logs.

A large web application with high access rate can easily produce several gigabytes of log data in a matter of few hours. In order to collect and store a large amount of data an OLAP (Online Analytics Processing) system, built upon a fast ETL (Extract, Transform and Load) architecture is required.

Collecting logs at client side can have some drawbacks depending on how the log collection is designed (HERNÁNDEZ et al., 2017). Generally, it requires code injection in web pages and depends on cookies or JavaScript code, leading to privacy concerns. Another drawback could be the overhead in download time of web pages, affecting the user experience or even slowing the users' interactions. Therefore, the log collecting mechanism must be designed to be fast without affecting the user experience.

Either server or client logs have to be translated to something that represents an user interaction in such a way that analysts can extract information about the user behavior. In other words, a data model is required to represent user interaction, containing relevant data to describe the user's actions over web pages. This model must be both expressive enough in order not to lose any valuable information, and simple enough to be easily stored in a database.

1.1 Motivation

Currently, there are several analytics tools available. In general, different analytics tools provide different features for analyzing user behavior. For instance, Google Analitics (2005) provides a highly customizable event tracking system; CrazyEgg (2005) provides heat maps of web pages, and Hotjar (2014) offers facilities for recording the user interaction on web pages.

Analytics tools must be properly configured before install into websites and often require at least a minimum code injection in each web page. Some tools also require more extensive code injection. Google Analytics, for instance, provides a mechanism to push user defined events, called *Custom Dimensions and Metrics*. To use this feature, a specific HTTP server call is needed, which methods require some coding development.

Therefore, analytics tools provide interesting features for tracking user events in web applications. However, using this features requires learning inner details of the available tools. Moreover, as most of these tools process server logs, it is not possible to associate user events to page elements, making it hard to grasp the details of each interaction. Finally, it is worth to notice that most of these tools are paid, and using different analytics tools may be costly, specially for small business.

This scenario shows that efforts to provide facilities for understanding the user behavior in web applications present limitations. This work proposes a system that aims at tackling these limitations.

1.2 Objectives

In order to help understanding the behavior of users in web applications, this work proposes the Footstep system, an automated system grounded by graph theory designed to collect and process events on web page elements' level. The Footstep system is built meeting the requirement of minimum code injection, providing real-time information about web usage flow among pages and elements. To achieve this aim, the following objectives were set out:

1. Define a graph model to represent web usage at web elements' level;
2. Collect events from web pages in a less intrusive way, avoiding code injection and minimizing data consumption. The collection mechanism must provide detailed information about web page elements;
3. Provide a fast and reliable way to consume, process and translate the collected events to the graph model;
4. Provide a feature for tagging pages and elements as user definition;
5. Build a visualization tool to extract and display web usage flow among pages and elements.

Footstep system proposes a different approach to organize and analyze pages and elements, it is built upon graph theory which brings a great capability of relations identification and extraction. At the heart of this approach is a generalization procedure that allows grouping similar pages and elements, what shortens the amount of analyzed data. By exploiting the graph, it is possible to draw flow diagrams between pages, determine which flow converts better, plot pages' heat maps, identify elements with more clicks and conversion rates, and analyze how far the user is from a conversion point. Moreover, it is possible to set conversion points and measure the effectiveness of different navigation flows.

1.3 Methodology

To achieve the aim of this research it was needed to develop new data models to facilitate storing and processing client logs efficiently (objectives 1 and 4), and to build an OLAP system to support the storage, processing and analysis of this data (objectives 2, 3 and 5).

In order to achieve objective 1, a graph data model called UsaGraph (SIQUEIRA; BALDOCHI, 2018) was developed. UsaGraph maps user interaction to four main types of entities to describe an event triggered by user in web pages: *User*, *Event*, *Page* and *Element*. Each entity was mapped to a specific node type in the model, containing each one specific attributes describing the event in detail, such as its event type and trigger date, the page and elements that this event was triggered on, and some attributes which address the required level of detail.

For addressing objective 2, a JavaScript library was developed to listen to events in web pages, collecting all required information to fulfill UsaGraph model needs. This library was designed to be light and small in order not to affect the loading of web pages. This library was also designed to push events in an economic fashion to the event collector systems. Each event is automatically translated to a single HTTP call containing as less bytes as possible without missing any important information.

A structure built upon a messaging broker tool and an asynchronous messaging consumer was developed to accomplish objective 3. All events collected by the JavaScript library are pushed to a server connected directly to the messaging broker queue, which has the ability to receive and dispatch these events to several consumer systems. Therefore, the messaging consumer tool retrieves the events from queues, processes and parses these events to the UsaGraph model and persists them into the graph database.

To attain objective 4 and provide customization to UsaGraph, two types of nodes were defined to enrich the model: *PageLabel* and *ElementLabel*. *PageLabel* is meant to connect similar *Page* nodes by identifying different URLs that match the same pattern. *ElementLabel* connect similar *Element* nodes by its tag and attributes similarity. A document based database was used to hold user defined patterns to create *PageLabel* nodes.

Finally, to achieve objective 5, a web application was developed to provide all features to manage, monitor and visualize web usage flow. Using this application, administrators can get the JavaScript code snippet to install in their pages in order to collect user interactions. They can configure their URLs patterns to cluster pages in a flexible way, monitor the audience on pages and on page elements, plot page heat maps, define conversion points and detect abnormalities in users interactions, for instance, links that pull users away of desired flows.

1.4 Dissertation structure

This dissertation is organized as follows:

Chapter 2 presents the state of art in user behavior analysis, showing the approaches for tracking, collecting and interpreting web usage. This chapter also presents the efforts to detect characteristics that might give insights about the behavior of web users.

Chapter 3 describes how UsaGraph was designed, the purpose of each node type and how they are connected, and, finally, what information can be extracted from this graph model in order to get the desired information about user behavior.

The architecture of the Footstep system is presented in Chapter 4, detailing how the specific objectives 3 and 4 were achieved, in other words, how events triggered by the JavaScript library are consumed, parsed and stored as UsaGraph nodes.

Chapter 5 shows the experiments, presenting two different kinds of web systems taking advantage of Footstep to extract useful information about their users' interactions, providing insights about design of pages and how elements can affect interactions.

Finally, Chapter 6 presents final remarks highlighting the contributions of this work. This chapter also discuss future work.

2 User Behavior on Web Applications

2.1 Initial considerations

Product designers always develop new products thinking of their usefulness. All starts with the product design, when interaction and marketing issues are considered. An important part of product design is the measure of its usability and often the product is tested several times by a small groups of people in order to make sure that the product release is feasible.

Product designers usually expect certain behavior of its consumers, but it is very common that expectation and reality not always go together. Therefore, it is very important to know the actual usage of the product after its release, if all is going as designed and expected or if something unexpected is coming across. This is also known as user experience, which can be measured by analyzing the user behavior. Thus, user behavior analysis is important in order to assess the effectiveness of a product or service (BERNASCHINA et al., 2017).

For web applications, specially in e-business, user behavior analysis is of paramount importance. Retailers always want to increase their sales by making simple or significant changes in their stores, and the same is true for e-commerce websites. Changes in the design of user interfaces might result in a more pleasant user experience, increasing sales. But the opposite may also be true: wrong decisions in web page design may affect sales in a negative way.

In order to analyze the behavior of users on the Web, it is important to track them, collecting their interactions in web logs and, then, analyzing these logs in order to uncover relevant information to boost sales. Therefore, web tracking, web analytics and web mining are relevant topics in order to support the analysis of user behavior in web applications.

2.2 Web tracking

According to Ermakova et al. (2018), web tracking refers to a set of techniques for websites to construct user profiles. In a broader sense, web tracking is also understood as a widespread Internet technique that collects user data for purposes of online advertisement, user authentication, content personalization, advanced website analytics, social network integration, and website development (SANCHEZ-ROLA et al., 2017; MAYER; MITCHELL, 2012; ROESNER; KOHNO; WETHERALL, 2012; FOURIE; BOTHMA, 2007). For these goals, web tracking allows third-party or first-party websites to keep track of users' browsing behavior, including browsing configuration and history (SANCHEZ-ROLA et al., 2017).

Figure 1 presents an overview and conceptualization of modern web tracking, showing the role of its major stakeholders. A user accesses websites from a local device through an Internet Service Provider (ISP). Websites and ISPs may include tracking technology, either in-house or made available by Tracking Providers, which provide services to multiple sites (PUGLIESE, 2015). Tracking Providers support cross-site tracking and data aggregation of individual browsing habits and interests, which is not possible for in-house solutions. If the user switches to a different device or moves to another location, cross-device tracking and mobile tracking can be applied (BROOKMAN et al., 2017).

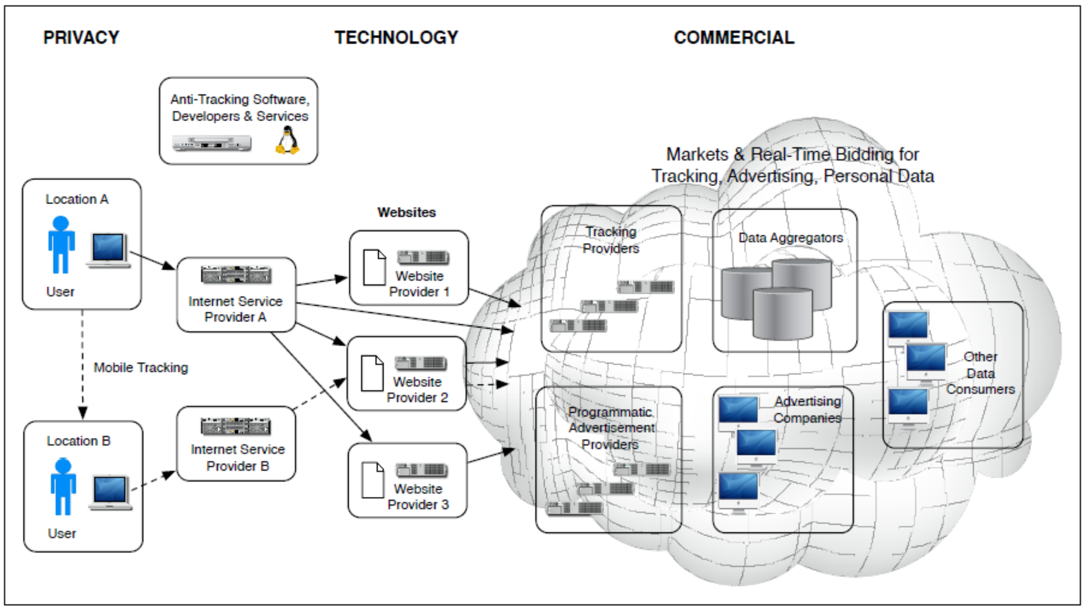


Figure 1 – Overview and conceptualization of web tracking (ERMAKOVA et al., 2018).

Ermakova et al. (2018) state that three main aspects of web tracking are specially relevant: the technology needed to support tracking users, the privacy concerns involved, and the benefits to e-commerce applications.

2.2.1 Technological aspects

According to Mayer e Mitchell (2012) there are two types of tracking techniques: stateful and stateless. Stateful tracking stores the data required for user identification on the client side. On the other hand, stateless tracking gathers users' browser and OS information to proceed the identification.

Stateful tracking techniques exploit cookies, ETags and web storage (PUGLIESE, 2015). Cookies are used to store authentication data. The most common form of tracking is by exploiting third-party cookies, which are used by domains that do not correspond to the currently visited website and are often caused by content provisioning of third parties (PUGLIESE, 2015).

ETag, or Entity Tag, is part of the HTTP mechanism that provides web cache validation and is intended to control how long a particular file is cached on the client side. ETags can be exploited to track users in a similar way to persistent cookies, and a tracking server can continually send ETags to a client browser, even though the contents do not change in the server. By doing this, a tracking server can maintain a session with the client machine that persists indefinitely (HASSAN; HIJAZI, 2017). Web storage involve caches on the client device and can be accessed by browsers and plugins (PUGLIESE, 2015).

Stateless tracking is divided in active and passive fingerprinting (MAYER; MITCHELL, 2012). Fingerprinting is "the process of an observer or attacker uniquely identifying (with a sufficiently high probability) a device or application instance based on multiple information elements communicated to the observer or attacker" (COOPER et al., 2013).

Fingerprinting is ideally suited to identify the device used in the client side (PUGLIESE, 2015). Browser fingerprinting exploits JavaScript to this end. Canvas fingerprinting uses the differences of pixel maps when rendering fonts and WebGL scenes in the browser. Pugliese (2015) also mentions behavioral biometric features, namely those dynamics that occur when typing, moving and clicking the mouse, or touching a touch screen. Such behavioral biometric features can be used to improve stateless tracking.

According to Falahrastegar et al. (2016), increasing awareness of users on data protection and privacy led to browser settings and extensions to delete or prevent certain kinds of cookies and trackers, but new methods are constantly being developed and changed continuously in order to track and identify users.

2.2.2 Privacy aspects

According to Mayer e Mitchell (2012), an user browsing history always present sensitive information. It is possible to reveal users' location, interests, purchases, employment status, sexual orientation, financial challenges, medical conditions, news consumption, and can be used as instrument for mass surveillance by intelligence agencies. Moreover, very personal information regarding the user may be revealed by tracking tools, such as how often a user drinks, smokes and consumes drugs, if the user is pregnant or in menopause, etc (MAYER; MITCHELL, 2012).

These examples show that web tracking can have considerably negative consequences for end users. There exist, however, several techniques can be applied to protect the privacy of the user, such as third-party cookie blocking, clearing the client-side state, blocking popups, AdBlock Plus, Adblock Edge, Ghostery, BetterPrivacy, Site Isolation, EFF's Privacy Badger and private browsing mode (IKRAM et al., 2017).

End-users may take actions in order to protect them while browsing. For instance, a private browsing mode prevents specific data from falling into the hands of other users on the same computer; it also prevents long-term tracking based on stateful techniques. But as long as JavaScript is enabled or certain plugins are installed, device fingerprinting cannot be prevented (PUGLIESE, 2015). Therefore, Sanchez-Rola et al. (2017) suggest disabling other secondary features used in web tracking. Disabling third-party cookies, for instance, can be effective (PUGLIESE, 2015). Another option to fight web tracking is the usage of antitracking web browsers, such as FlowFox (GROEF et al., 2012), TrackingFree and Privaricator (NIKIFORAKIS; JOOSEN; LIVSHITS, 2015).

Akkus et al. (2012) suggest web analytics without tracking. Interestingly, even those prototypes that were designed to protect privacy cannot protect end users against all types of web tracking. Thus, despite the many efforts to protect the privacy of users, being 100% protected is virtually impossible.

2.2.3 Commercial aspects

According to Gill et al. (2013), the collection and use of personal information about users facilitates targeted advertising. Online companies usually apply web tracking to analyze user browsing experience in order to optimize their websites (MELICHER et al., 2016; PUGLIESE, 2015; SANCHEZ-ROLA et al., 2017). By using web tracking, advertising companies are able to offer personalized content and advertisements to users (CLARK; BLAZE; SMITH, 2015).

Ermakova et al. (2018) mentioned the existence of analytics services for third-party websites designed to provide a better understanding of web usage by tracking visitors, such as Adobe Analytics, Quantcast and Google Analytics. Social network companies also provide social integration with single sign-on and track users actions to offer personalized content (MAYER; MITCHELL, 2012). Prominent examples are Facebook like button and Twitter tweet, commonly used to infer users preferences and trends (ERMAKOVA et al., 2018).

2.3 Web Mining

Cooley, Mobasher e Srivastava (1997) defined Web Mining as "the discovery and analysis of useful information from the World Wide Web". This term is also defined as "the application of data mining techniques in large web for large web datasets" (COOLEY; MOBASHER; SRIVASTAVA, 1999).

Web mining is widely used to solve practical problems. Recommender systems takes full advantage from web mining to build powerful personalization systems (PIERRAKOS et al., 2003). E-commerce sites increase their conversion rates (i.e., converting store visits into purchases) by predicting each customer's probability of purchasing based on history of visits and purchases (MOE; FADER, 2004). The performance of web applications can be measured from information extracted from log files (RUFFO et al., 2004), by modeling log information into workload characterization of e-commerce websites.

According to Velásquez e Palade (2008), there are three types of data mining techniques for web data:

- **Web Content Mining**: the content of each web page is used as input to mining algorithms;
- **Web Structure Mining**: the data mining algorithms use as input the structure of web pages and the links among them;
- **Web Usage Mining**: exploit the logs collected from web applications, which are used together with external data (in general, profile data) such as the gender, age, preferences, etc., of the user.

As the Web is a huge collection of heterogeneous, semi-structured and distributed data, web mining is not a trivial task (VELÁSQUEZ; PALADE, 2008). In other to guide such a complex task, web mining generally follows the KDD (Knowledge Discovery on Databases) steps defined by Fayyad et al. (1996), specially the pre-processing, pattern discovery and pattern analysis steps (SRIVASTAVA et al., 2000).

2.3.1 Web Content Mining

Web content mining aims at discovering and extracting useful information and knowledge from the content available on web pages (KLEFTODIMOS; EVANGELIDIS, 2013). Web content mining is very similar to text mining, thus, well know algorithms used in information retrieval may be applied to mine page contents. The main research challenges in this area are (SRIVASTAVA; DESIKAN; KUMAR, 2005):

- Topic discovery, which exploit data mining techniques such as association rules and clustering;
- Associative pattern extraction;
- Web page clustering; and
- Web page classification.

2.3.2 Web Structure Mining

Web structure mining exploits the topology of hyperlinks of websites. This technique may be used to classify the relevance of pages. For instance, Patil e Patil (2012) used web structure mining to classify pages in two categories: relevant and transition. They used this classification to evaluate the similarity of websites.

The literature reports the usage of web structure mining to solve relevant problems. Li e Kit (2005) studied the correlation among the structure of websites and usability problems. Keller e Nussbaumer (2012) developed the MenuMiner algorithm, which allows the extraction of the content structure of large websites. Alqurashi e Wang (2014) developed an approach based on graphs to analyze the navigability of websites.

2.3.3 Web Usage Mining

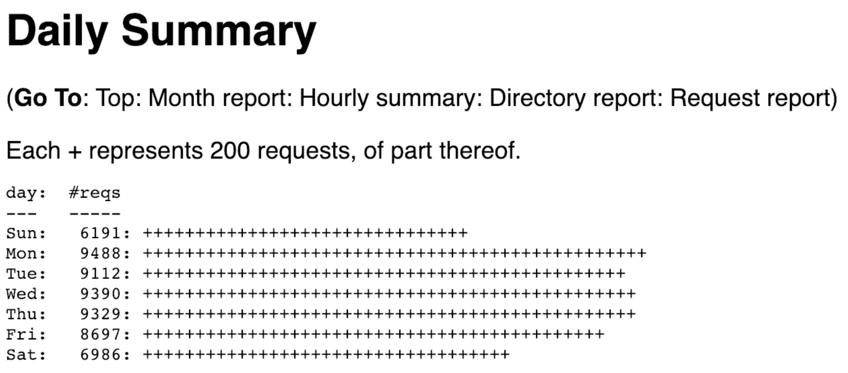
The term Web Usage Mining (WUM) was first introduced by Cooley, Mobasher e Srivastava (1997), according to whom it was "the discovery of usage patterns from web server logs". According to Velásquez e Palade (2008), the aim of WUM is to find usage patterns exploiting different data mining techniques, statistical analysis, association rules, clustering, classification and sequential pattern detection.

Modern web applications are much more complex, because an "one size fits all" website does not provide what users expect. Websites need to know the expectations and needs of their users in order to provide the right content to the right audience. Thus, according to Sudhamathy (2010), web logs need to be analyzed in order to provide usage trends for website developers and administrators, with the aim to support application adaptations.

Chen e Liu (2006) proposed an algorithm that cluster users according to interest similarity detected in their navigation history. Lu, Yao e Zhong (2003) proposed an algorithm that mines server logs in order to detect frequent access that characterize the user navigation in websites. Masseglia, Poncelet e Cicchetti (2000) work exploits association rules to detect relevant sequential patterns. By exploiting these patterns, the developer is able to improve the navigation flow on websites.

2.4 Web Analytics

When users access resources in websites, server logs are usually generated. These logs capture not only the fact that someone hit the website, but also the resource name, time, Internet Protocol (IP), the website/page making the request, etc. As web traffic raised, server log files got larger, and when a script was first written to automatically parse the log, extracting some basic metrics as depicted in Figure 2, Web Analytics was officially born (KAUSHIK, 2007).



Daily Summary

(**Go To**: Top: Month report: Hourly summary: Directory report: Request report)

Each + represents 200 requests, of part thereof.

```
day:  #reqs
---   -----
Sun:   6191: +++++++++++++++++++++++++++++++
Mon:   9488: ++++++++++++++++++++++++++++++++++++++++++++++++
Tue:   9112: ++++++++++++++++++++++++++++++++++++++++++++++
Wed:   9390: +++++++++++++++++++++++++++++++++++++++++++++++
Thu:   9329: +++++++++++++++++++++++++++++++++++++++++++++++
Fri:   8697: ++++++++++++++++++++++++++++++++++++++++++++
Sat:   6986: +++++++++++++++++++++++++++++++++++
```

Figure 2 – A sample report from Analog, version 0.9 beta (KAUSHIK, 2007).

According to the Web Analytics Association, further renamed to Digital Analytics Association (2012), "Web Analytics is the objective tracking, collection, measurement, reporting and analysis of quantitative internet data to optimize websites and web marketing initiatives".

By the year 2000, with the popularity of the Internet growing fast, web analytics gained visibility, and companies such as Accrue, WebTrends, WebSideStory, and CoreMetrics appeared providing solutions to process and analyze massive amounts of data (KAUSHIK, 2007). In 2005, Google bought Urchin, a Web analytics software designed to track the behavior of unique website visitors, which lead to creation of Google Analytics in 2006 (KAUSHIK, 2007).

Web analytics prior work explored basic metrics using log files, such as the number of visitors and the visit duration. As data collection also evolved with the advent of JavaScript tagging, basic metrics evolved to Key Performance Indicators, a.k.a. KPI, in compliance of type of business (KAUSHIK, 2007).

Lamberti et al. (2017) proposed a framework to process data associated to a portion of page, called viewport, and combine it with information about the page structure, obtained from the page's Document Object Model, the DOM-tree structure. Then, results were presented using heat maps, a type of visualization of information that explores colors to express numeric values, used to display clicks or mouse movements.

2.4.1 Web analytics modeling

As pointed in Section 1.1, the analysis of the user interaction requires an appropriate data model, which needs to be expressive enough for representing the user interaction. Server and client logs are noisy, non structured, verbal. The logs have to be structured into something that can be stored and retrieved by analytics tools, in order to display valuable data about the behavior of users. Aiming the prediction of user's next request in a website, Gündüz e Özsu (2003) proposed a model that exploits visiting time and visiting frequencies of pages to extract users' interests, trying to predict navigation patterns.

Eirinaki, Vazirgiannis e Kapogiannis (2005) proposed variations of Markov models to model navigational behavior. By extending the properties of Markov models, they presented a probabilistic predictive model aided by PageRank algorithm to classify the importance of pages according to users' browsing.

Vasconcelos e Jr (2012) proposed a model, called COP model, to define a set of declared interactions as tasks. In other words, by using their model it is possible to define an optimal path for accomplishing a task. This approach measure the similarity between the actual users' interactions and the defined optimal path, in order to calculate an usability index for each task. The COP model is defined by Container, Object and Page, where *object* is any element of the *page*, surrounded by another page element named *container*. This model was designed to define a set of properties to match web elements by its HTML properties, such as tag name, id or class. That way, it is possible to cluster events triggered over many different elements that share common properties.

Cattuto et al. (2013) mapped time-varying social network data into a graph-based model transforming information about users' interactions into labeled graph nodes and relationships. This is another approach using graph theory to model web navigation into graphs to improve mining efficiency by applying path traversal algorithms to extract patterns from users' navigation (MORE, 2014).

Iitsuka et al. (2017) took advantage from graph theory to establish relation between products by extracting the difference between user browsing and purchase behaviors. This approach revealed the superiority relation between competitive products, a win-lose relation between products.

2.4.2 Web analytics architectural concerns

As mentioned in Chapter 1, lots of usage data are required to feed analyzers in order to extract useful data that might represent meaningful information. To acquire that amount of data, preferably in near real-time, it is required an architecture designed to extract and process various types of data in a fast and efficient way (SABTU et al., 2017; PHANIKANTH; SUDARSAN, 2016).

An ETL system, also known as Extract, Transform and Load system, is a good choice to deal with the movement of heterogeneous and volatile data. On the other hand, designing a real-time or near real-time data gathering and processing architecture requires to take into consideration (SABTU et al., 2017):

- Asynchronous data extraction;
- Multiple data source integration;
- Fast and reliable cleaning and transformation;
- Simple data model in compliance with the data warehouse requirements;
- High availability;
- Horizontal scalability.

Chen, Wang e Zhang (2017) proposed a distributed OLAP system to support heterogeneous data types and large amount of calculation in making decisions for big data. The proposed architecture for a distributed OLAP system presents four modules, acquisition module, data storage module, OLAP analysis module and data visualization module.

The data acquisition module purpose is mainly to collect structured non-streaming data, such as data stored in relational database and unstructured streaming data, such as log files. The data storage module is mainly to store raw and processed data. The OLAP analysis module is to organize processed data and to provide services to data visualization module. Finally, the data visualization module is to provide graphical presentation and to offer subject analysis (CHEN; WANG; ZHANG, 2017).

Section 1.1 pointed some commercial tools designed to collect events at client side. Some tools were designed to accomplish specific goals, others were designed as a full customizable solution to map users' interactions.

2.4.3 Google Analytics

According to W3Techs (2009), Google Analytics is used by 55.5% of all websites, which represents a market share of 85.7%. It is a full customizable solution to gather events at client's side, store and visualize valuable information about website users, such as page views, visits and visitors, conversion rates, and much more analytical information. It depends on JavaScript snippet injection on each page in order to activate the collection of data (GAUR et al., 2016).

Section 1.1 mentioned the Google Analytics way to customize events, called *Custom Dimensions and Metrics*. It is a powerful way to collect user defined events, bringing fine-grained information to analysis. However, some configuration and code injections are required to enable websites to collect these *Custom Dimensions and Metrics*.

2.4.4 CrazyEgg

CrazyEgg is a click analytics tool designed to record clicks made by users in a website, analyze and interpret collected data and display events using graphical visualization tools, such as heatmaps, scrollmaps, confetti view and overlay reports (KAUR; SINGH, 2016). It also requires a JavaScript snippet injection and some setup to acquire correctly the interactions.

2.4.5 HotJar

HotJar is a tool also designed to collect users interactions, providing heat maps of pages, conversion funnels and something different from the aforementioned tools, which is a visitor recording Lamberti et al. (2017). It records videos from random and anonymous users' interactions, very useful to detect trouble with pages' design, for instance, whether a user is having trouble with a register form, or not able to place a purchase order. As other tools, is also requires a JavaScript snippet injection.

2.5 Final remarks

The Footstep system exploits web tracking to provide user identification by the use of cross-domain cookies. Footstep identifies users, not necessarily exposing personal information, by defining a random ID at first user's interaction, which is saved into a cross-domain cookie.

The Footstep approach exploits Web Mining concepts, specially web structure mining, by the automatic discovering and identification of pages' elements related to events, such as clicks and taps. The element discovering allows Footstep to measure element audience and analyze how elements affect navigation and conversion.

Footstep also relies on Web analytics ideas by taking advantage of the UsaGraph model to provide insights and information about the behavior of users, providing dashboards and charts to display navigation flows, conversion rates, as well as pages and elements audience.

Chapter 3 presents UsaGraph model, explains its characteristics and shows how to take advantage from this model to extract useful information regarding on users' interactions.

3 UsaGraph Model

3.1 Initial considerations

Many efforts have been spent on analysis of user behavior and modeling is paramount to translate human behavior to a computational environment. Each data model should contemplate as many as the characteristics of the subject of study in an intuitive way. In order to address a specific problem solution, a data model must take into account computational and user defined requirements, such as acceptance criteria, response speed, ease of use, available technologies and so on.

In order to define a concise data model, it is paramount to understand the situation, which problems has to be tackled, how they should be solved and what are the expected results.

Section 3.2 describes the main characteristics of users' navigation. Then, section 3.3 explains how users' interactions can be mapped to a graph data model.

3.2 Web navigation flow

One of the definitions of flow is the movement of something in one direction. Users can navigate among web pages using links, opening a page, clicking on page elements, filling and submitting forms, and so on. On the other hand, most websites are designed to have a logical sequence of events, a concise data flow, as an e-commerce website, for instance, which has a list of products, an *add to cart* feature, a shopping cart, a checkout page and, finally, a *thank you* page, where users can see the purchased products. Each user has its own life cycle inside a web page, a sequence of events telling a story about the pages and page elements that the user had interacted with. In a high-level view, each event triggered on a website shares the following characteristics:

- One user triggers an event;
- This event occurs in a page;
- This event may have a target element;
- The target element belongs to a structured tree of parent elements.

3.3 Event data modeling

There are different ways to model data and each one has its strengths and weaknesses. Our approach leverages from graph theory concepts, exploiting the facilities to extract correlations from graph structures.

On web applications, events have properties and characteristics that can be expressed by nodes and relationships among these nodes (SIQUEIRA; BALDOCHI, 2018):

- An event has an event type, a date when it was triggered, as well as information about the mouse position;
- A page have an URL, a HTML document composed of HTML elements and it is rendered in different browsers and devices;
- A user can be uniquely identified;
- A HTML element has a tag name and common attributes, such as id, name, class, etc. It also belongs to a structured HTML document, forming the so called DOM-tree structure, where each element has a parent element, until the root element of the HTML document.

3.4 DOM-tree structure and event propagation

The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM, documents have a logical structure organized as a tree of elements, since each element has a single parent element and may have none or many children elements[1].

Section 3.5.1 presents a hypothetical Register Page in Figure 5 to demonstrate how to instantiate UsaGraph. Giving the same hypothetical register form, understand its DOM-tree structure is paramount to illustrate how a triggered event can carry detailed information about the target element. Listing 3.1 shows the HTML code of register page with its elements placed under a well-defined document structure, technically known as DOM-tree structure, and Figure 3 illustrates how elements are organized as a tree.

[1] What is the Document Object Model? [online] <https://www.w3.org/TR/DOM-Level-2-Core/introduction.html>

Listing 3.1 – Hypothetical Register Form HTML

```
<html>
  <head>
    <title>Website</title>
  </head>
  <body>
    <div class="container">
      <form id="form-register">
        <input name="email" type="text" class="required" placeholder="E-mail" />
        <input name="password" type="password" class="required" placeholder="Password" />
        <button id="register" class="btn␣btn-primary">Register</button>
      </form>
    </div>
  </body>
</html>
```

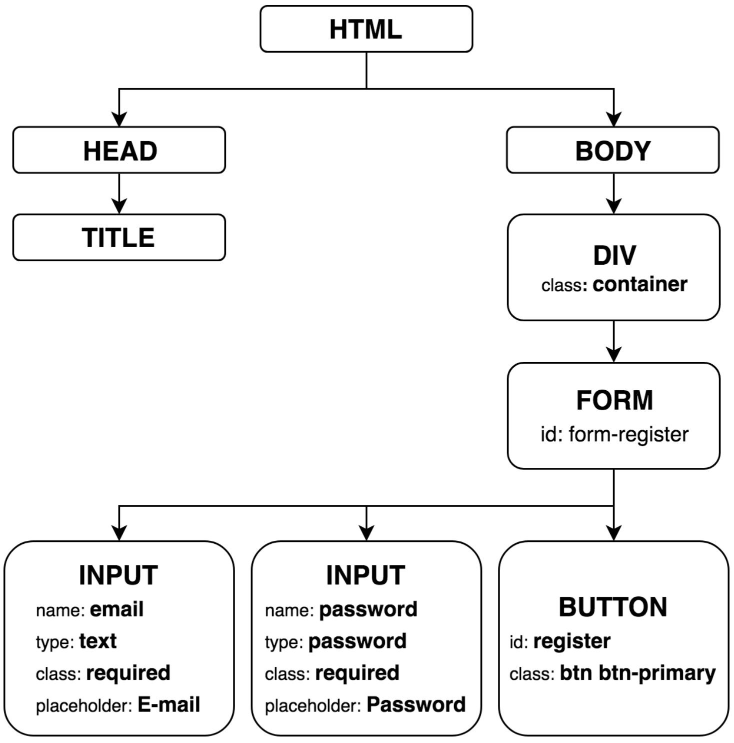


Figure 3 – Tree representation of HTML register page code.

When an event is triggered in a web page, it may occurs over an element or not. For example, when a page is loaded, a document load event is triggered, which is not necessarily triggered over one element, but is triggered in document level. On the other hand, when event is triggered over an element, this event may propagates through this element's parents. In other words, taking the Register page as scenario, when user clicks on password input, its click is also triggered over form, container div, body and html elements. This chain of event propagation brings an opportunity to extract information about websites' structures, and how they are designed, how frequent elements are accessed.

It is important to notice that two distinct events may occur over two distinct elements under the same parent, which also receives these two events, thus both events was triggered over the same parent element.

We exploit these properties and characteristics to build Usagraph – a graph-based data model that is efficient for processing web logs.

3.5 UsaGraph Model

The UsaGraph model was designed to represent the characteristics described in Section 3.3 using the graph theory. It is initially composed by four main types of nodes: `User`, `Event`, `Page` and `Element` nodes. Each node `User` represents the user that triggers an event, which is unique by the `USER-ID` identifier. Each node `Event` represents the triggered event itself and stores the type, such as `pageview` or `click` for instance, and the date time of the event. Each node `Page` is unique by its URL address. Each node `Element` represents an element and its ancestors in the DOM tree, and stores information such as the element's tag name, id, class, name and type. The UsaGraph model is further enriched by two more types of nodes: `PageLabel` and `ElementLabel`. Their main purpose is clustering, respectively, `Page` and `Element` nodes, according to rules that make it possible to generalize these nodes. The four main types are connected by labeled relationships, following the characteristics presented in Section 3.3. Finally, pages and elements can be generalized, labeled and grouped. UsaGraph model is graphically represented in Figure 4.

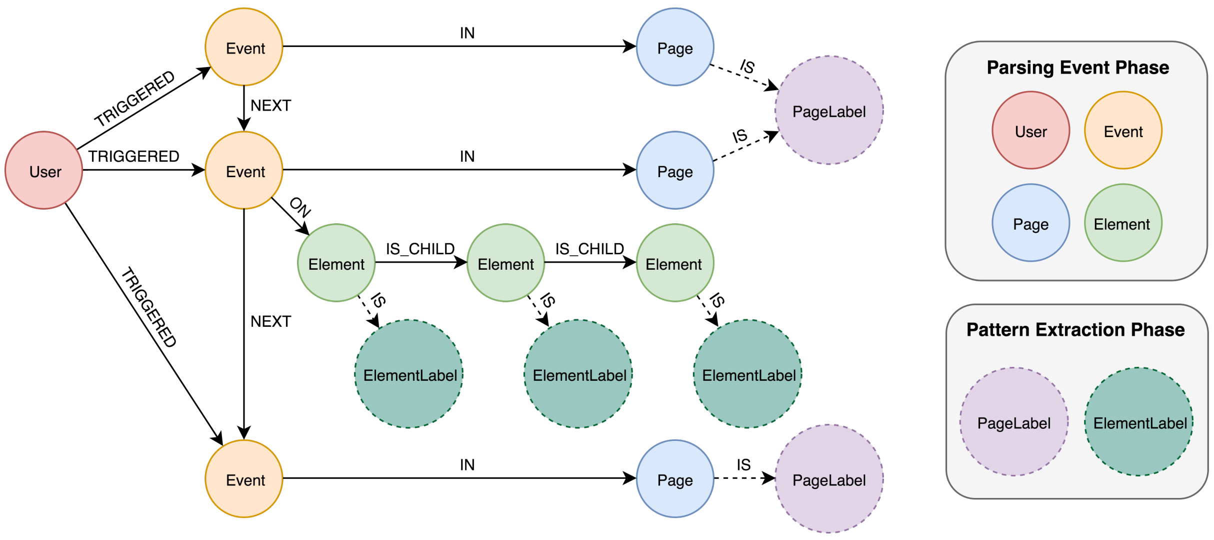


Figure 4 – UsaGraph model.

The `PageLabel` node type is responsible to classify `Page` nodes according to some predefined rule. It is possible to join pages by any URL's pattern, or even by any parameter present in URL. It is also possible to join pages according to business rules, such as home page, listings page or products page. It was designed to be an abstraction of `Page`, a high-level way to represent them.

The `ElementLabel` node type has a similar purpose, but at `Element` node's level. It was designed to detect similarities between elements and, as opposite of `PageLabel`, has a more restrict rule to cluster elements. It takes some of the main characteristics of the elements to classify them as one `ElementLabel`:

- elements must have the same tag name;
- they have to be at the same level in document structure, belong to the same parents;
- same attribute id, if defined;
- same attribute name, if defined;
- same attribute type, if defined;
- at least one of the class attribute in common, if defined.

3.5.1 UsaGraph Instantiation

In order to present UsaGraph in action, let us consider a simple website register page, illustrated in Figure 5. As an example, a user may open this page and perform three clicks, the first one on the e-mail input field, the second on the password input field and, finally, the third on the register button. This chain of events, translated to UsaGraph model, can be represented by one `User` node, one `Page node`, four `Event` nodes (one pageview, three clicks) and some `Element` nodes. Figure 6 shows the UsaGraph representation of this user's interaction. It is important to note that this extracted graph has only the four main types of the UsaGraph nodes, because until now there is no need to make any kind of generalization due to the simplicity of this scenario.

To demonstrate the generalization feature of our model, let us consider another hypothetical scenario of two different pages from the same website, as in Figure 7. It is possible to note some similarities between them. Both have header, products list and footer sections, but display different products and links. Therefore, let us also consider that these two pages belong to the same page type, i.e., the products page.

Since these two pages belong to the same page type, only one `PageLabel` is created and labeled as PRODUCTS, and both `Page` nodes must be connected to it. From there, their elements can be generalized, as long as they share most of its characteristics. To demonstrate the element generalization, consider that two users are browsing these pages: User A performs a click on **Product ABC** of Page 1, while User B clicks on **Link 1** of Page 2. After the translation of these two clicks to the UsaGraph model, we would have

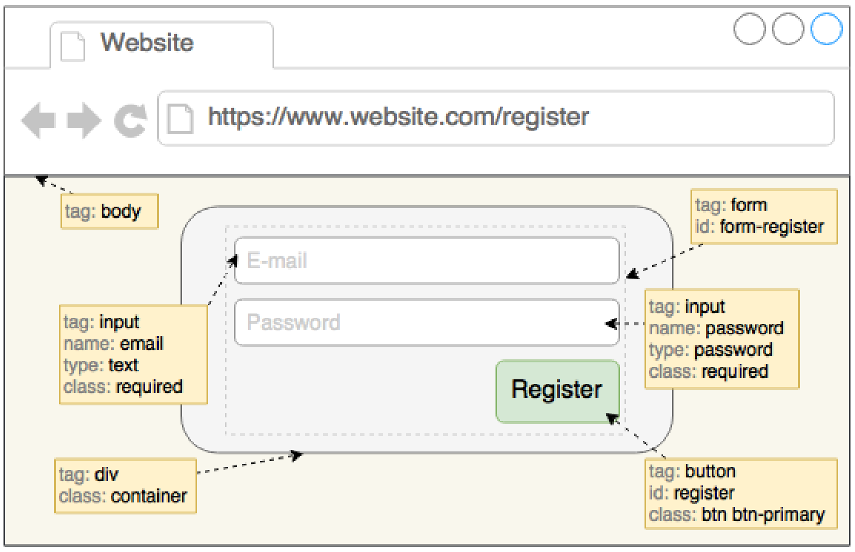


Figure 5 – Hypothetical website with a simple register form.

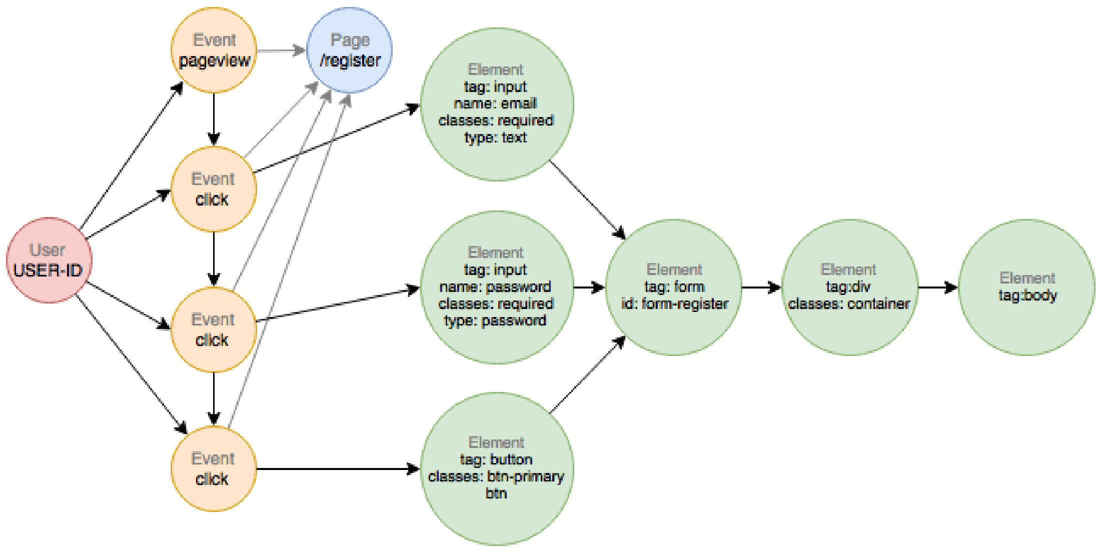


Figure 6 – Hypothetical scenario translated to UsaGraph model.

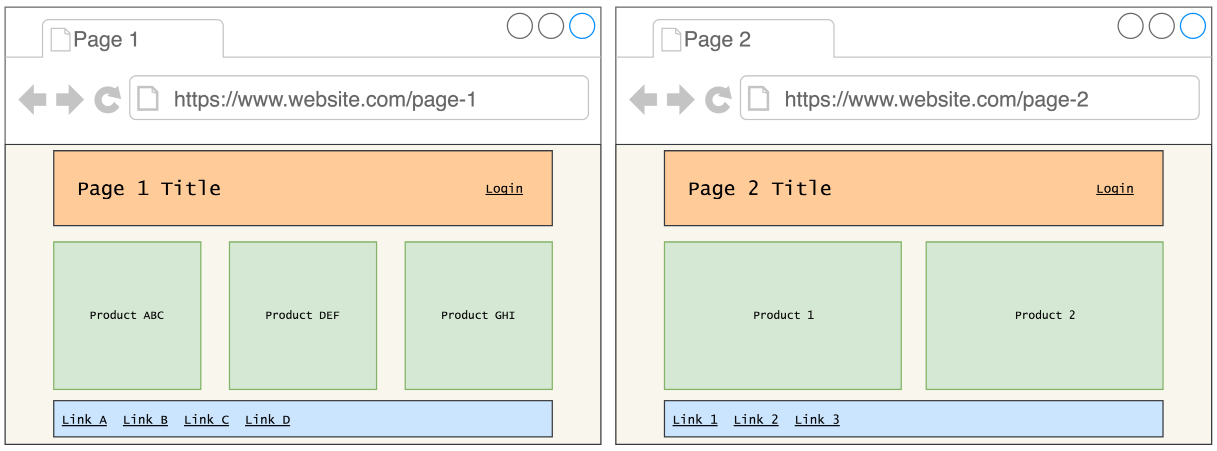


Figure 7 – Two different pages from same website.

two distinct users' interactions, as shown in Figure 8. The first gray area demarcates User A's interaction in Page 1, on Product ABC link element (and its ancestors elements). The second gray area demarcates User B's interaction in Page 2 on Link 1 link element.

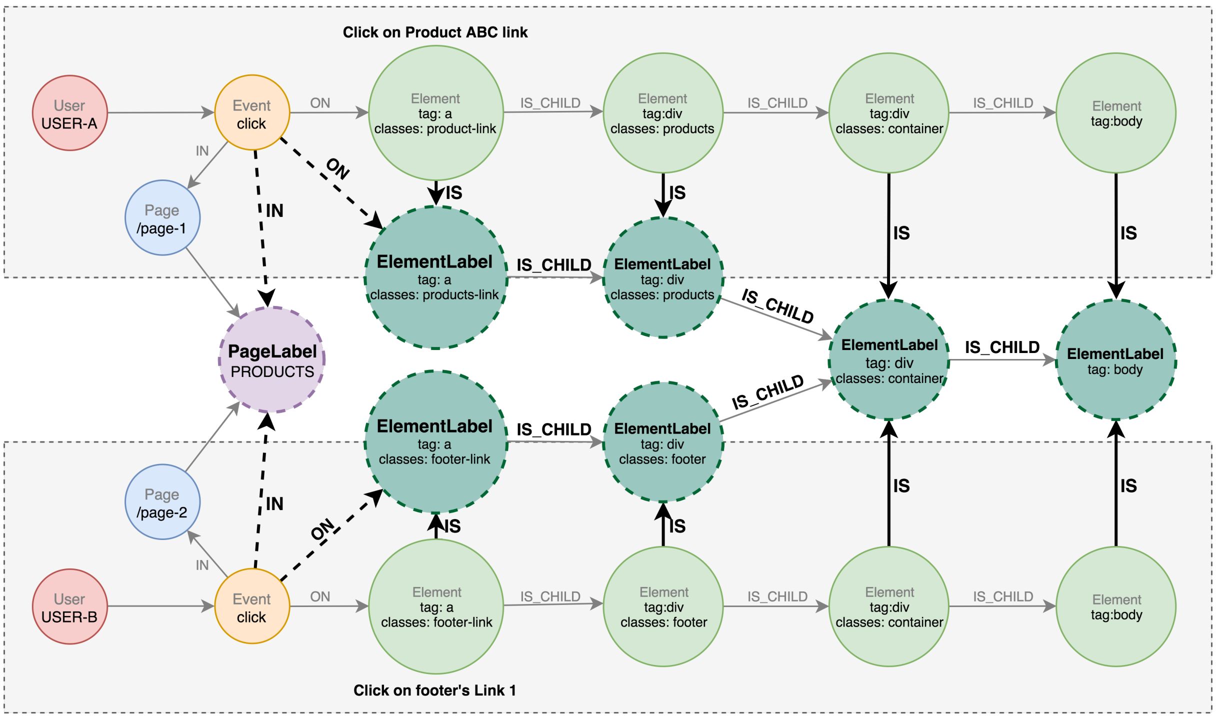

Click on Product ABC link
User USER-A
Event click
ON
Element tag: a classes: product-link
IS_CHILD
Element tag:div classes: products
IS_CHILD
Element tag:div classes: container
IS_CHILD
Element tag:body
IN
Page /page-1
IN
ON
IS
IS
IS
IS
ElementLabel tag: a classes: products-link
IS_CHILD
ElementLabel tag: div classes: products
IS_CHILD
PageLabel PRODUCTS
ElementLabel tag: div classes: container
IS_CHILD
ElementLabel tag: body
IS_CHILD
ElementLabel tag: a classes: footer-link
IS_CHILD
ElementLabel tag: div classes: footer
Page /page-2
IN
ON
IS
IS
IS
IS
IN
User USER-B
Event click
ON
Element tag: a classes: footer-link
IS_CHILD
Element tag:div classes: footer
IS_CHILD
Element tag:div classes: container
IS_CHILD
Element tag:body
Click on footer's Link 1

Figure 8 – UsaGraph model representation of two clicks performed by different users, in different pages.

The events' target elements, **Product ABC** and **Link 1** links, are placed in different sections, respectively, in **div with class products** and **div with class footer**. This means they could not belong to the same `ElementLabel`, that is, each target element is linked to a respective new `ElementLabel` node. The same occurred with theirs immediate parents, because they do not fulfill the requirements to be clustered with the same element label. The elements **div with class container** and **body** could be clustered. They are at the same position in DOM-tree document structure, shares tag name and at least one class. So, one `ElementLabel` node was created for two elements **div with class container** and other for two elements **body**. It is important to note two new types of relationship created to reduce queries overhead: `Event-IN->PageLabel` and `Event-ON->ElementLabel` relationships. As expected, the number of `PageLabel` and `ElementLabel` is smaller than the number of `Page` and `Element` nodes. So, the creation of these two relationships allows to query smaller graphs, instead of querying over `Page` and `Element` nodes.

As mentioned in Section 3.5, the UsaGraph model was designed to take advantage from graph theory to discover paths between pages and element labels. To demonstrate how UsaGraph can be used to extract navigation flow patterns through pages and elements, let us consider two users navigation on a website. Their events represented by UsaGraph model is shown in Figure 9.

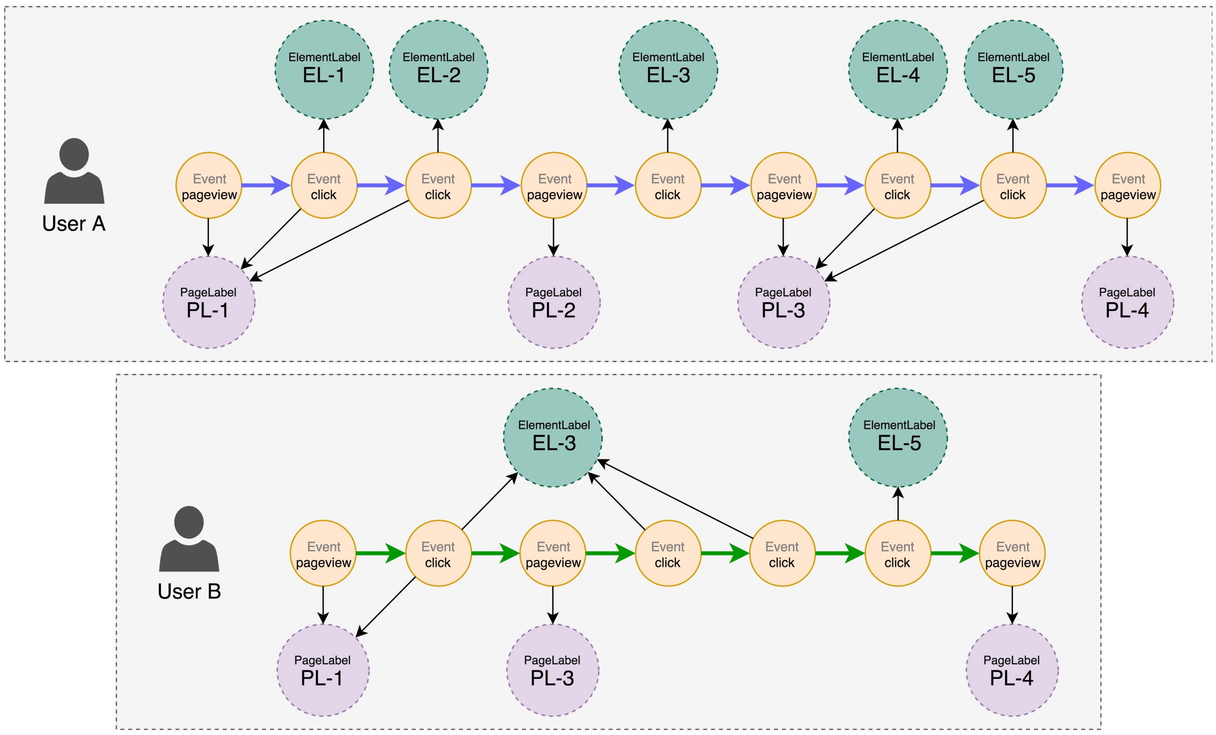


Figure 9 – UsaGraph representation of events triggered by User A and User B.

Both users started accessing page label PL-1 and both also ended their navigation in page label PL-4, so, it is possible to say that there is a flow of two users from PL-1 to PL-4. However, each user made a different path to reach PL-4. User A interacted with two elements from PL-1, visited PL-2, then PL-3, and finally, PL-4, in a total of 9 events triggered. User B took only 7 events to reach PL-4, did not visit PL-2, never interacted with elements EL-1, EL-2 and EL-4, illustrated by Figure 10. It is important to point out that Figure 10 is only an illustration of extracted flow using `PageLabel` and `ElementLabel` nodes, it is not a UsaGraph representation.

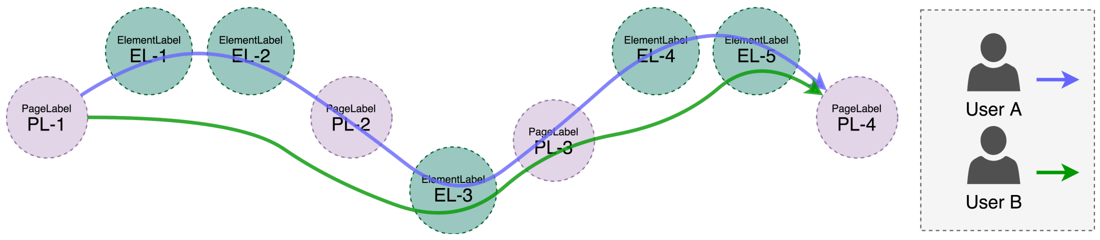


Figure 10 – Illustration of navigation flows from PL-1 to PL-4.

Figure 10 brings more insights about more valuable information that could be extracted from UsaGraph model. It is possible, for instance, to measure the utility of page's elements. If the elements EL-1 and EL-2 were really important to users, something is wrong, because User B could avoid them and reach page PL-3 directly. This directly affects the conversion. If a conversion point is defined, such as a page view on a specific page, or click on a specific element, it is possible to determine the interactions that reached this conversion point and the interactions that avoided it. Looking at Figure 9 again, if the conversion point was defined as a page view on PL-2, only User A converted. However, if the conversion point was defined as a click on EL-5, both users converted, although User B converted faster, because it took fewer steps than User A to click on EL-5.

3.6 Final remarks

This chapter presented UsaGraph, a graph-based model that allows the representation of client events so as to favour the discovering of usage patterns and other relevant information for the management of any website.

The main advantage of the proposed model is the usage of detailed events, that allows the identification of the page elements related to each user event. By exploiting this data, it is possible to deliver more fine-grained information about the users' navigation, such as navigation patterns between pages and elements. It is possible, for instance, to trace a specific user, at a specific time, and know where exactly he or she got lost from a desired action. More importantly, UsaGraph allows the identification of the interface elements related to the problem. Thus, identifying the source of the problem becomes easier.

Chapter 4 presents the Footstep System architecture and how it was designed to fulfill real-time ETL system requirements pointed out in Section 2.4.2.

4 The Footstep System

4.1 Initial considerations

Tracking user actions using client logs usually requires collecting a large amount of data. As modern e-business web applications have hundreds or even thousands of users browsing their pages concurrently, several gigabytes of log data may be produced in a matter of minutes.

An important feature of any analytics tool is its ability to provide up-to-date data regarding the usage of a website. Thus, it is important to collect, store and process logs as fast as possible.

There are several analytics tools designed to take advantage of browsers' native support to JavaScript, as discussed in Section 2.4.2. These tools usually provide libraries to be installed on web pages. Ideally, those libraries should be small and collect events in an economic fashion.

Available analytics tools are not fully capable to meet these requirements. On the other hand, there are many tools and programming languages that can be combined to build an approach that fulfill the required needs to deal with high volume of data in a fast and effective way.

This chapter presents Footstep, an approach built to collect, transform, store and process log data. Footstep collects client logs, parses this data exploiting the UsaGraph model and stores the resulting nodes and vertexes in a graph database so as to allow fast processing. Finally, Footstep provide a visualization tool that allows summarizing valuable information for the management of websites.

4.2 Architectural overview

Figure 11 presents an overview of Footstep's architecture, from collecting user events on websites to the analysis of these events using a graph database. The designed architecture is composed of four main modules: Gathering Events, Consumer & Parser, Generalization and Analytics.

In order to collect events at users' browsers side, the website must add a JavaScript library responsible for capturing, translating and sending users' events to the Events Listener Service. The Gathering Events Module must be fast in order not to burden the website experience, thus, it only passes on the events to an event queue, which is asynchronously processed. This queue must be consumed very quickly in order not to become a bottleneck to the entire system. The Consumer & Parser Module is in charge of consuming this queue, parsing events to the UsaGraph Model.

Then, all parsed events are persisted into a graph database. The Generalization Module provides an application responsible for identifying and extracting navigation patterns over the persisted events' nodes. This application depends on another database, which describes commons URL structures useful for clustering different pages' URL's. Finally, the Analytics Module provides an application responsible for querying the database and providing usable dashboards, charts and analytical data about navigation flows.

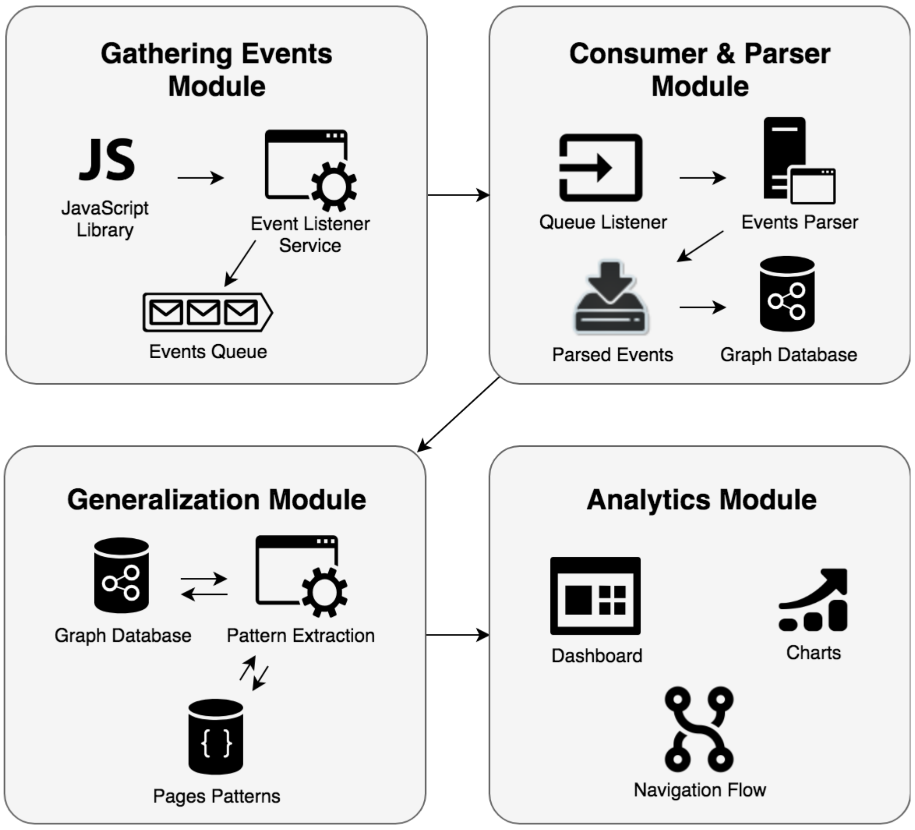


Figure 11 – Footstep architectural overview.

4.3 Gathering Events Module

The Gathering Events Module is composed by three components: Event Collector JavaScript Library, Event Listener Service and Events Queue. Websites must reference the JavaScript Library at the end of their HTML pages in order to capture users' interactions on pages. Each event captured by this library is sent to the Event Listener Service, which is responsible to receive events and send them to the Consumer & Parser Module in a asynchronous and fast way, using the Events Queue.

4.3.1 Events collection library design

The Footstep architecture provides a *JavaScript* library, responsible for collecting detailed client events, which was designed to be as unintrusive as possible and fully customizable. This library was designed to collect web events like *page view*, *click*, *key up*, *focus*, *blur*, *tap*, *swipe*, among others. For each collected event, information regarding the target element and its ancestor in DOM-tree structure is also collected. For each element and ancestor, the library collects the tag name and the following attributes: *id*, *class*, *name* and *type*. It is important to mention that this library does not collect any sensitive data, only DOM-tree elements (and some attributes) are collected, and users can turn off the event tracking, if desired. Each event triggered by user is translated and sent asynchronously to a collector system that logs this event in log files. It is recommended to install this library at the end of the HTML document in order not to affect the users' experience. In this way, when the library is loaded, all HTML document is already loaded too.

Each collected event must be sent to the Event Listener Service using a HTTP request. This can be achieved by using Asynchronous JavaScript and XML (Ajax). Ajax calls require JavaScript object creation and may offer an overhead of processing at client side. If the HTTP request is a GET method request, there is a lighter and tricky way to do the same, that consists on appending an image tag on the HTML document whose image source address is the URL of the HTTP request. When an image tag is placed into the HTML document, the browser will try to access the image source in order to display it. When browser tries to access this image URL, a HTTP GET call is made to the server where URL points to. Thus, the only requirement to server is to respond a transparent image, containing just a few bytes, in order to emulate a valid image to place into the HTML document with no side effect to pages design. This mechanism is fast, asynchronous, easy to implement and is inspired by the way Google Analitics (2005) collects events.

As Section 2.4.3 mentioned, Google Analytics enable websites to collect fine-grained events, by using *Custom Dimensions and Metrics*. However, to achieve this, websites must inject some code in order to push events containing those custom information. Footstep JavaScript library aims to address this shortcoming by automatic element detection and send the entire element information among events to the Event Listener Service.

4.3.2 The anatomy of detailed client event

Section 3.5 proposed a model that represents an HTML document structure using nodes and vertexes. Section 4.3.1 described how the collector library builds and sends each action performed by the user to the collector system. Section 3.4 explained how events propagate though elements and its parents. In order to capture events containing detailed information about elements, each event must be translated to a single call to the Event Listener Service. To fulfill the requirement of economy of data transfer, each call should be as short as possible and still contain detailed information. The level of detail extracted by the collector library includes the following information:

- User's identifier;
- User's session identifier;
- Date when the event was triggered;
- Type of event (*pageview*, *click*, *keyup*, *focus*, *blur*, *tap*, *swipe*, etc.);
- Page's URL where the event occurred;
- User agent information (browser, operating system, etc.);
- Mouse position (the exact position of the cursor, if the event was triggered by mouse);
- Element that received the event and its ancestors in HTML document structure (optional). Each element may have these information:
  - Tag name (*a*, *div*, *span*, *label*, *input*, etc.);
  - ID (HTML attribute: *id*);
  - Classes (HTML attribute: *class*);
  - Name (HTML attribute: *name*);
  - Type (HTML attribute: *type*);
  - Position (the exact position of element in document);

Each event to be pushed to the Event Listener Service must comply with a predefined format in order to be further processed without any loss of information. Listing 4.1 shows events that are not related to elements, such as a *page view*. Listing 4.2 shows events triggered on elements. It is important to mention that both Listing 4.1 and Listing 4.2 are represented in many rows just for a better understanding.

Listing 4.1 – Page view event (no element as target)

```
https://EVENT-LISTENER-SERVICE-HOST/collect?
  id=WEBSITE-ID&
  u=USER-ID&
  i=SESSION-ID&
  e=EVENT-ID&
  p=PAGE-URL&
  ua=USER-AGENT
```

Listing 4.2 – Event performed on an element

```
https://EVENT-LISTENER-SERVICE-HOST/collect?
  id=WEBSITE-ID&
  u=USER-ID&
  i=SESSION-ID&
  e=EVENT-ID&
  p=PAGE-URL&
  ua=USER-AGENT&
  m=MOUSE-TOP,MOUSE-LEFT&
  el=t:TAG,i:ID,n:NAME,tp:TYPE,c:CLASS,ot:TOP,ol:LEFT&
  an=
    t:AN-0-TAG,i: ... ,c:AN-0-CLASSES, ... ,ol:AN-0-LEFT,
    t:AN-1-TAG,i: ... ,c:AN-1-CLASSES, ... ,ol:AN-1-LEFT,
    ...
    t:AN-N-TAG,i: ... ,c:AN-N-CLASSES, ... ,ol:AN-N-LEFT
```

This format was designed in this way to transfer less data through Internet and thus impose less overhead to these calls, fulfilling the economy of data transfer requirement, so each argument name is as short as possible and represent an unique type of information:

- Website's identifier: **`id`**;
- User's identifier: **`u`**;
- Session's identifier: **`i`**;
- Event's identifier: **`e`**;
- User Agent's identifier: **`ua`**;
- Mouse position's identifier: **`m`**;
- Element's identifier: **`el`**;
- Element's ancestors' identifier: **`an`**.

Each element and ancestor element is translated to another predefined format, containing information about its tags and some of its attributes, also represented by unique short argument names:

- Tag name: **`t`**;
- Tag attribute id: **`i`**;
- Tag attribute class: **`c`**;
- Tag attribute name: **`n`**;
- Tag attribute type: **`tp`**;

- Tag offset top position: **`ot`**;
- Tag offset left position: **`ol`**.

Taking the same hypothetical scenario described in Section 3.5.1, when the user opens the Register page and performs three clicks, Figure 6 shows the UsaGraph representation of these four events. Listings 4.3, 4.4, 4.5 and 4.6 demonstrate inner details of these events.

Listing 4.3 – Hypothetical scenario's first event: Open register page

```
https://EVENT-LISTENER-SERVICE-HOST/collect?
  id=website&
  u=user-id&
  i=session-id&
  e=pageview&
  p=https%3A%2F%2Fwww.website.com%2Fregister&
  ua=Mozilla
```

Listing 4.4 – Hypothetical scenario's second event: Click E-mail field

```
https://EVENT-LISTENER-SERVICE-HOST/collect?
  id=website&
  u=user-id&
  i=session-id&
  e=click&
  p=https%3A%2F%2Fwww.website.com%2Fregister&
  ua=Mozilla&
  m=112,23&
  el=t:input,n:email,tp:text,c:register-input,ot:100,ol:20&
  an=t:form,i:form-register,ot:30,ol:10;t:div,c:container;t:body
```

Listing 4.5 – Hypothetical scenario's third event: Click Password field

```
https://EVENT-LISTENER-SERVICE-HOST/collect?
  id=website&
  u=user-id&
  i=session-id&e=click&
  p=https%3A%2F%2Fwww.website.com%2Fregister&
  ua=Mozilla&
  m=156,34&
  el=t:input,n:password,tp:password,c:register-input,ot:150,ol:20&
  an=t:form,i:form-register,ot:30,ol:10;t:div,c:container;t:body
```

Listing 4.6 – Hypothetical scenario's third event: Click Register button

```
https://EVENT-LISTENER-SERVICE-HOST/collect?
  id=website&
  u=user-id&
  i=session-id&
  e=click&
  p=https%3A%2F%2Fwww.website.com%2Fregister&
  ua=Mozilla&
  m=215,51&
  el=t:button,i:register,c:btn%20btn-primary,ot:200,ol:40&
  an=t:form,i:form-register,ot:30,ol:10;t:div,c:container;t:body
```

4.3.3 Event Listener Service

As Section 4.2 pointed out, events are captured in websites using the native browsers' support to JavaScript and sent to the Event Listener Service. The Event Listener Service consists in a single Web service endpoint receiving meaningful data as parameters. Section 4.3.2 described these parameters containing the required data to be further translated to UsaGraph model.

Each call to Event Listener Service must take into account the economy of data and this also applies to the service response. In fact, the Event Listener Service response is useless. Once the Footstep JavaScript captured and sent detailed event to the event listener, no response back is expected.

There are some ways to make HTTP calls from browsers, such as *XMLHttpRequest* calls. But, based on Google Analytics, there is another tricky way to make GET HTTP requests through HTML image sources. When browsers detect image tags, they evaluate their source attribute expecting an image response, containing response headers such as `image/gif`, `image/jpeg`, `image/png`. If Event Listener Service returns an image content type response, it can be used as image tag source attribute. In other words, event listener `/collect` calls respond a fake and transparent GIF image containing a minimum length as possible. So, Footstep JavaScript place an image tag in page HTML document for every event containing the collect call with detailed event parameters as image tag source attribute.

4.3.4 Events Queue

To make a bridge between producer and consumer, our architecture exploits an open source, lightweight message broker, widely used by many systems, called RabbitMQ (2007), which supports multiple messaging protocols and meet high-scale, high-availability requirements. RabbitMQ's architecture (PIVOTAL SOFTWARE, INC., 2017), as shown in Figure 12, has two structures designed to deal with producers and consumers (Exchanges and Queues, respectively) (NANNONI, 2015). Producers should not be aware of the routing of their messages, or even which consumers will ingest this information. For the same reason, consumers ingest queues' messages, regardless of the origin of those messages. That logical separation between exchanges and queues brings flexibility on message routing strategies. RabbitMQ can be also configured as a distributed system, including many clusters, and can operate either synchronously or asynchronously.

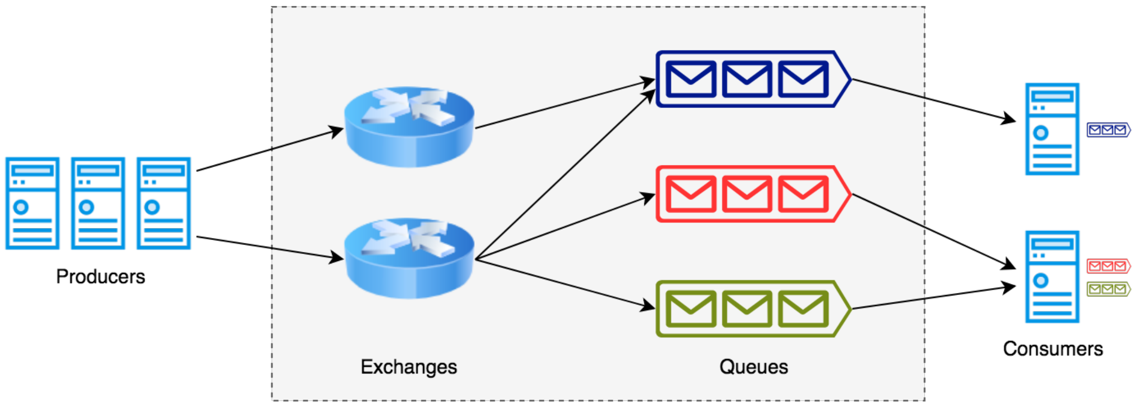


Figure 12 – Simplified RabbitMQ architecture.

4.4 Consumer & Parser Module

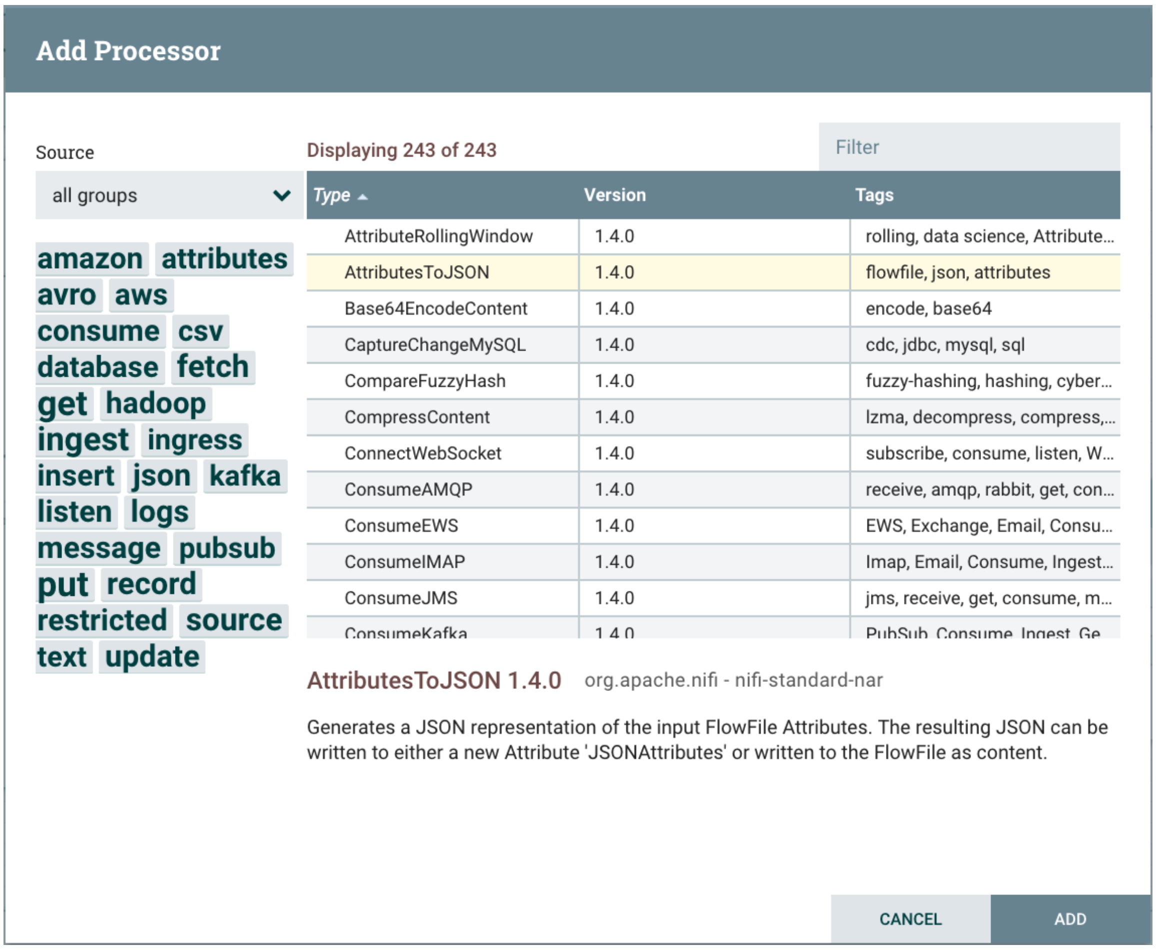
Consumer & Parser Module presents two main components: the Events Queue consumer and the Events Parser. Data integration is paramount in Footstep architecture, and there are several possibilities in terms of technology to implement distributed systems capable of data exchange and processing. The Consumer & Parser Module is built upon Apache NiFi (2006), a powerful and reliable system designed to performs data routing and transformation, as well as system mediation logic. Apache NiFi is suitable for Consumer & Parser Module because it has powerful ways to transfer and transform data.

4.4.1 Apache NiFi

Apache NiFi is highly configurable, contains many types of fully customizable processors, responsible for consuming, processing, transforming and delivering data. All processors run asynchronously and can connect to each other by queues, providing back pressure mechanism and guaranteed delivery of data. The processors can be organized and logically grouped into process groups, and further used as new components. Apache NiFi provides a Web-based user interface called NiFi Flow. By using this tool it is possible to add and connect processors as needed.

Apache NiFi version 1.4 offers many types of processors, such as queue consumers, processors to read and write to different databases, for reading and writing files and different kinds of text processors. Figure 13 depicts the screen where it is possible to select a processor. Beyond selecting predefined processors, it is also possible to define and implement custom processors.

NiFi processors operate over *FlowFiles*. A *FlowFile* is a data record, which consists of a pointer to its content (payload) and attributes to support the content (APACHE NIFI, 2018). As NiFi is a flow-based architecture, understanding how *FlowFiles* are created and transferred between processors is paramount to design and connect processors in a concise way.



Add Processor

Source

all groups

amazon attributes avro aws consume csv database fetch get hadoop ingest ingress insert json kafka listen logs message pubsub put record restricted source text update

Displaying 243 of 243

Filter

Type	Version	Tags
AttributeRollingWindow	1.4.0	rolling, data science, Attribute...
AttributesToJSON	1.4.0	flowfile, json, attributes
Base64EncodeContent	1.4.0	encode, base64
CaptureChangeMySQL	1.4.0	cdc, jdbc, mysql, sql
CompareFuzzyHash	1.4.0	fuzzy-hashing, hashing, cyber...
CompressContent	1.4.0	lzma, decompress, compress,...
ConnectWebSocket	1.4.0	subscribe, consume, listen, W...
ConsumeAMQP	1.4.0	receive, amqp, rabbit, get, con...
ConsumeEWS	1.4.0	EWS, Exchange, Email, Consu...
ConsumeIMAP	1.4.0	Imap, Email, Consume, Ingest...
ConsumeJMS	1.4.0	jms, receive, get, consume, m...
ConsumeKafka	1.4.0	PubSub, Consume, Ingest, Ge

AttributesToJSON 1.4.0 org.apache.nifi - nifi-standard-nar

Generates a JSON representation of the input FlowFile Attributes. The resulting JSON can be written to either a new Attribute 'JSONAttributes' or written to the FlowFile as content.

CANCEL ADD

Figure 13 – Apache NiFi - Add Processor Window.

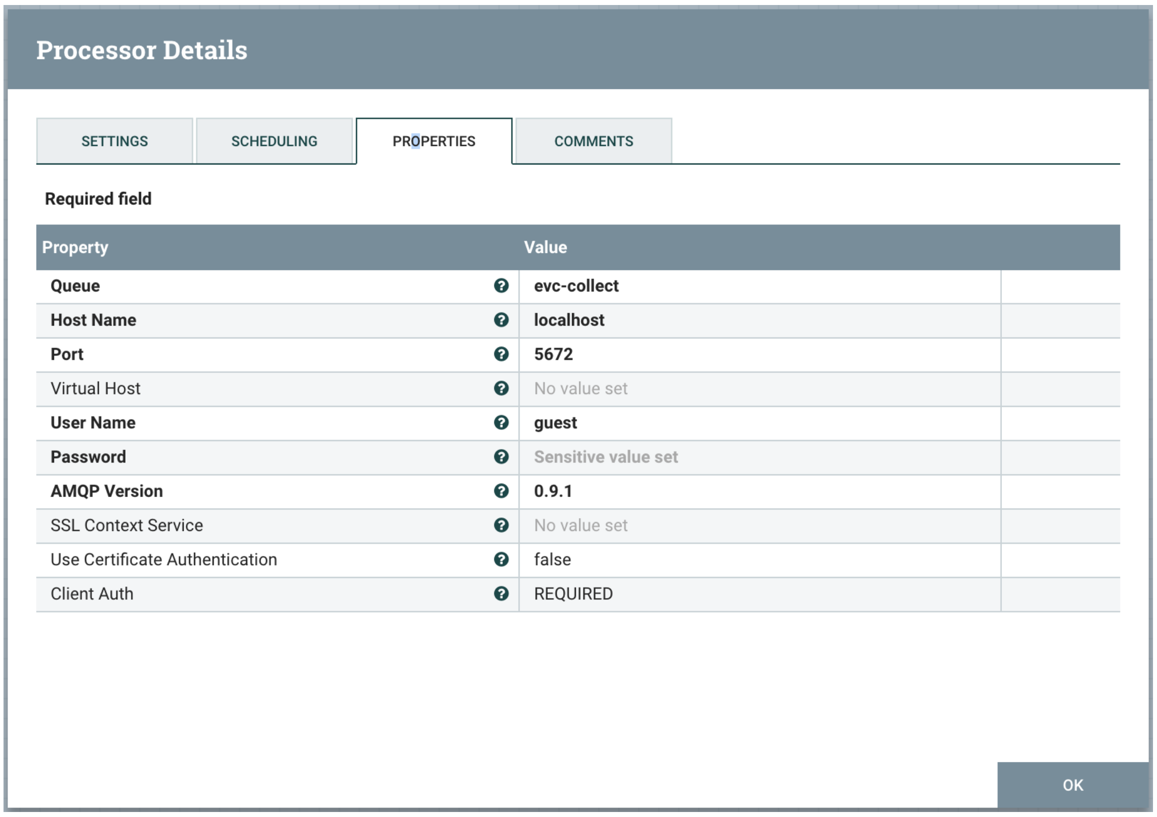
4.4.2 Events Queue Listener

In a nutshell, Footstep's Consumer & Parser Module is actually implemented and organized as groups of processors in NiFi. All start with Events Queue Listener, a NiFi processor responsible for consuming the event queue presented in Section 4.3. Consumer processor is a NiFi processor designed to consume queues such as those provided by RabbitMQ, among other types of messaging queues. Figure 14 shows the processor configuration properties as RabbitMQ queue name, host and port to connect, and authorization data.

The Events Queue Listener processor is connected to a processor group responsible for parsing events to the UsaGraph model format, as depicted in Figure 15.

4.4.3 Events Parser

The Events Queue Listener processor is connected to the Parser process group, which contains several processors responsible for reading and parsing events to extract user, page, event type and HTML elements, and finally translate them to UsaGraph nodes and vertexes. All parsing and text transformations are performed by specific processors


Processor Details

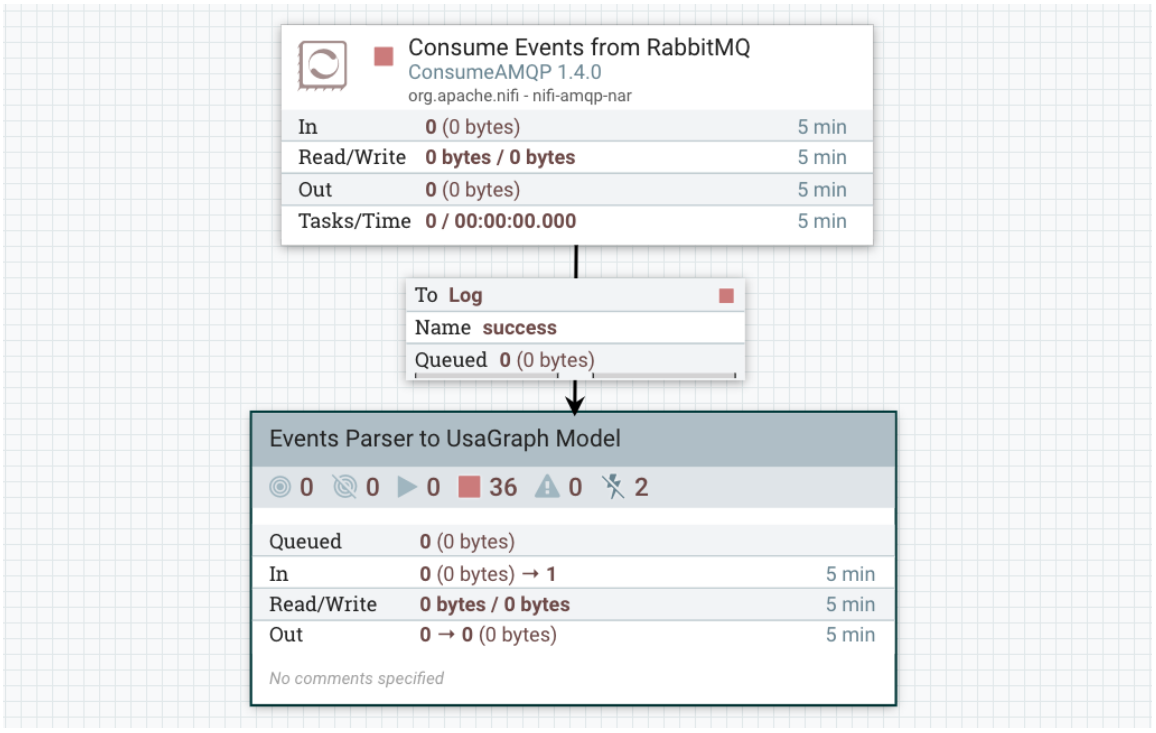
SETTINGS | SCHEDULING | PROPERTIES | COMMENTS

Required field

Property	Value
Queue	**evc-collect**
Host Name	**localhost**
Port	**5672**
Virtual Host	No value set
User Name	**guest**
Password	Sensitive value set
AMQP Version	**0.9.1**
SSL Context Service	No value set
Use Certificate Authentication	false
Client Auth	REQUIRED

OK

Figure 14 – Events Queue Listener Processor.


Consume Events from RabbitMQ
ConsumeAMQP 1.4.0
org.apache.nifi - nifi-amqp-nar

In	**0** (0 bytes)	5 min
Read/Write	**0 bytes / 0 bytes**	5 min
Out	**0** (0 bytes)	5 min
Tasks/Time	**0 / 00:00:00.000**	5 min

To **Log**
Name **success**
Queued **0** (0 bytes)

Events Parser to UsaGraph Model
0 0 0 36 0 2

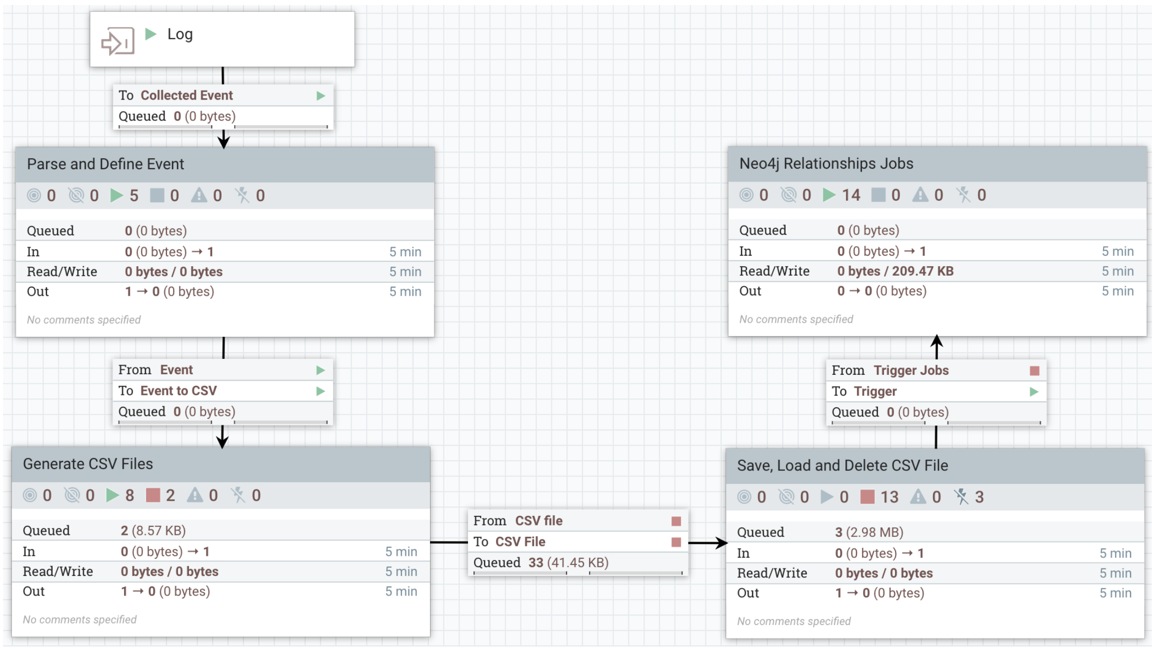
Queued	**0** (0 bytes)	
In	**0** (0 bytes) → **1**	5 min
Read/Write	**0 bytes / 0 bytes**	5 min
Out	**0** → **0** (0 bytes)	5 min

No comments specified

Figure 15 – Consumer Processor and Parser Process Group.

which operate over text using Regular Expressions to extract specific parts according to defined patterns. Figure 16 shows the processors inside Parser process group:

- Parse and Define Event: responsible to read event log, identify the UsaGraph entities (event type, user, page, elements);
- Generate CSV Files: prepare the parsed events to insert them into Graph Database;
- Save, Load and Delete CSV File: processor group responsible to call `LOAD CSV` commands in Neo4j using CSV files containing parsed events;
- Neo4j Relationship Jobs: processor group containing several jobs to ensure UsaGraph relationships between all entities extracted from event logs.



Parse and Define Event
0 0 ▶ 5 ■ 0 ⚠ 0 0
Queued 0 (0 bytes)
In 0 (0 bytes) → 1 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 1 → 0 (0 bytes) 5 min
No comments specified

From Event
To Event to CSV
Queued 0 (0 bytes)

Generate CSV Files
0 0 ▶ 8 ■ 2 ⚠ 0 0
Queued 2 (8.57 KB)
In 0 (0 bytes) → 1 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 1 → 0 (0 bytes) 5 min
No comments specified

From CSV file
To CSV File
Queued 33 (41.45 KB)

Save, Load and Delete CSV File
0 0 ▶ 0 ■ 13 ⚠ 0 3
Queued 3 (2.98 MB)
In 0 (0 bytes) → 1 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 1 → 0 (0 bytes) 5 min
No comments specified

From Trigger Jobs
To Trigger
Queued 0 (0 bytes)

Neo4j Relationships Jobs
0 0 ▶ 14 ■ 0 ⚠ 0 0
Queued 0 (0 bytes)
In 0 (0 bytes) → 1 5 min
Read/Write 0 bytes / 209.47 KB 5 min
Out 0 → 0 (0 bytes) 5 min
No comments specified

Figure 16 – Parser Process Group.

4.4.3.1 Parse and Define Event process group

Section 4.3.2 described how event log is built as a pre-formatted text containing all relevant data to translate an event triggered by the user to the UsaGraph model. The Footstep JavaScript library is responsible for encoding events in this format. Thus, a further step is required to decode event logs, extracting all data to translate to the UsaGraph model.

Inside the Parse and Define Event process group, as depicted in Figure 17, there are three processors, named as *Extract Event Parameters*, *Decode Page* and *Set Page, Host, Date and Elements* processors.

The first processor, *Extract Event Parameters*, is responsible for reading the event log, extracting all parameters defined in Section 4.3.2 through regular expressions to NiFi *FlowFiles* attributes, as Figure 18 shows.

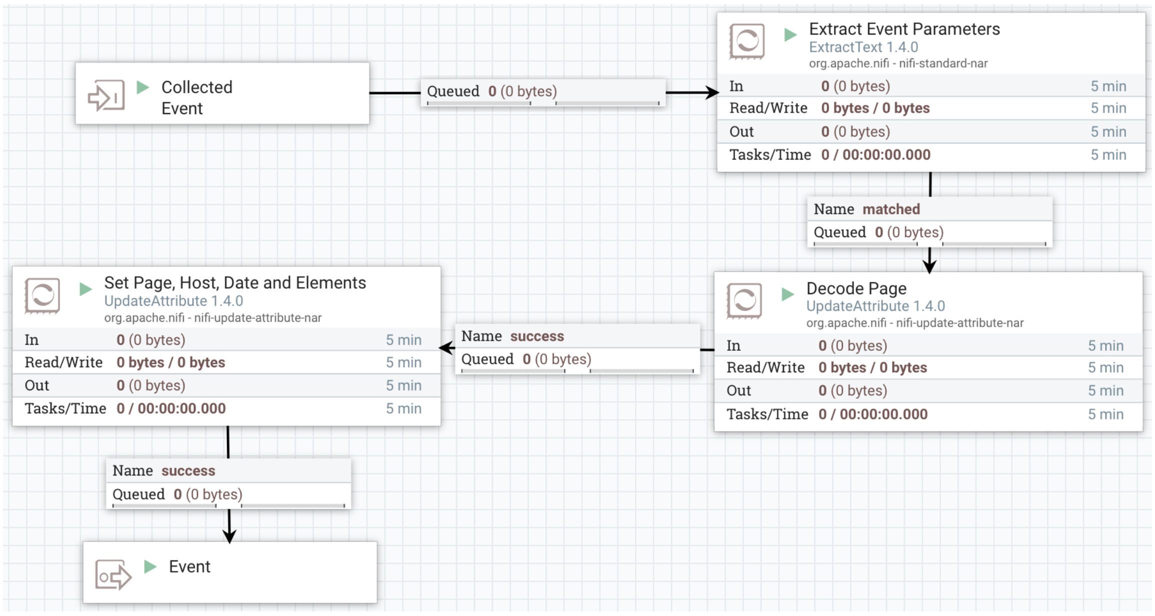
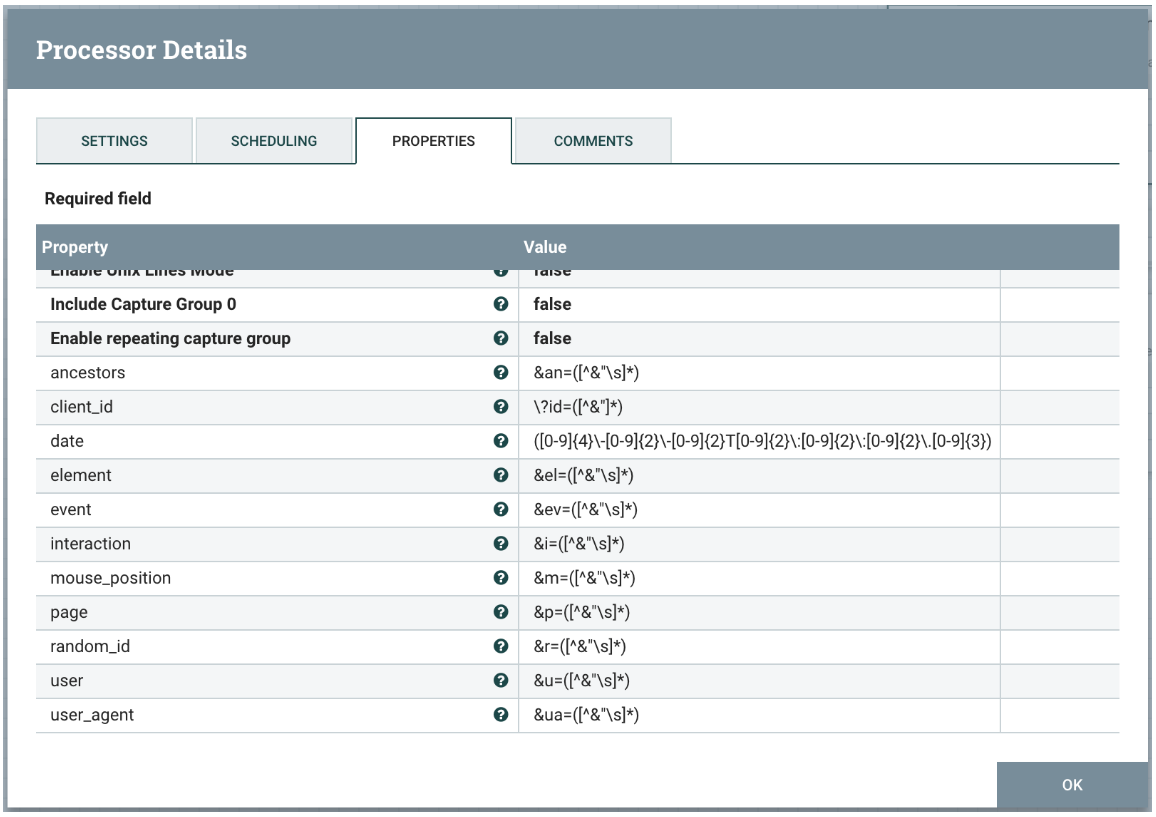

Collected Event
Queued 0 (0 bytes)
Extract Event Parameters
ExtractText 1.4.0
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name matched
Queued 0 (0 bytes)
Decode Page
UpdateAttribute 1.4.0
org.apache.nifi - nifi-update-attribute-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 0 (0 bytes)
Set Page, Host, Date and Elements
UpdateAttribute 1.4.0
org.apache.nifi - nifi-update-attribute-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 0 (0 bytes)
Event

Figure 17 – Parse and Define Event process group.



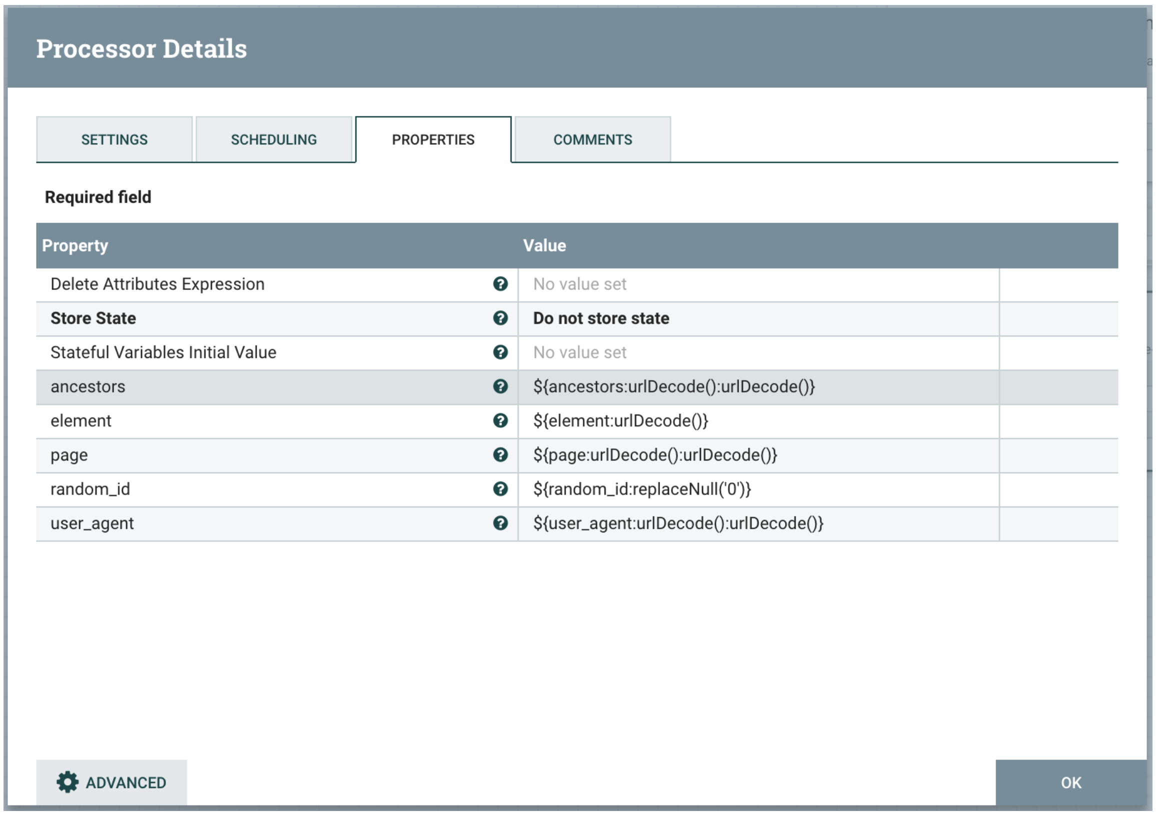
Required field

Property	Value
Enable Unix Lines Mode	false
Include Capture Group 0	**false**
Enable repeating capture group	**false**
ancestors	&an=([^&"\s]*)
client_id	\?id=([^&"]*)
date	([0-9]{4}\-[0-9]{2}\-[0-9]{2}T[0-9]{2}\:[0-9]{2}\:[0-9]{2}\.[0-9]{3})
element	&el=([^&"\s]*)
event	&ev=([^&"\s]*)
interaction	&i=([^&"\s]*)
mouse_position	&m=([^&"\s]*)
page	&p=([^&"\s]*)
random_id	&r=([^&"\s]*)
user	&u=([^&"\s]*)
user_agent	&ua=([^&"\s]*)

Figure 18 – Extracting parameters from event log using regular expressions.

The second processor, *Decode Page*, decodes and cleans attributes as depicted in Figure 19. These parameters need to be decoded due to the fact they were previously encoded to be sent through `/collect` parameters to Events Listener Service.

And the third processor, *Set Page, Host, Date and Elements*, defines important attributes to be further translated to UsaGraph model, as Figure 20 shows, such as



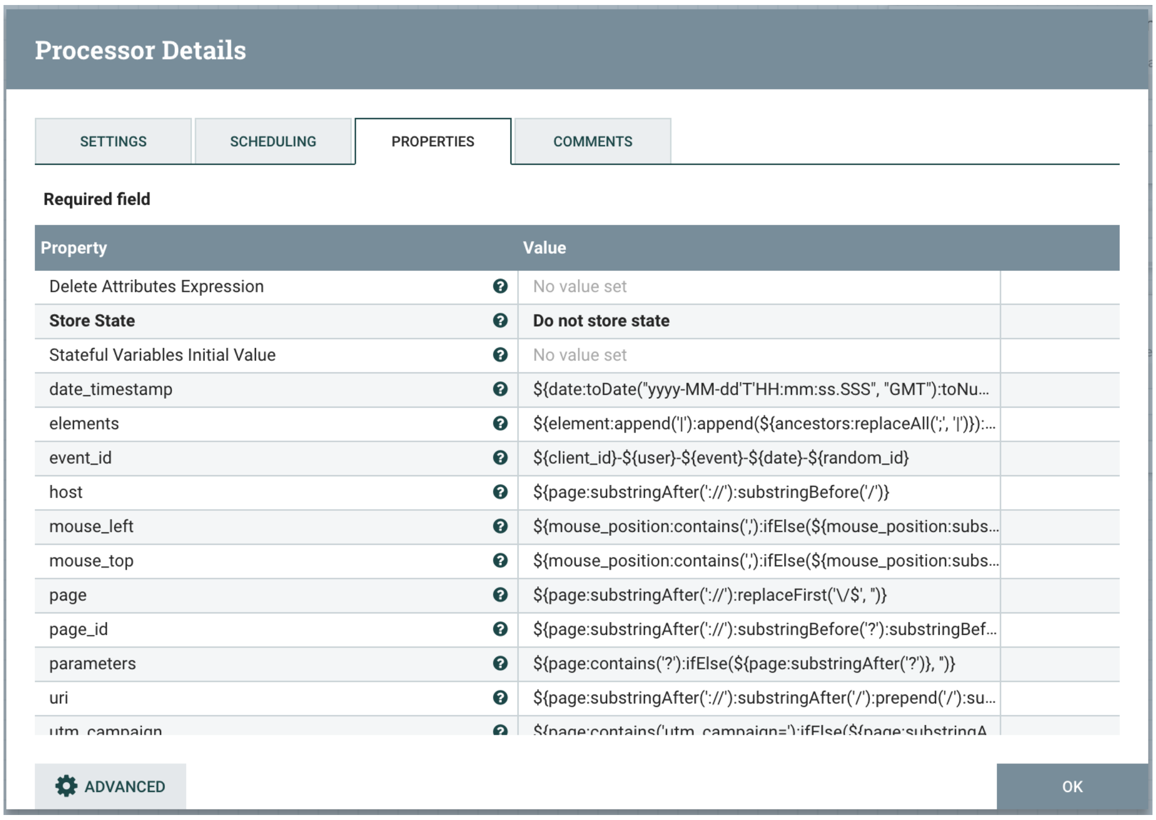
Processor Details

Required field

Property	Value
Delete Attributes Expression	No value set
Store State	**Do not store state**
Stateful Variables Initial Value	No value set
ancestors	${ancestors:urlDecode():urlDecode()}
element	${element:urlDecode()}
page	${page:urlDecode():urlDecode()}
random_id	${random_id:replaceNull('0')}
user_agent	${user_agent:urlDecode():urlDecode()}

Figure 19 – Decoding parameters from extracted parameters.

`page_id`, `host`, `event_id`, `date_timestamp` and others attributes.



Processor Details

Required field

Property	Value
Delete Attributes Expression	No value set
Store State	**Do not store state**
Stateful Variables Initial Value	No value set
date_timestamp	${date:toDate("yyyy-MM-dd'T'HH:mm:ss.SSS", "GMT"):toNu...
elements	${element:append('\|'):append(${ancestors:replaceAll(';', '|')}):...
event_id	${client_id}-${user}-${event}-${date}-${random_id}
host	${page:substringAfter('://'):substringBefore('/')}
mouse_left	${mouse_position:contains(','):ifElse(${mouse_position:subs...
mouse_top	${mouse_position:contains(','):ifElse(${mouse_position:subs...
page	${page:substringAfter('://'):replaceFirst('\/$', '')}
page_id	${page:substringAfter('://'):substringBefore('?'):substringBef...
parameters	${page:contains('?'):ifElse(${page:substringAfter('?')}, '')}
uri	${page:substringAfter('://'):substringAfter('/'):prepend('/'):su...
utm_campaign	${page:contains('utm_campaign='):ifElse(${page:substringA

Figure 20 – Refining and defining parameters from decoded parameters.

4.4.3.2 Generate CSV Files processor group

A CSV file contains several fields separated by comma character, one record per line. As Neo4j Import Guide (NEO4J, 2007a) recommends, one of the fastest ways to import data into a Neo4J running database is by using `LOAD CSV` command to persist nodes and relationships using CSV files present in Neo4j import folder.

Figure 21 presents Generate CSV Files processor group containing processors responsible for organizing *FlowFiles* attributes previously extracted into CSV files in order to persist them into Neo4j according to the UsaGraph model. This process group also detects events triggered on elements in order to organize elements and its ancestors to maintain the hierarchy according to the DOM structure, to be further persisted into Neo4j as `Element` nodes.

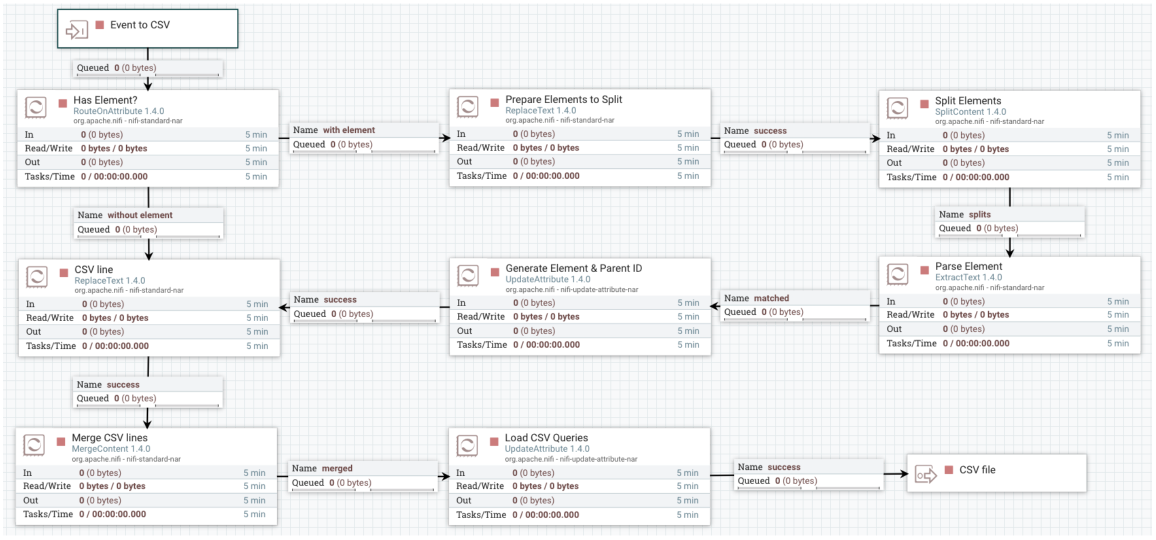


Figure 21 – Generate CSV Files processor group.

4.4.3.3 Save, Load and Delete CSV Files processor group

This process group contains processors responsible for persisting CSV files into Neo4j import group, for executing Neo4j `LOAD CSV` command and, finally, for deleting CSV files after the import process.

4.4.3.4 Neo4j Relationships Jobs processor group

A final step is required to maintain consistency between imported nodes in Save, Load and Delete CSV Files processor group. Each CSV line corresponds to nodes to be inserted, but the relationships are part of the UsaGraph model. This process group calls jobs responsible for joining `Event` nodes belonging to the same user's interaction using `NEXT` relationships. There are also jobs connecting `Element` nodes to respective `Element` parent node, connecting `Page` nodes to `PageLabel` nodes and finally connecting `Element` to `ElementLabel` nodes.

4.4.4 Graph Database

UsaGraph model is built upon graph theory, therefore it is straightforward to use a graph database to store events translated to nodes and vertexes. There are many available technologies and databases designed to store graphs. Footstep stores its data into Neo4J Neo4j (2007b), an open source native graph database, largely used by companies. Neo4J is a mature solution in terms of database platform, supporting transactional applications and graph analytics, providing drivers and APIs to developers, as well as tools to monitor and manage databases, as shown in Figure 22.

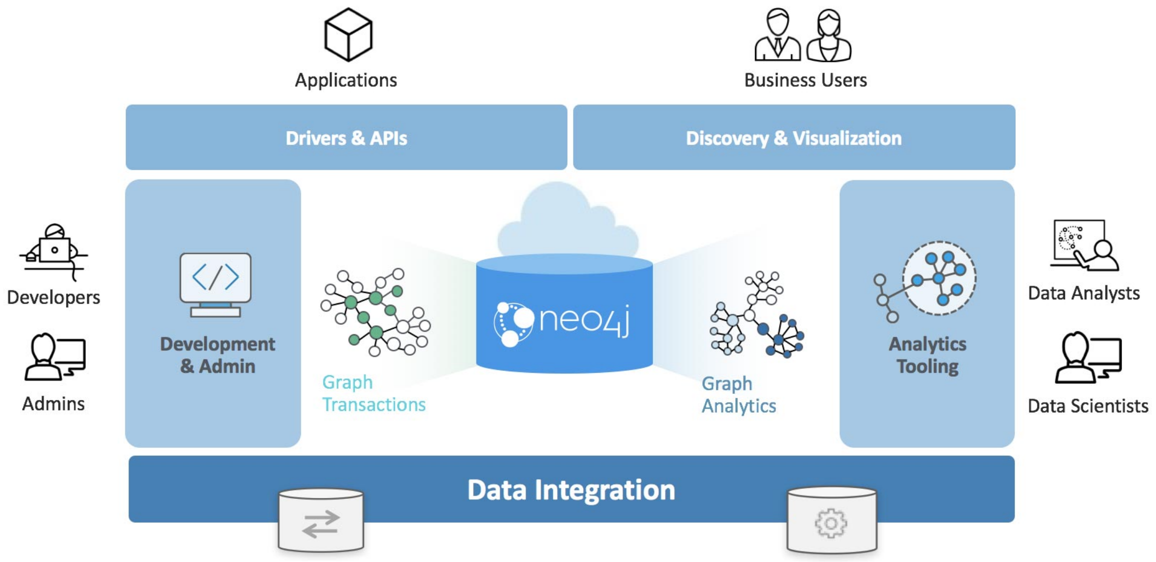


Figure 22 – Neo4j's Graph Platform (NEO4J'S GRAPH PLATFORM, 2007).

According to its developers, Neo4J is as a *Labeled Property Graph Model*, as Figure 23 illustrates. Nodes are the main data elements, which can have properties, stored as key-value pairs. Relationships connecting two nodes are directed and can also store properties. The term *Labeled Graph Model* comes from the fact that Neo4J uses labels to identify and separate nodes and relationships into sets. The term *Property Graph Model* comes from its ability to store properties into nodes and relationships, and also from the possibility to add indexes and constraints to properties.

Neo4J provides a graph query language, called Cypher, described as a vendor-neutral open graph query language with a familiar syntax that provides a readable way to match patterns of nodes and relationships within graph datasets[1]. Cypher was available exclusively for Neo4J, but in 2017 Neo4J Inc. decided to open source Cypher aiming to leverage it as the SQL for graphs[2]. Cypher has a specific way to query for nodes and relationships, as Figure 24 illustrates.

[1] Cypher, The Graph Query Language <https://neo4j.com/cypher-graph-query-language/>
[2] The Open Cypher Project <http://www.opencypher.org/about/>

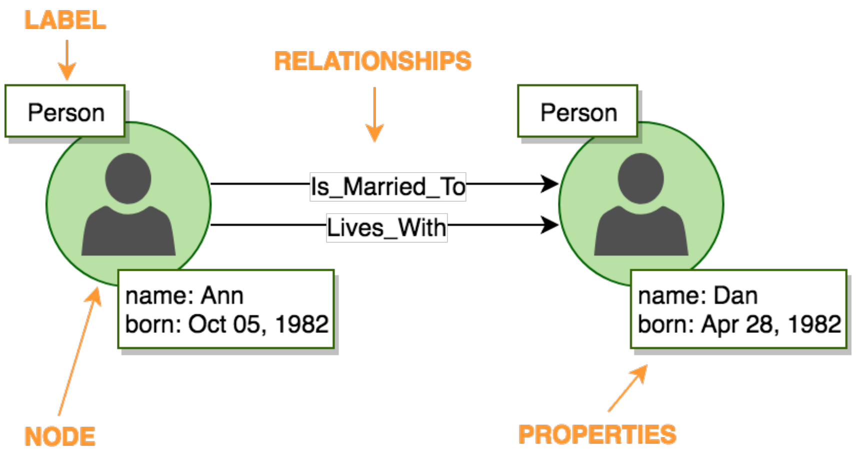


Figure 23 – Neo4j's Labeled Property Graph Model.

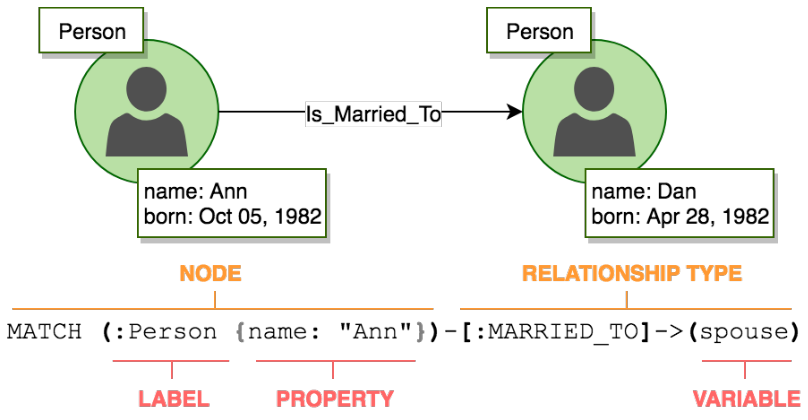


Figure 24 – Cypher syntax for matching nodes and relationships.

Neo4J has some different ways to insert nodes and relationships, such as `CREATE` Cypher command to insert nodes, `MERGE` Cypher command to insert or update nodes, `LOAD CSV` Cypher command to load nodes from CSV files, and there is also a Neo4J import tool to create a graph database from scratch. Figure 21 presented the second part of events parsing process, in which all parsed event data is saved as a CSV file.

4.5 Generalization Module

One of the limitations of our approach is the lack of ability to infer pages' similarity automatically. It is required an explicit manual intervention to make possible to say that a given page A and another page B belongs to the same page set, or, using UsaGraph terms, that two `Page` nodes connect to the same `PageLabel` node. This is possible by configuring patterns that match `Page` URL's, mapping them to a specific `PageLabel`. Those patterns may contain static or dynamic parts in order to provide flexibility to the URL matching algorithm.

Listing 4.7 illustrates the procedure of grouping pages into page sets. Giving an hypothetical website with products and services pages, if the generalization is defined by how many parts an URL is divided, three `PageLabel` would be created: Home page, One part pages and Two part pages. In this case, the generalization is made by simply counting URL's parts, thus, it is not necessary to define which dynamic parts belong to which page type. By convention, URL parts defined with asterisk accepts anything.

Listing 4.7 – URL generalization by URL's parts count

```
https://www.website.com/                        <- Home page
https://www.website.com/products                <- Products page
https://www.website.com/products/product-A      <- Product A page
https://www.website.com/products/product-B      <- Product B page
https://www.website.com/services                <- Services page
https://www.website.com/services/service-c      <- Service C page
https://www.website.com/services/service-D      <- Service D page

Generalizing by URL's parts:
https://www.website.com/                        <- Home page
https://www.website.com/*                       <- One part pages:
                                                   Products, Services
https://www.website.com/*/*                     <- Two part pages:
                                                   Products A, B;
                                                   Services C, D
```

Considering another hypothetical scenario, in which the same URLs presented in Listing 4.7 are used, but now the aim is to identify products and services pages separately, no matter whether the page is the list of products or the product page itself. Listing 4.8 illustrates how to separate pages into products and services pages, simply by checking the first part of the URL. In this case, it is necessary to define two patterns to identify the same page type.

Listing 4.8 – URL generalization by URL's parts count

```
Generalizing by products and services pages:
https://www.website.com/                        <- Home page

1: https://www.website.com/products             <- Products pages:
2: https://www.website.com/products/*              Products, Product A,
                                                   B and others

1: https://www.website.com/services             <- Services pages:
2: https://www.website.com/services/*              Services, Service C,
                                                   D and others
```

When the website presents more dynamic URLs, the generalization procedure becomes more complicated. Listing 4.9 adds different URLs to the previous examples, a login and register pages, and three pages that represent something about locations. Now, the requirement is to separate pages into these page types: Home, Users (login and register), Locations (location-1, location-2 and location-3), Listings (products and services), Product page, Service Page and Unmapped pages. This is possible by using keywords that define the permitted parts, such as `USER-PART` which accepts only `login` and `register`. By convention, URL parts surrounded by curly braces represent keywords and must define its possible values. To match Unmapped pages, another convention is defined by negating keywords with the exclamation mark, as shown in Listing 4.9. The mechanism of matching pages patterns must compare those pages starting with the most restrictive URL pattern, otherwise asterisk would match all pages and label them.

Listing 4.9 – URL generalization by URL's parts count

```
https://www.website.com/                      <- Home page
https://www.website.com/login                 <- Login page
https://www.website.com/register              <- Register page
https://www.website.com/location-1            <- Location 1 page
https://www.website.com/location-2            <- Location 2 page
https://www.website.com/location-3            <- Location 3 page
https://www.website.com/products              <- Products page
https://www.website.com/products/product-A    <- Product A page
https://www.website.com/products/product-B    <- Product B page
https://www.website.com/services              <- Services page
https://www.website.com/services/service-c    <- Service C page
https://www.website.com/services/service-D    <- Service D page

Generalizing by requirements:
https://www.website.com/                      <- Home page
https://www.website.com/{USER-PART}           <- User pages:
                                                 login and register
https://www.website.com/{LOCATION-PART}       <- Location pages:
                                                 location-1, location-2
                                                 and location-3
https://www.website.com/{LISTINGS-PART}       <- Listings pages:
                                                 products and services
https://www.website.com/products/*            <- Product page:
                                                 product-A, product-B
                                                 and others
https://www.website.com/services/*            <- Services page:
                                                 service-C, service-D
                                                 and others

Unmapped pages
https://www.website.com/!USER_PART|!LOCATION-PART|!LISTINGS-PART
https://www.website.com/!USER_PART|!LOCATION-PART|!LISTINGS-PART/*

URL permitted parts:
- USER-PART: login and register
- LOCATION-PART: location-1, location-2, location-3
- LISTING-PART: products and services
```

4.5.1 Page Label Configuration

The Generalization Module stores and reads Page Label's configurations in a document-based database called MongoDB[3]. MongoDB stores data in a flexible way, using JSON-like documents[4] into collections. Footstep expects only two types of documents to

[3] MongoDB [online] <https://www.mongodb.com/>
[4] JavaScript Object Notation <https://www.json.org/>

perform generalization: page label and pattern documents, each one stored in a different MongoDB collection, as Listing 4.10 illustrates.

Listing 4.10 – Basic signature of Footstep collections for pages generalization

```
Collection page_labels:
{
  "id" : "NAME",
  "urls" : [
    "URL-1",
    "URL-2"
  ]
}

Collection patterns:
{
  "id" : "KEYWORD-NAME",
  "patterns" : [
    "PATTERN-1",
    "PATTERN-2"
  ]
}
```

Using the configuration described in Listing 4.9 to operate in Footstep, documents must be inserted into the MongoDB collections `page_labels` and `patterns`, as shown in Listing 4.11. Therefore, patterns `USER-PART`, `LOCATIONS-PART` and `LISTINGS-PART` and page labels `HOME-PAGE`, `USER-PAGE`, `LOCATIONS-PAGE`, `LISTINGS-PAGE`, `PRODUCT-PAGE`, `SERVICE-PAGE` and `UNMAPPED-PAGE` must be declared and inserted into collections.

Listing 4.11 – Configuration of hypothetical scenario of generalization

```
Collection patterns:
{
  "id" : "USER-PART",
  "patterns" : [
    "login",
    "register"
  ]
}
{
  "id" : "LOCATIONS-PART",
  "patterns" : [
    "location-1",
    "location-2",
    "location-3"
  ]
}
{
  "id" : "LISTINGS-PART",
  "patterns" : [
    "products",
    "services"
  ]
}

Collection page_labels:
{
  "id" : "HOME-PAGE",
  "urls" : [
    "www.website.com",
    "www.website.com/"
  ]
}
{
  "id" : "USER-PAGE",
  "urls" : [
    "www.website.com/{USER-PART}"
  ]
}
{
  "id" : "LOCATIONS-PAGE",
  "urls" : [
    "www.website.com/{LOCATIONS-PART}"
  ]
}
{
  "id" : "LISTINGS-PAGE",
  "urls" : [
    "www.website.com/{LISTINGS-PART}"
  ]
}
{
  "id" : "PRODUCT-PAGE",
  "urls" : [
    "www.website.com/product/*"
  ]
}
{
  "id" : "SERVICE-PAGE",
  "urls" : [
    "www.website.com/service/*"
  ]
}
{
  "id" : "UNMAPPED-PAGE",
  "urls" : [
    "www.website.com/{!USER_PART}{!LOCATION-PART}{!LISTINGS-PART}",
    "www.website.com/{!USER_PART}{!LOCATION-PART}{!LISTINGS-PART}/*"
  ]
}
```

4.5.2 Pattern Extraction

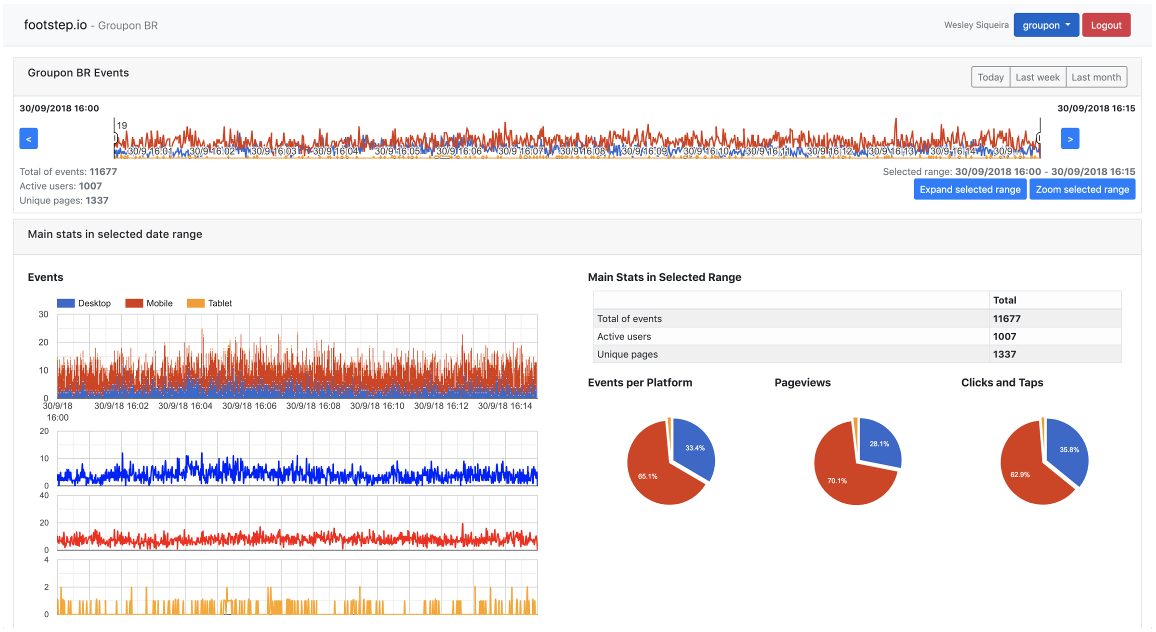
As pages are classified using Page Label configuration patterns, a second step in Generalization Module comes into action: the element generalization. Section 3.5.1 demonstrated in Figure 8 how different elements in different pages can be generalized and represented by a `ElementLabel` node. This generalization could be made regardless of `PageLabel` generalization, but the computational overhead could be significant. In other words, elements generalization without `PageLabel` would compare elements of different types of pages, for example, comparing elements from register page with elements from products listings page. The odds of an efficient element match among different types of pages are lower than element matching of similar pages. So, pages generalization aids elements generalization by reducing the spectrum of elements' comparisons that might belong to the same type of page.

Element generalization does not require an outside configuration, since only elements' type and attributes are used to make comparisons between them. This is also made by a NiFi processor, which queries Neo4J for unmatched `Element` nodes, compares them and create `ElementLabel` nodes in case of matching.

4.6 Analytics Module

The Analytics Module is a web application designed to support website administrators, allowing them to visualize usage data in real time. Among its features, the Analytics Module provides dashboards to analyze (i) the usage flow in pages, (ii) the audience on page elements, (iii) the conversion in pages and elements. The data analytics provided by this module separates the interaction according to the used device: desktop, smartphone (mobile) and tablet. This module also offers facilities for the configuration of Page Labels according to the URL policy used in the analyzed website.

Figure 25 depicts Footstep's main dashboard presenting a histogram of events and valuable statistics about number of events, users and pages, as well as the distribution of events among the the supported device types.



Total of events: **11677**
Active users: **1007**
Unique pages: **1337**

Selected range: **30/09/2018 16:00 - 30/09/2018 16:15**

Main Stats in Selected Range

	Total
Total of events	11677
Active users	1007
Unique pages	1337

Figure 25 – Analytics dashboard.

Footstep's main dashboard also presents the page audience information, as Figure 26 shows. Each box presents data about a Page Label, showing the amount of active users, those whose last interaction occurred in this Page Label, and total users, which represents the total of users that accessed this Page Label during the collection time.

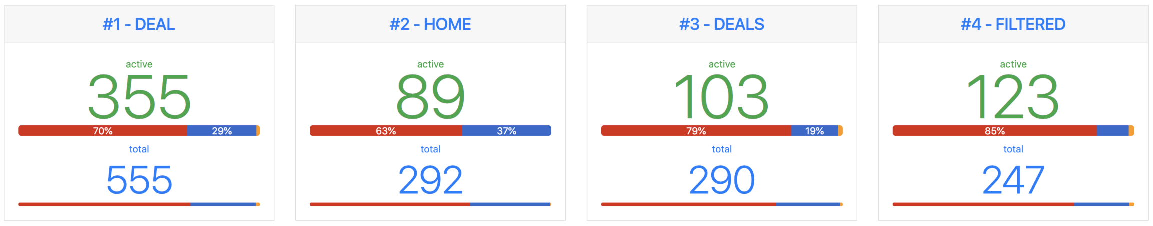


Figure 26 – Page audience dashboard.

The Analytics Module also provides element audience, navigation flow and conversion dashboards, where website administrators are able to keep track of users and their actions. Element audience dashboard shows the total of events triggered in elements, in a specific page and period of analysis, as Figure 27 shows.

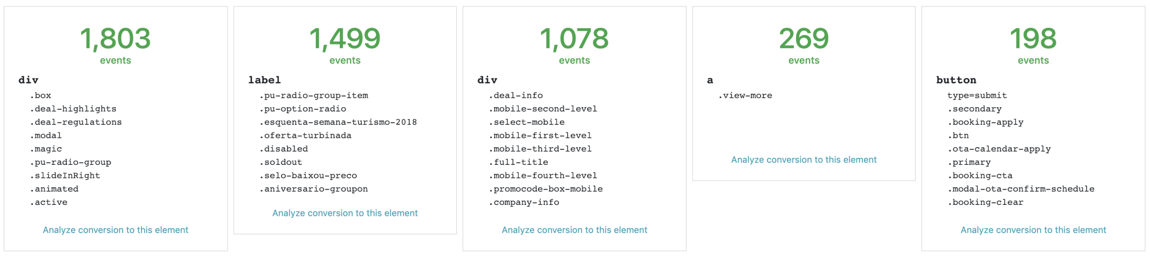


Figure 27 – Element audience dashboard.

Navigation flow dashboard shows how pages are connected to other pages and the flow of navigation between them, during a specific period of time, as depicted in Figure 28.

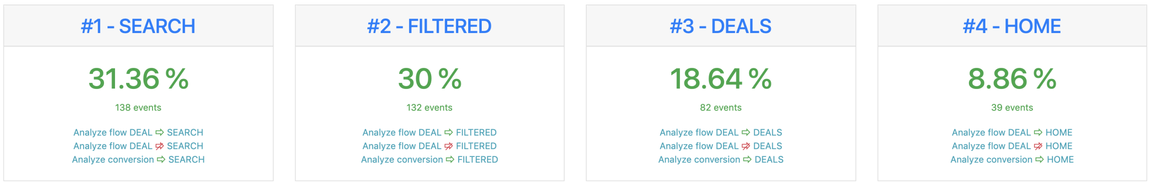
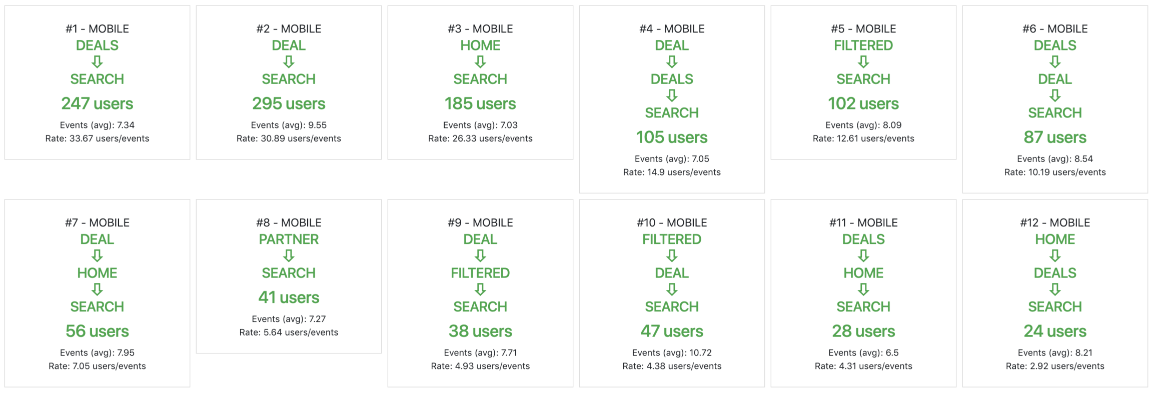

#1 - SEARCH
31.36 %
138 events
Analyze flow DEAL ⇨ SEARCH
Analyze flow DEAL ⇨ SEARCH
Analyze conversion ⇨ SEARCH
#2 - FILTERED
30 %
132 events
Analyze flow DEAL ⇨ FILTERED
Analyze flow DEAL ⇨ FILTERED
Analyze conversion ⇨ FILTERED
#3 - DEALS
18.64 %
82 events
Analyze flow DEAL ⇨ DEALS
Analyze flow DEAL ⇨ DEALS
Analyze conversion ⇨ DEALS
#4 - HOME
8.86 %
39 events
Analyze flow DEAL ⇨ HOME
Analyze flow DEAL ⇨ HOME
Analyze conversion ⇨ HOME

Figure 28 – Page flow dashboard.

Finally, the conversion dashboard shows how users convert to specific page or element. In other words, this dashboard shows navigation paths made by users to reach some point (page or element), the total of users that followed this navigation path, the total of events spent to reach the conversion point as the rate of users per events. Figures 29 and 30 show, respectively, page and element conversion dashboards.



#1 - MOBILE
DEALS
⇩
SEARCH
247 users
Events (avg): 7.34
Rate: 33.67 users/events

#2 - MOBILE
DEAL
⇩
SEARCH
295 users
Events (avg): 9.55
Rate: 30.89 users/events

#3 - MOBILE
HOME
⇩
SEARCH
185 users
Events (avg): 7.03
Rate: 26.33 users/events

#4 - MOBILE
DEAL
⇩
DEALS
⇩
SEARCH
105 users
Events (avg): 7.05
Rate: 14.9 users/events

#5 - MOBILE
FILTERED
⇩
SEARCH
102 users
Events (avg): 8.09
Rate: 12.61 users/events

#6 - MOBILE
DEALS
⇩
DEAL
⇩
SEARCH
87 users
Events (avg): 8.54
Rate: 10.19 users/events

#7 - MOBILE
DEAL
⇩
HOME
⇩
SEARCH
56 users
Events (avg): 7.95
Rate: 7.05 users/events

#8 - MOBILE
PARTNER
⇩
SEARCH
41 users
Events (avg): 7.27
Rate: 5.64 users/events

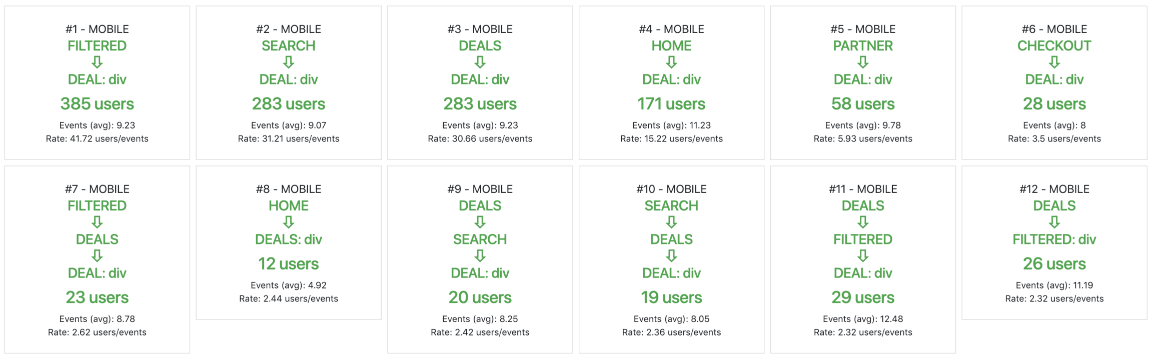
#9 - MOBILE
DEAL
⇩
FILTERED
⇩
SEARCH
38 users
Events (avg): 7.71
Rate: 4.93 users/events

#10 - MOBILE
FILTERED
⇩
DEAL
⇩
SEARCH
47 users
Events (avg): 10.72
Rate: 4.38 users/events

#11 - MOBILE
DEALS
⇩
HOME
⇩
SEARCH
28 users
Events (avg): 6.5
Rate: 4.31 users/events

#12 - MOBILE
HOME
⇩
DEALS
⇩
SEARCH
24 users
Events (avg): 8.21
Rate: 2.92 users/events

Figure 29 – Page conversion dashboard.



#1 - MOBILE
FILTERED
⇩
DEAL: div
385 users
Events (avg): 9.23
Rate: 41.72 users/events

#2 - MOBILE
SEARCH
⇩
DEAL: div
283 users
Events (avg): 9.07
Rate: 31.21 users/events

#3 - MOBILE
DEALS
⇩
DEAL: div
283 users
Events (avg): 9.23
Rate: 30.66 users/events

#4 - MOBILE
HOME
⇩
DEAL: div
171 users
Events (avg): 11.23
Rate: 15.22 users/events

#5 - MOBILE
PARTNER
⇩
DEAL: div
58 users
Events (avg): 9.78
Rate: 5.93 users/events

#6 - MOBILE
CHECKOUT
⇩
DEAL: div
28 users
Events (avg): 8
Rate: 3.5 users/events

#7 - MOBILE
FILTERED
⇩
DEALS
⇩
DEAL: div
23 users
Events (avg): 8.78
Rate: 2.62 users/events

#8 - MOBILE
HOME
⇩
DEALS: div
12 users
Events (avg): 4.92
Rate: 2.44 users/events

#9 - MOBILE
DEALS
⇩
SEARCH
⇩
DEAL: div
20 users
Events (avg): 8.25
Rate: 2.42 users/events

#10 - MOBILE
SEARCH
⇩
DEALS
⇩
DEAL: div
19 users
Events (avg): 8.05
Rate: 2.36 users/events

#11 - MOBILE
DEALS
⇩
FILTERED
⇩
DEAL: div
29 users
Events (avg): 12.48
Rate: 2.32 users/events

#12 - MOBILE
DEALS
⇩
FILTERED: div
26 users
Events (avg): 11.19
Rate: 2.32 users/events

Figure 30 – Element conversion dashboard.

4.7 Final Remarks

This chapter presented the Footstep System, an approach based on the UsaGraph data model, designed to collect users' interactions among websites in order to extract useful and detailed information about navigation at DOM elements level.

Footstep System is divided in four specialized modules, decoupling event collection, processing, persisting and analyzing phases. Each module in autonomous, bringing scalability to the system.

Chapter 5 shows the experiments made do validate our approach by exploiting different kinds of websites and expatiating the collected data and results.

5 Experiments

5.1 Initial considerations

The main contribution of Footstep system is the analysis of web usage considering the interaction with HTML elements and its ancestors, by exploiting its graph data model, UsaGraph. Section 4.2 described the proposed architecture for automated event collection, consumption and processing in websites, being an union of tiny, less intrusive JavaScript library, and asynchronous, fast and reliable event processing system.

This chapter presents experiments with real web applications that demonstrate Footstep's main features, highlighting how these features are useful to support the administration of e-commerce websites. Footstep provides, in real time, dashboards to monitor website's audience, showing flow diagrams considering pages and HTML elements inside pages. This features allow website administrators to detect anomalies and understand the behavior of users. Footstep also provides page heat maps and a customization feature to define and measure conversion points on e-commerce applications.

To demonstrate Footstep's features, experiments were performed in two different types of e-business: e-commerce marketplace and transportation (load/freight) marketplace. Three websites were selected to collect, extract and analyze web usage data:

- E-commerce marketplaces:
  - Peixe Urbano (2010a) (Brazil)
  - Groupon Brazil (2017) (Brazil)
- Transportation marketplace:
  - TruckPad (2015)

Each business has different goals: an e-commerce marketplace aims at selling goods to their users, while the transportation marketplace aims at supporting the supplier/customer relationship, connecting truckers to truckloads. Even so, both business websites are designed to lead their users from a given point inside the website to another, that is, both have specific conversion targets. The experiments performed in Peixe Urbano, Groupon and TruckPad were accompanied by some of its software engineers, who were in charge of installing Footstep's JavaScript library, as well as defining the different types of pages to help the creation of *PageLagel*'s according to the specificities of each business.

The three experiments presented in this chapter aims at exploring the following features:

- Pages and elements audience: the frequency of access of pages and elements over a specific period of time;
- Navigation flow: The flow between pages and elements;
- Conversion points: How users convert to pages and elements.

5.2 Peixe Urbano

Peixe Urbano is the largest local e-commerce company in Brazil, with more than 30 million users. It was founded in early 2010 and elected as "The Best International Startup of the Year" by Crunchies Awards (PEIXE URBANO, 2010b). Peixe Urbano contains thousands of local offers in gastronomy, entertainment, tourism, goods, etc. Several thousands of users visit Peixe Urbano every day.

The installation of Footstep JavaScript library in Peixe Urbano did not require any development by engineer team due to the fact that Peixe Urbano uses Google Tag Manager (2012), a tag management system that allows updating tracking codes and related code fragments collectively known as tags on a web application (GOOGLE TAG MANAGER, 2012). In other words, Google Tag Manager allows the injection of Footstep JavaScript library on-the-fly, with no deploying needed.

Peixe Urbano was instrumented by Footstep JavaScript library in September 30, 2018, from 3:45 PM to 5:45 PM, Brasilia Time Zone. During this period, a total of 281,882 events were collected, containing 27,996 unique users, 13,896 visited pages and 390,359 interaction with HTML elements. The Consumer & Parser module presented no significant increase of processing, consuming the event queue very fast.

According to the engineers of Peixe Urbano, among all 13,896 visited pages, only 28 page types were identified. The most important page types for the analysis of Peixe Urbano are:

- HOME: home page containing listings of offers;
- LOGIN: login page;
- REGISTER: register page;
- RECOVER_PASSWORD: recover password page;
- DEALS: listing of offers per city;

- FILTERED: listing of offers per city, filtered by one or more categories;
- SEARCH: listing of offers per city, filtered by query search;
- PARTNER: listing of offers per partner;
- DEAL: page of a specific offer;
- CHECKOUT: shopping cart and checkout page;
- PROCESSING: purchases processing page;
- AUTHORIZING: processing purchases that require extra information to be approved;
- NOT_AUTHORIZED: purchases not authorized by their payment system;
- RECEIPT: approved purchases;
- ACCOUNT: user's account page.

Engineers of Peixe Urbano also pointed out the desired flow of conversion, i. e., how their pages was designed to lead users to purchase. According to them, all HOME, DEALS, FILTERED and SEARCH page types contains list of offers. Generally, navigation starts in HOME page, but depending on the origin of access, navigation might also starts in DEALS, FILTERED or SEARCH pages. If the access is organic, that is, if the user types the address of Peixe Urbano manually on her browser, the navigation often starts in HOME page. But if the user accesses Peixe Urbano through a search result using a search engine website, navigation might starts in a DEALS page, or in a FILTERED page, or even in a DEAL page. Nevertheless, engineers designed Peixe Urbano's navigation flow as the following sequence of steps:

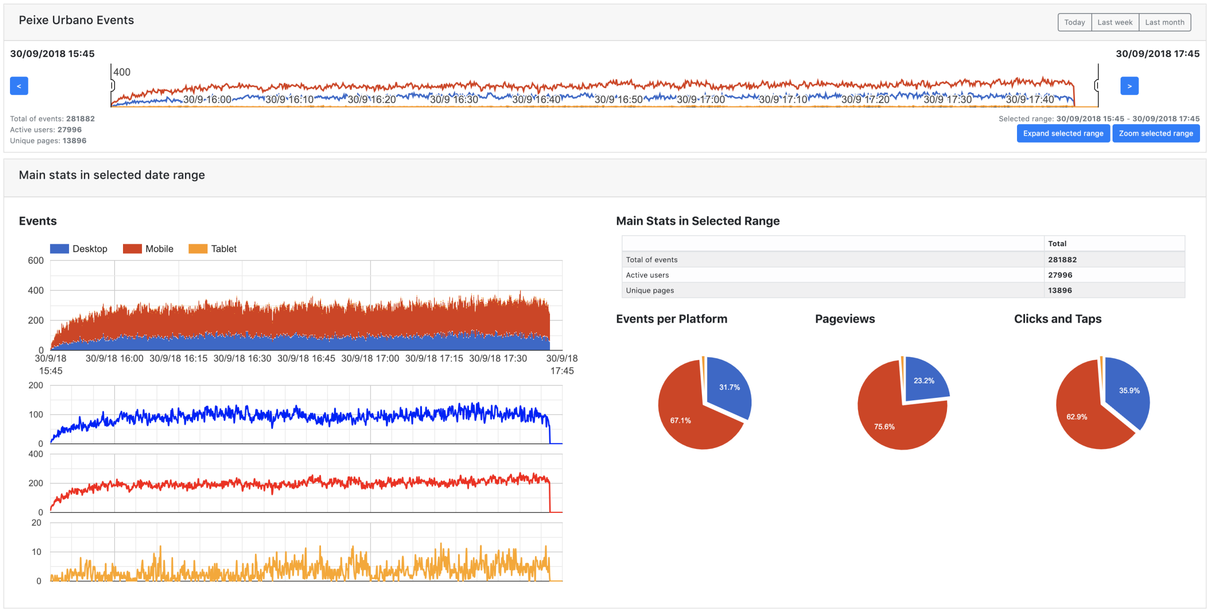
1. Access a listing of offers: HOME or DEALS pages;
2. Perform some filtering, searching for the best or desired offer: FILTERED or SEARCH pages;
3. Access the offer: DEAL page;
4. Add the offer to shopping cart: CHECKOUT page;
5. Submit the purchase: PROCESSING or RECEIPT pages.

These sequence of steps compose a well known term in e-commerce business: the conversion funnel. It is called funnel because the number of users accessing these pages decreases as users reach further steps. In other words, the number of users performing step 1 is larger than the number of users doing step 2, which is more accessed than pages in step 3, and so forth.

5.2.1 Page audience

During user's navigation in a specific website, there is always a page being accessed and any user interaction might imply an event being triggered on a web element. Footstep system is able to detect, in a specific period of time, pages and elements accessed by users. Footstep also can identify the page last seen in this period, by each user, what is called *active state* of user interaction. It is important to mention that *active state* might refer to an event occurred in the past. It depends on the selected period of analysis, but it represents the last accessed page in the selected period of time.

The Footstep visualization tool allows website administrators to choose the time range of analysis. The events are displayed in a time based chart, divided into three types of platforms: Desktop, Smartphone (Mobile) and Tablet. Figure 31 shows collected events during all the period of event collection and displays the share between events performed on Desktop, Mobile and Tablet, both page views and clicks (taps) are presented in pie charts. It is possible to notice that a great part of audience comes from mobile devices.


Peixe Urbano Events

30/09/2018 15:45
30/09/2018 17:45

Total of events: 281882
Active users: 27996
Unique pages: 13896

Selected range: 30/09/2018 15:45 - 30/09/2018 17:45

Main stats in selected date range

Main Stats in Selected Range

	Total
Total of events	281882
Active users	27996
Unique pages	13896

Figure 31 – Events from Peixe Urbano in September 30, 2018, from 3:45 PM to 5:45 PM BRT.

Taking into consideration the types of pages pointed by Peixe Urbano engineer team, Footstep detected the most accessed page types as shown Figure 32. Each box represents a specific page type, displays the number of unique visitors and the number of active users. The number written in blue represents the total of unique users who visited the page type in the period of analysis. The number written in green represent the active users in that page type, i. e., how much of these users are visiting that page type at the end of the period of analysis.

It is possible to notice that `HOME` page is the most accessed page type, with 5,271

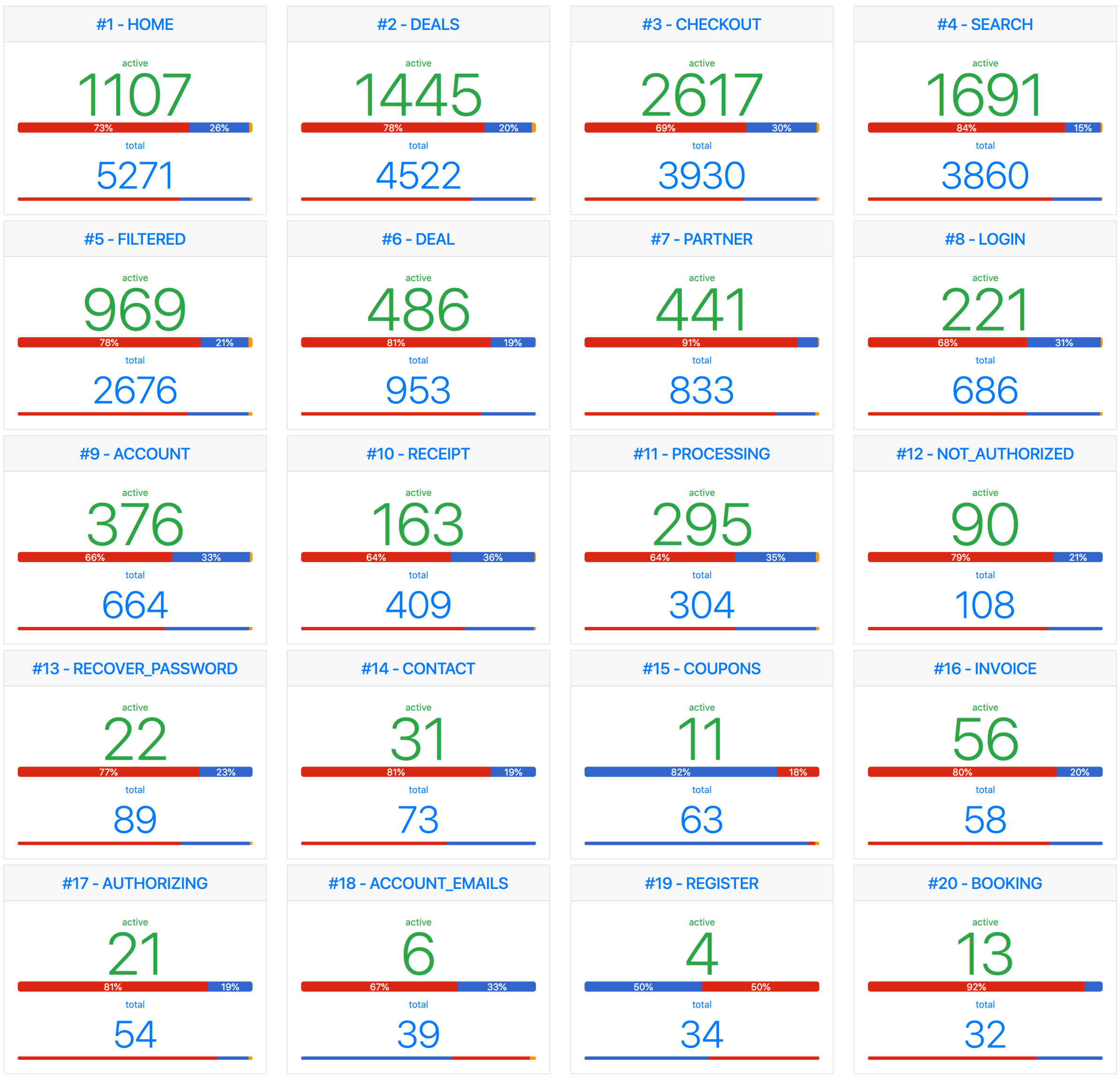

#1 - HOME
active 1107 73% 26% total 5271
#2 - DEALS
active 1445 78% 20% total 4522
#3 - CHECKOUT
active 2617 69% 30% total 3930
#4 - SEARCH
active 1691 84% 15% total 3860
#5 - FILTERED
active 969 78% 21% total 2676
#6 - DEAL
active 486 81% 19% total 953
#7 - PARTNER
active 441 91% total 833
#8 - LOGIN
active 221 68% 31% total 686
#9 - ACCOUNT
active 376 66% 33% total 664
#10 - RECEIPT
active 163 64% 36% total 409
#11 - PROCESSING
active 295 64% 35% total 304
#12 - NOT_AUTHORIZED
active 90 79% 21% total 108
#13 - RECOVER_PASSWORD
active 22 77% 23% total 89
#14 - CONTACT
active 31 81% 19% total 73
#15 - COUPONS
active 11 82% 18% total 63
#16 - INVOICE
active 56 80% 20% total 58
#17 - AUTHORIZING
active 21 81% 19% total 54
#18 - ACCOUNT_EMAILS
active 6 67% 33% total 39
#19 - REGISTER
active 4 50% 50% total 34
#20 - BOOKING
active 13 92% total 32

Figure 32 – Pages audience during event collection in Peixe Urbano.

unique users and 1,107 unique active users, although other page types have more active users, such as `CHECKOUT` (3,930 unique visitors and 2,617 active users), `SEARCH` (3,860 total visitors and 1,691 active visitors) and `DEALS` (4,522 total visitors and 1,445 active users).

All events hold when they were triggered, so it is possible to filter events in different ranges of date. For example, Figure 33 displays the last 15 minutes of event collection in Peixe Urbano, in September 30, 2018, from 5:30 PM to 5:45 PM BRT.

5.2.2 Element audience

The page type dashboard is a dashboard designed to extract information related to a specific page type, displaying, in a specific period, the most accessed elements divided by desktop, mobile and tablet platforms. For example, by accessing HOME page type

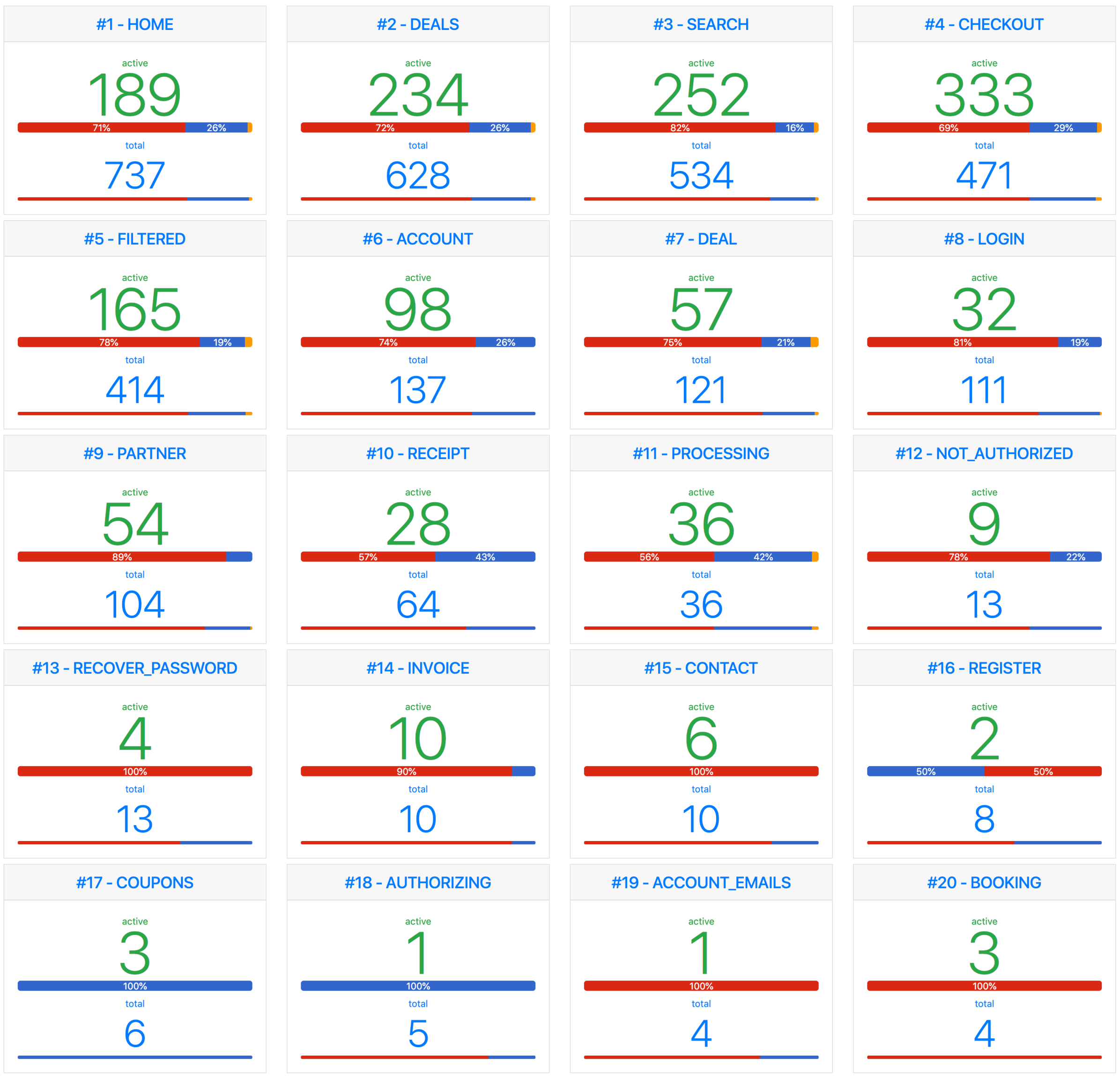
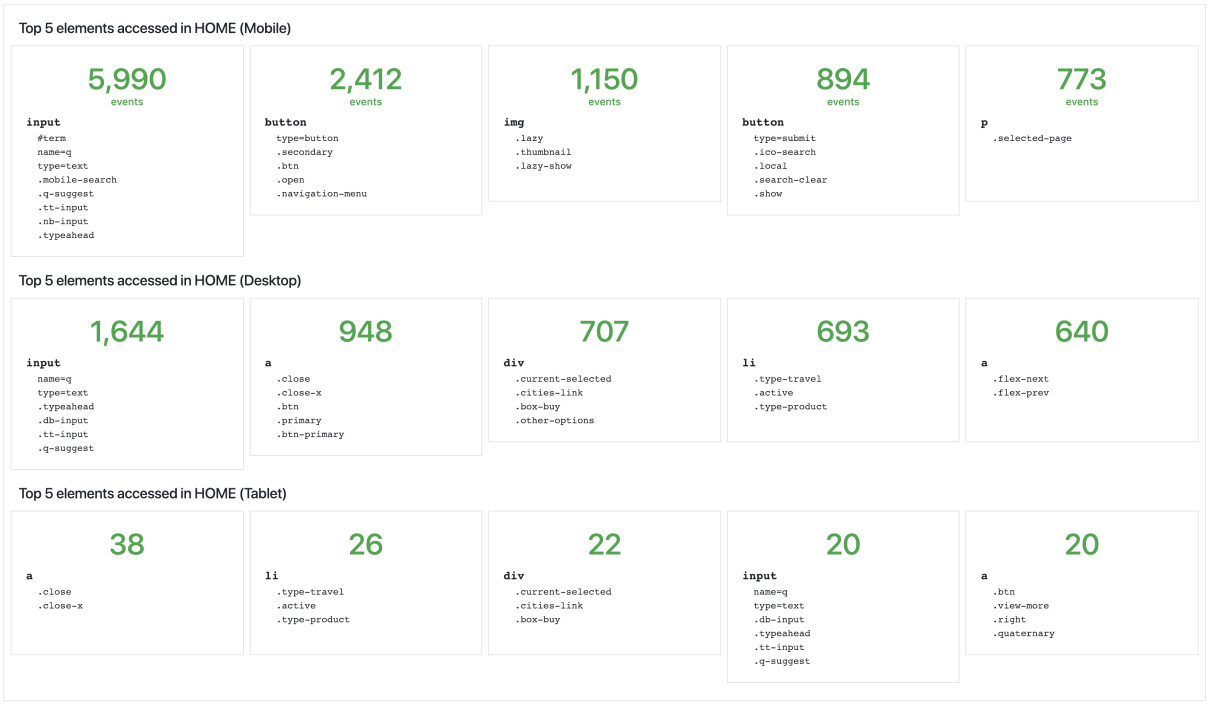


Figure 33 – Pages audience during the last 15 minutes of event collection in Peixe Urbano.

dashboard, for all period of event collection in Peixe Urbano, it is possible to extract the most accessed elements, as Figure 34 displays. Analyzing the results for the Mobile platform, it is possible to notice that the most accessed element is a tag `input` with id `term`, with 5,990 events triggered on this element in this period. This dashboard information provides an important insight about page's design undesired collateral effect by showing elements taking too much attention over other more important elements.

Figure 35 shows the 5 most accessed elements in DEAL page. This page is the product page and it is very important to e-commerce websites, because it can directly affect the company sales. Therefore, a wrong design decision related to this page may affect the user experience, compromising sales conversion. Therefore, website administrators need to track and monitor user interactions on this page in order to find issues that may cause losses in conversions.



Top 5 elements accessed in HOME (Mobile)

5,990 events
```
input
  #term
  name=q
  type=text
  .mobile-search
  .q-suggest
  .tt-input
  .nb-input
  .typeahead
```

2,412 events
```
button
  type=button
  .secondary
  .btn
  .open
  .navigation-menu
```

1,150 events
```
img
  .lazy
  .thumbnail
  .lazy-show
```

894 events
```
button
  type=submit
  .ico-search
  .local
  .search-clear
  .show
```

773 events
```
p
  .selected-page
```

Top 5 elements accessed in HOME (Desktop)

1,644
```
input
  name=q
  type=text
  .typeahead
  .db-input
  .tt-input
  .q-suggest
```

948
```
a
  .close
  .close-x
  .btn
  .primary
  .btn-primary
```

707
```
div
  .current-selected
  .cities-link
  .box-buy
  .other-options
```

693
```
li
  .type-travel
  .active
  .type-product
```

640
```
a
  .flex-next
  .flex-prev
```

Top 5 elements accessed in HOME (Tablet)

38
```
a
  .close
  .close-x
```

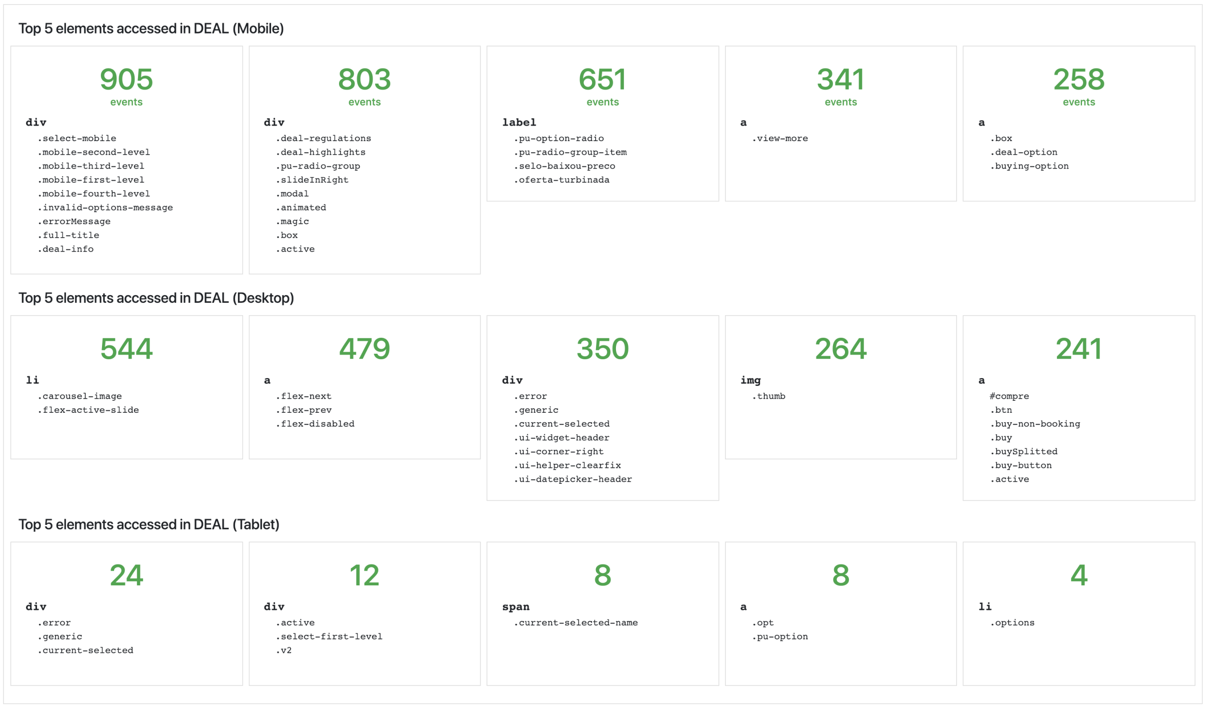
26
```
li
  .type-travel
  .active
  .type-product
```

22
```
div
  .current-selected
  .cities-link
  .box-buy
```

20
```
input
  name=q
  type=text
  .db-input
  .typeahead
  .tt-input
  .q-suggest
```

20
```
a
  .btn
  .view-more
  .right
  .quaternary
```

Figure 34 – Elements audience in page type HOME during event collection in Peixe Urbano.



Top 5 elements accessed in DEAL (Mobile)

905 events
```
div
  .select-mobile
  .mobile-second-level
  .mobile-third-level
  .mobile-first-level
  .mobile-fourth-level
  .invalid-options-message
  .errorMessage
  .full-title
  .deal-info
```

803 events
```
div
  .deal-regulations
  .deal-highlights
  .pu-radio-group
  .slideInRight
  .modal
  .animated
  .magic
  .box
  .active
```

651 events
```
label
  .pu-option-radio
  .pu-radio-group-item
  .selo-baixou-preco
  .oferta-turbinada
```

341 events
```
a
  .view-more
```

258 events
```
a
  .box
  .deal-option
  .buying-option
```

Top 5 elements accessed in DEAL (Desktop)

544
```
li
  .carousel-image
  .flex-active-slide
```

479
```
a
  .flex-next
  .flex-prev
  .flex-disabled
```

350
```
div
  .error
  .generic
  .current-selected
  .ui-widget-header
  .ui-corner-right
  .ui-helper-clearfix
  .ui-datepicker-header
```

264
```
img
  .thumb
```

241
```
a
  #compre
  .btn
  .buy-non-booking
  .buy
  .buySplitted
  .buy-button
  .active
```

Top 5 elements accessed in DEAL (Tablet)

24
```
div
  .error
  .generic
  .current-selected
```

12
```
div
  .active
  .select-first-level
  .v2
```

8
```
span
  .current-selected-name
```

8
```
a
  .opt
  .pu-option
```

4
```
li
  .options
```

Figure 35 – Elements audience in page type DEAL during event collection in Peixe Urbano.

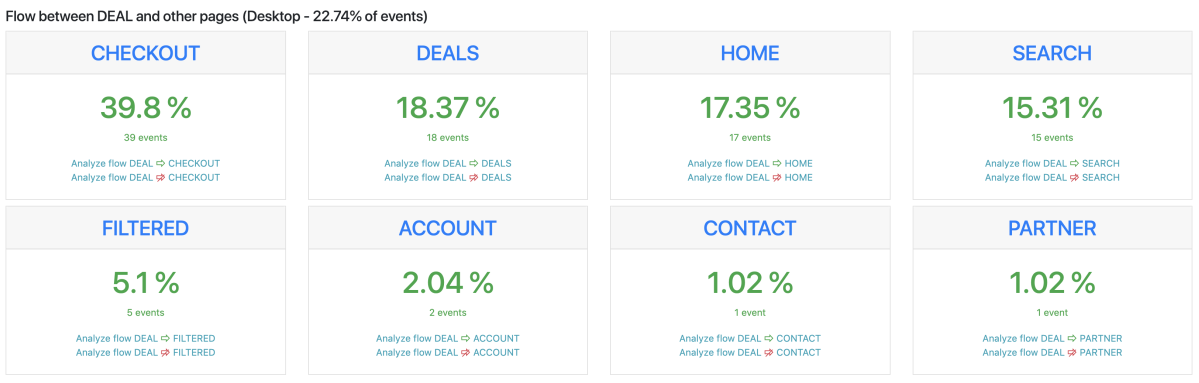
Another important point to notice is that a click event not always leads to another page. Only link elements lead to other pages, but Footstep JavaScript library capture clicks wherever they occur. In other words, if a user clicks on a text element, without a link, nothing happens, but a click event is triggered and sent to Footstep events queue.

So, several elements such as `div`, `span` or `img` may appear in elements dashboard as if they lead to other pages, but, in fact, what happened was that the user has clicked a non link event right before clicking on the browser's back button, which actually has leaded to another page.

5.2.3 Navigation flow

The page type dashboard also displays the navigation flow between two page types. The navigation flow is extracted from UsaGraph event nodes by taking two consecutive events belonging to same interaction and the page/element where these events were triggered. Based on a specific period, this dashboard displays how users navigate through pages, giving to website administrators trends and important insights about the design of pages. When designers and engineers define pages, they imagine a desired flow of navigation and want a way to validate and measure the actual navigation flow inside the website.

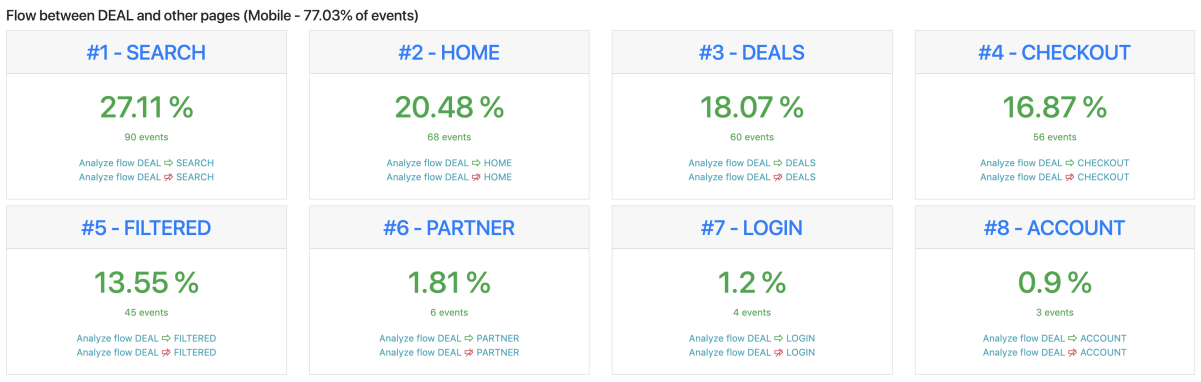
Figure 36 shows the flow between DEAL page type and other page types for the desktop platform. As engineers had foreseen, the most accessed page type after DEAL is CHECKOUT page type, representing 39.8% of the outgoing flow from DEAL. Then, DEALS page type is the second most accessed page type (18.37%) followed by HOME (17.35%), SEARCH (15.31%) and FILTERED (5.1%) page types.


Flow between DEAL and other pages (Desktop - 22.74% of events)

CHECKOUT	DEALS	HOME	SEARCH
39.8 %	18.37 %	17.35 %	15.31 %
39 events	18 events	17 events	15 events
Analyze flow DEAL ⇨ CHECKOUT	Analyze flow DEAL ⇨ DEALS	Analyze flow DEAL ⇨ HOME	Analyze flow DEAL ⇨ SEARCH
Analyze flow DEAL ⇏ CHECKOUT	Analyze flow DEAL ⇏ DEALS	Analyze flow DEAL ⇏ HOME	Analyze flow DEAL ⇏ SEARCH
FILTERED	**ACCOUNT**	**CONTACT**	**PARTNER**
5.1 %	2.04 %	1.02 %	1.02 %
5 events	2 events	1 event	1 event
Analyze flow DEAL ⇨ FILTERED	Analyze flow DEAL ⇨ ACCOUNT	Analyze flow DEAL ⇨ CONTACT	Analyze flow DEAL ⇨ PARTNER
Analyze flow DEAL ⇏ FILTERED	Analyze flow DEAL ⇏ ACCOUNT	Analyze flow DEAL ⇏ CONTACT	Analyze flow DEAL ⇏ PARTNER

Figure 36 – Flow between DEAL and other page types (Desktop) during event collection in Peixe Urbano.

When the same analysis is performed for the mobile platform, the result is quite different. As previously stated by engineers, the expected page type after DEAL IS CHECKOUT, however, as depicted in Figure 37, CHECKOUT page is only the fourth most accessed page from users that visited a DEAL page. This information is an alert to engineers showing that there is something wrong in the design of pages that is affecting the navigation flow, as 27.11% of users go from DEAL to SEARCH page, 20.48% from DEAL to HOME, 18.07% from DEAL to DEALS, and only 16.87% go from DEAL to CHECKOUT page type, which is an undesired and unexpected user behavior. This result is aggravated by the fact that mobile devices generates 77.03% of all tracked events from DEAL to another page.


Flow between DEAL and other pages (Mobile - 77.03% of events)

#1 - SEARCH
27.11 %
90 events
Analyze flow DEAL ⇨ SEARCH
Analyze flow DEAL ⇏ SEARCH

#2 - HOME
20.48 %
68 events
Analyze flow DEAL ⇨ HOME
Analyze flow DEAL ⇏ HOME

#3 - DEALS
18.07 %
60 events
Analyze flow DEAL ⇨ DEALS
Analyze flow DEAL ⇏ DEALS

#4 - CHECKOUT
16.87 %
56 events
Analyze flow DEAL ⇨ CHECKOUT
Analyze flow DEAL ⇏ CHECKOUT

#5 - FILTERED
13.55 %
45 events
Analyze flow DEAL ⇨ FILTERED
Analyze flow DEAL ⇏ FILTERED

#6 - PARTNER
1.81 %
6 events
Analyze flow DEAL ⇨ PARTNER
Analyze flow DEAL ⇏ PARTNER

#7 - LOGIN
1.2 %
4 events
Analyze flow DEAL ⇨ LOGIN
Analyze flow DEAL ⇏ LOGIN

#8 - ACCOUNT
0.9 %
3 events
Analyze flow DEAL ⇨ ACCOUNT
Analyze flow DEAL ⇏ ACCOUNT

Figure 37 – Flow between DEAL and other page types (Mobile) during event collection in Peixe Urbano.

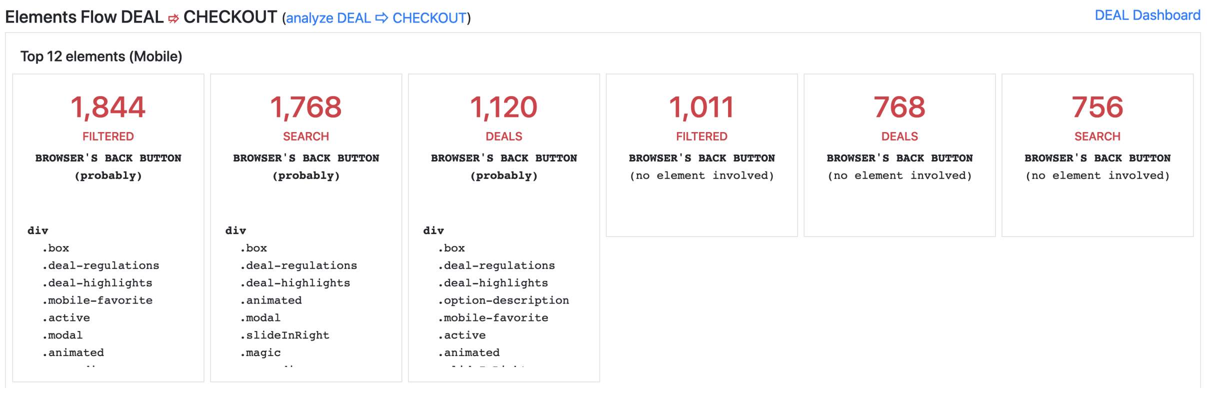
Another information that is rather important for engineers and website administrators is related to the elements involved in the navigation flow. A success navigation flow happens when an event occurring in a given page type leads to the a desired page type. For example, a flow between DEAL and CHECKOUT pages is considered a success flow. Thus, a failure flow occurs when an event leads the interaction far from the desired page type. For instance, when an event occurs in DEAL page and leads the user to other page than CHECKOUT page, which is the case verified in Figure 37, where SEARCH, HOME and DEALS page types receive more events from a DEAL page than CHECKOUT page.

Therefore, it is important to analyze a failure flow in order to identify anomalies in user interaction with the aim of putting users back in track of the desired flow. Figure 38 shows, for the mobile platform, the 12 most used way outs from the desired flow, which is going from DEAL to CHECKOUT page. Most users perform click events over `div` tag with classes `box, deal-regulations, deal-highlights` and end up in different pages than `CHECKOUT` page: 1,844 users has gone to `FILTERED` page, 1,786 users to `SEARCH` page and 1,120 users to `DEALS` page. The following way outs are by clicking in browser's back button: 1,011 users has returned to `FILTERED` page, 768 users to `DEALS` page and 756 users do `SEARCH` page.

5.2.4 Conversion points

Every commercial website has one or more specific goals and often administrators want to measure if users are behaving as expected. Therefore, it is important to provide tools to measure how users convert to specific pages and elements.

By exploiting UsaGraph capabilities, it is possible to measure how many users triggered events to reach specific nodes of the graph. According to the UsaGraph model, events triggered by the same user in the same interaction are translated to `Event` nodes



Elements Flow DEAL ⇨ CHECKOUT (analyze DEAL ⇨ CHECKOUT) DEAL Dashboard

Top 12 elements (Mobile)

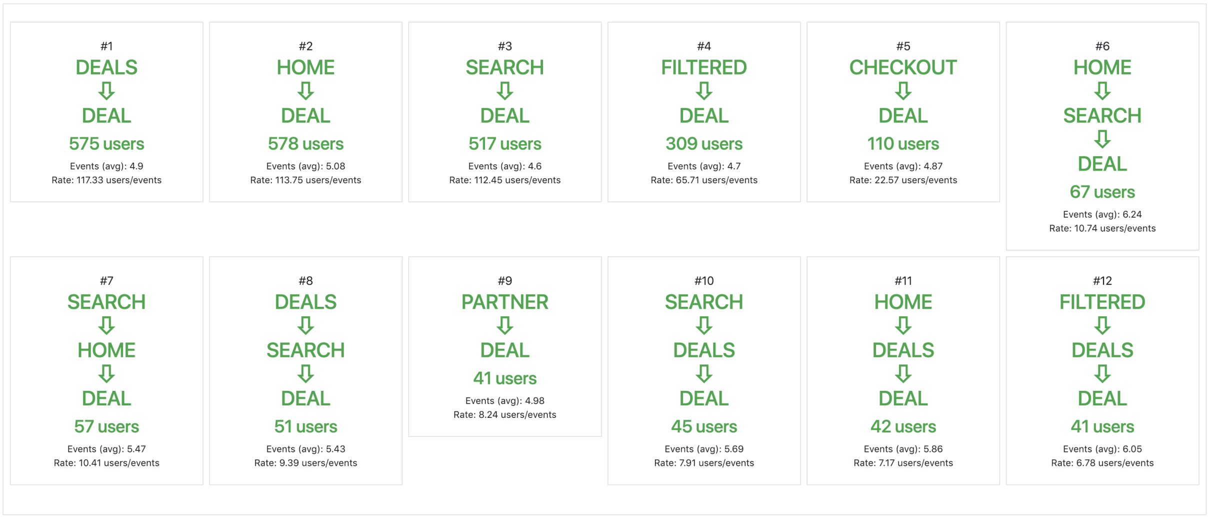
1,844	1,768	1,120	1,011	768	756
FILTERED	SEARCH	DEALS	FILTERED	DEALS	SEARCH
BROWSER'S BACK BUTTON (probably)	BROWSER'S BACK BUTTON (probably)	BROWSER'S BACK BUTTON (probably)	BROWSER'S BACK BUTTON (no element involved)	BROWSER'S BACK BUTTON (no element involved)	BROWSER'S BACK BUTTON (no element involved)
div .box .deal-regulations .deal-highlights .mobile-favorite .active .modal .animated	div .box .deal-regulations .deal-highlights .animated .modal .slideInRight .magic	div .box .deal-regulations .deal-highlights .option-description .mobile-favorite .active .animated			

Figure 38 – Failure flow between DEAL and CHECKOUT page type (Mobile) during event collection in Peixe Urbano.

chained and connected to each other by its relationship `NEXT`. Event nodes are also connected to `PageLabel` nodes and might be connected to `ElementLabel` nodes, depending on the type of event. By exploring this concept and using the collected events from Peixe Urbano, it is possible to measure, over a specific period of time, the number of users who interacted with specific pages and elements and how many events on average were spent to reach conversion points. This way, Footstep is able to calculate the conversion rate per page and per element.

Engineers are free to define conversion points. In the following example, the DEAL page type is defined as conversion point. Figure 39 shows users that interacted with DEAL page type and how they did it. Each box displays the navigation pattern used by users to reach a DEAL page, the number of users that performed this action, how many events in average were necessary until reaching the conversion point and the conversion rate. The first box shows 575 users, taking in average 4.9 events to navigate from DEALS page type to DEAL page type. When this navigation flow is compared to the second most used path, from HOME to DEAL page type, the conversion rate are very similar. But when the third and fourth navigation flows are compared, it is possible to notice that SEARCH performs much better than FILTERED page type. From SEARCH, 517 users took in average 4.6 events to visit a DEAL page, in contrast, from FILTERED page type, 309 users took 4.7 events to reach a DEAL page. Thus, the conversion rate of SEARCH is much better than the conversion rate of FILTERED.

Footstep also allows to take elements as conversion points. To illustrate this feature, lets consider the third most accessed element in DEAL page type. As illustrated in Figure 35, the element `label` with classes `.pu-option-radio`, `.pu-radio-group-item`, `.selo-baixou-preco` and `.oferta-turbinada` is the target. Figure 40 shows that the most used navigation flow to click on this label element is from SEARCH page type, where 141 users took in average 6.18 events to click on the conversion point element, resulting a rate of 22.83 users/event. The second most used navigation flow to convert to



#1
DEALS
⇩
DEAL
575 users
Events (avg): 4.9
Rate: 117.33 users/events

#2
HOME
⇩
DEAL
578 users
Events (avg): 5.08
Rate: 113.75 users/events

#3
SEARCH
⇩
DEAL
517 users
Events (avg): 4.6
Rate: 112.45 users/events

#4
FILTERED
⇩
DEAL
309 users
Events (avg): 4.7
Rate: 65.71 users/events

#5
CHECKOUT
⇩
DEAL
110 users
Events (avg): 4.87
Rate: 22.57 users/events

#6
HOME
⇩
SEARCH
⇩
DEAL
67 users
Events (avg): 6.24
Rate: 10.74 users/events

#7
SEARCH
⇩
HOME
⇩
DEAL
57 users
Events (avg): 5.47
Rate: 10.41 users/events

#8
DEALS
⇩
SEARCH
⇩
DEAL
51 users
Events (avg): 5.43
Rate: 9.39 users/events

#9
PARTNER
⇩
DEAL
41 users
Events (avg): 4.98
Rate: 8.24 users/events

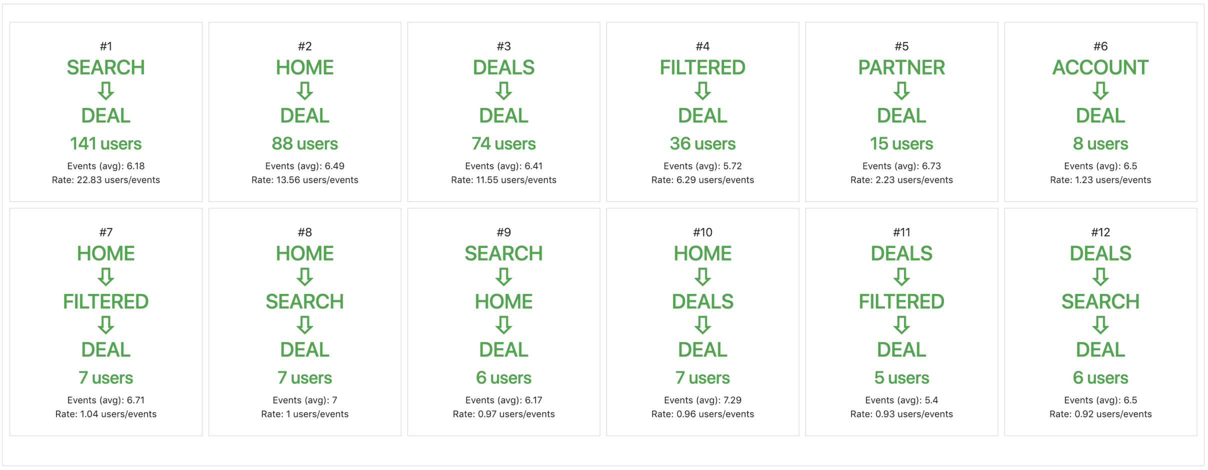
#10
SEARCH
⇩
DEALS
⇩
DEAL
45 users
Events (avg): 5.69
Rate: 7.91 users/events

#11
HOME
⇩
DEALS
⇩
DEAL
42 users
Events (avg): 5.86
Rate: 7.17 users/events

#12
FILTERED
⇩
DEALS
⇩
DEAL
41 users
Events (avg): 6.05
Rate: 6.78 users/events

Figure 39 – Users' navigation patterns to convert to DEAL page type.

this element is through HOME (88 users, 6.49 events) followed by DEALS (74 users, 6.41 events) and FILTERED (36 users, 5.72 events) page types.



#1
SEARCH
⇩
DEAL
141 users
Events (avg): 6.18
Rate: 22.83 users/events

#2
HOME
⇩
DEAL
88 users
Events (avg): 6.49
Rate: 13.56 users/events

#3
DEALS
⇩
DEAL
74 users
Events (avg): 6.41
Rate: 11.55 users/events

#4
FILTERED
⇩
DEAL
36 users
Events (avg): 5.72
Rate: 6.29 users/events

#5
PARTNER
⇩
DEAL
15 users
Events (avg): 6.73
Rate: 2.23 users/events

#6
ACCOUNT
⇩
DEAL
8 users
Events (avg): 6.5
Rate: 1.23 users/events

#7
HOME
⇩
FILTERED
⇩
DEAL
7 users
Events (avg): 6.71
Rate: 1.04 users/events

#8
HOME
⇩
SEARCH
⇩
DEAL
7 users
Events (avg): 7
Rate: 1 users/events

#9
SEARCH
⇩
HOME
⇩
DEAL
6 users
Events (avg): 6.17
Rate: 0.97 users/events

#10
HOME
⇩
DEALS
⇩
DEAL
7 users
Events (avg): 7.29
Rate: 0.96 users/events

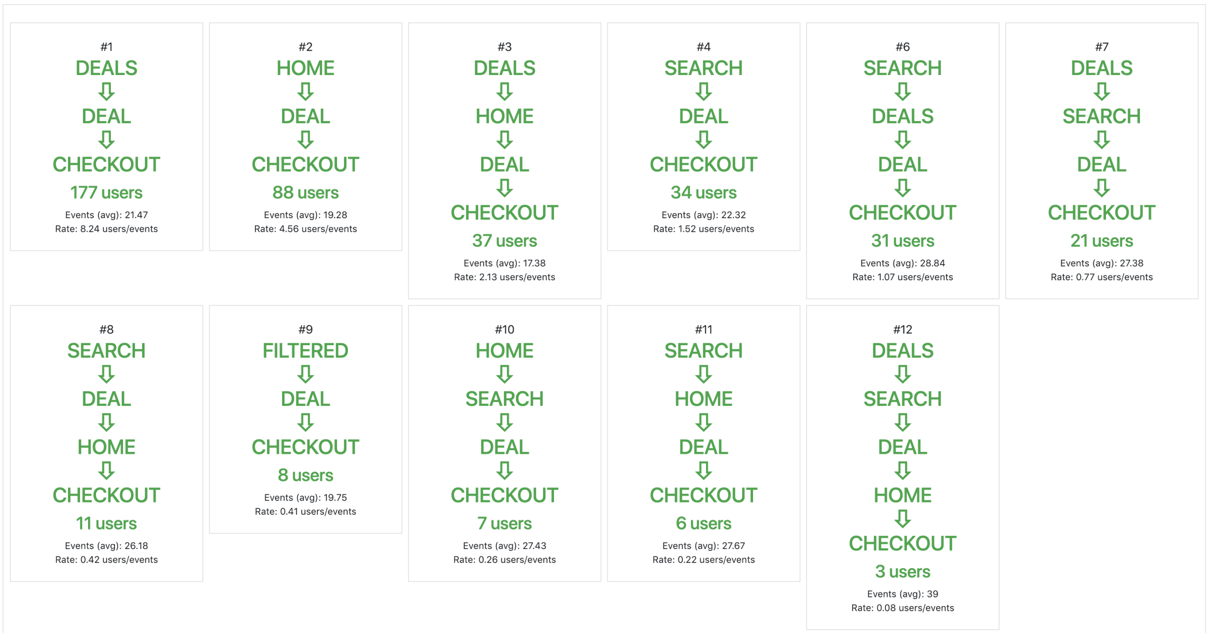
#11
DEALS
⇩
FILTERED
⇩
DEAL
5 users
Events (avg): 5.4
Rate: 0.93 users/events

#12
DEALS
⇩
SEARCH
⇩
DEAL
6 users
Events (avg): 6.5
Rate: 0.92 users/events

Figure 40 – Users' navigation patterns to convert to label element.

Footstep also allows setting more than one conversion point. For example, it is possible to extract how users accessed a page or clicked on a specific element and then accessed another page or clicked on another element. Using the same collected data of Peixe Urbano and setting two conversion points as a page view on DEAL page type and then a page view on CHECKOUT, it is possible to extract the most used navigation flows to reach these conversion points. Figure 41 shows that most part of users access a DEAL page and then a CHECKOUT page through DEALS page, with 177 users taking in average 21.47 events to reach these conversion points.

This conversion analysis shows users that reach CHECKOUT page passing by a DEAL page, but Footstep allows to set an element as conversion point. By setting the purchase submit button as final conversion point, it is possible to measure users' purchase



#1
DEALS ⇩ DEAL ⇩ CHECKOUT
177 users
Events (avg): 21.47
Rate: 8.24 users/events

#2
HOME ⇩ DEAL ⇩ CHECKOUT
88 users
Events (avg): 19.28
Rate: 4.56 users/events

#3
DEALS ⇩ HOME ⇩ DEAL ⇩ CHECKOUT
37 users
Events (avg): 17.38
Rate: 2.13 users/events

#4
SEARCH ⇩ DEAL ⇩ CHECKOUT
34 users
Events (avg): 22.32
Rate: 1.52 users/events

#6
SEARCH ⇩ DEALS ⇩ DEAL ⇩ CHECKOUT
31 users
Events (avg): 28.84
Rate: 1.07 users/events

#7
DEALS ⇩ SEARCH ⇩ DEAL ⇩ CHECKOUT
21 users
Events (avg): 27.38
Rate: 0.77 users/events

#8
SEARCH ⇩ DEAL ⇩ HOME ⇩ CHECKOUT
11 users
Events (avg): 26.18
Rate: 0.42 users/events

#9
FILTERED ⇩ DEAL ⇩ CHECKOUT
8 users
Events (avg): 19.75
Rate: 0.41 users/events

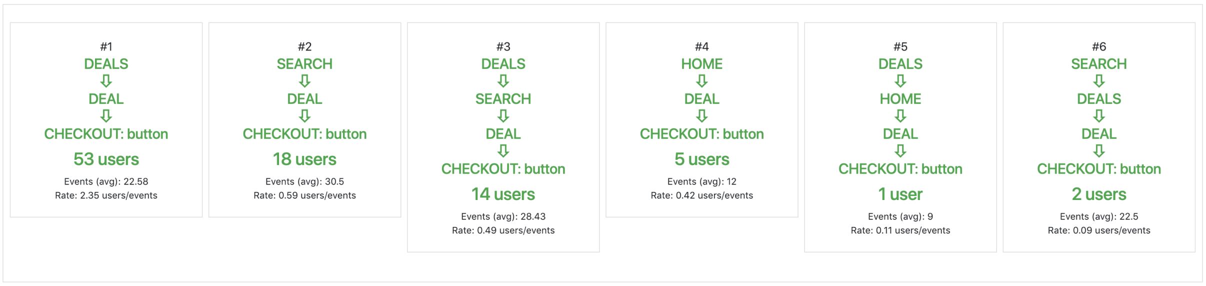
#10
HOME ⇩ SEARCH ⇩ DEAL ⇩ CHECKOUT
7 users
Events (avg): 27.43
Rate: 0.26 users/events

#11
SEARCH ⇩ HOME ⇩ DEAL ⇩ CHECKOUT
6 users
Events (avg): 27.67
Rate: 0.22 users/events

#12
DEALS ⇩ SEARCH ⇩ DEAL ⇩ HOME ⇩ CHECKOUT
3 users
Events (avg): 39
Rate: 0.08 users/events

Figure 41 – Users' navigation patterns to convert to DEAL and then CHECKOUT pages.

intention and extract navigation flows with better conversion. Figure 42 shows that 53 users navigate from DEALS, then accessed a DEAL page and finally hit purchase submit button in CHECKOUT page. They triggered in average 22.58 events to do this navigation flow, giving a conversion rate of 2.35 users/event. The second most accessed navigation flow includes again SEARCH page, with 18 users starting at SEARCH page, accessing a DEAL page and pressing the submit button, taking in average 30.5 events to perform this, a rate of 0.59 users/event. The third most accessed navigation flow also includes the SEARCH page, being accessed by 14 users with a rate of 0.49 users/event.



#1
DEALS ⇩ DEAL ⇩ CHECKOUT: button
53 users
Events (avg): 22.58
Rate: 2.35 users/events

#2
SEARCH ⇩ DEAL ⇩ CHECKOUT: button
18 users
Events (avg): 30.5
Rate: 0.59 users/events

#3
DEALS ⇩ SEARCH ⇩ DEAL ⇩ CHECKOUT: button
14 users
Events (avg): 28.43
Rate: 0.49 users/events

#4
HOME ⇩ DEAL ⇩ CHECKOUT: button
5 users
Events (avg): 12
Rate: 0.42 users/events

#5
DEALS ⇩ HOME ⇩ DEAL ⇩ CHECKOUT: button
1 user
Events (avg): 9
Rate: 0.11 users/events

#6
SEARCH ⇩ DEALS ⇩ DEAL ⇩ CHECKOUT: button
2 users
Events (avg): 22.5
Rate: 0.09 users/events

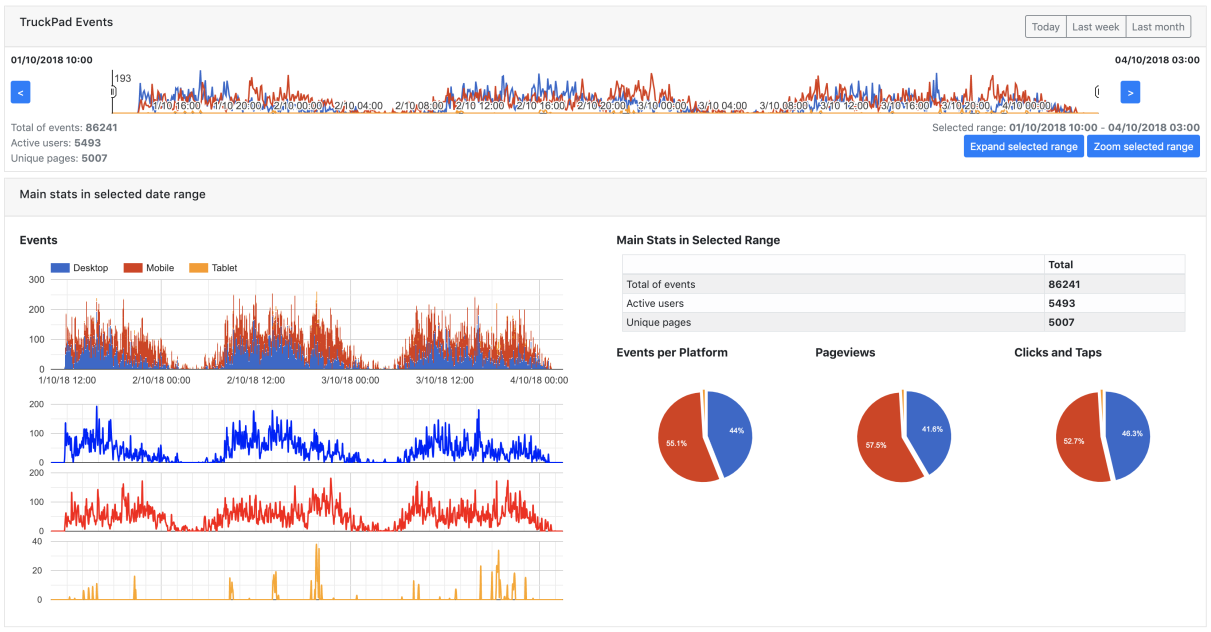
Figure 42 – Users' navigation patterns to convert to DEAL page and then click on submit purchase button (CHECKOUT page).

These results shows the importance of SEARCH page in terms of sales conversion, and give to engineers of Peixe Urbano a valuable feedback about the searching engine feature. Thus, according to these results, improving the search feature is likely to increase sales.

5.3 TruckPad

TruckPad is an online mobile marketplace where truckers can find truckloads using their mobile devices. It connects more than 10,000 shippers to carriers, helping to deliver more than 50,000 truckloads per month in Brazil. Because TruckPad is not a traditional e-commerce platform, it has been chosen to assess Footstep system in order to demonstrate the flexibility and versatility of our approach. Footstep system can be used in any website, because it deals with web pages and elements, therefore, the type of business is irrelevant.

As in Peixe Urbano, the installation of Footstep JavaScript library in TruckPad also did not required much effort by its engineer team, because TruckPad also uses Google Tag Manager. The collection of events on TruckPad occurred from October 1st to October 3rd, 2018. Figure 43 presents details of the collected events. A total of 86,241 events were collected, with 5,493 unique users, 5,007 unique pages and 85,709 elements.


TruckPad Events

01/10/2018 10:00 04/10/2018 03:00

Total of events: **86241**
Active users: **5493**
Unique pages: **5007**

Selected range: **01/10/2018 10:00 - 04/10/2018 03:00**

Main stats in selected date range

Main Stats in Selected Range

	Total
Total of events	**86241**
Active users	**5493**
Unique pages	**5007**

Figure 43 – Events from TruckPad from October 1 to October 3, 2018.

It was verified that TruckPad is divided in only three page types: home page, truckloads list and truckload page containing a green *Call Now* button, whose purpose is to display the phone number of the truckload shipper to the trucker. But truckloads list have several filters and sorting options, designed to truckers select the truckload type, the origin and the destiny of truckload, to sort the list by most recent or by weight or even by the declared value of the truckload. Every time a user applies a new filter, a new URL is accessed containing the old filters keywords added by the new filter keyword, each one as URL parts.

Thus, we configured 27 different types, most of them associated to a specific filter. This means that one page with more than one filter applied is connected to more than one page type. Section 3.5 mentioned that `PageLabel` nodes were designed to cluster different pages by matching URL patterns and there is no restriction to the number of clusters. Some of the registered page types are:

- HOME: `www.truckpad.com.br`;
- TRUCKLOADS: All URL's starting with `www.truckpad.com.br/fretes`, except those ending with truckload ID;
- TRUCKLOAD: URL ending with truckload ID (a sequence of 22 or 25 alphanumeric characters);
- TRUCKLOADS_RECENT: URL ending with the sort path param `por-ordem-de-data-postagem`;
- FROM_SP: URL containing `de-sp` or `carga-de-sp` in any position, listing truckloads from the state of São Paulo, Brazil.

In this way, a page with URL `www.truckpad.com.br/fretes/carga-de-sp /por-ordem-de-data-postagem` will be connected to page types FROM_SP and TRUCKLOADS_R Listing 5.1 shows more examples extracted from TruckPad and how each URL is classified by page types.

Listing 5.1 – TruckPad URL's and truckloads filters

```
URL: www.truckpad.com.br/fretes/carga-por-ordem-de-data-postagem
Page types: TRUCKLOADS, TRUCKLOADS_RECENT

URL: www.truckpad.com.br/fretes/carga-de-sp/por-ordem-de-data-postagem
Page types: TRUCKLOADS, FROM_SP, TRUCKLOADS_RECENT

URL: www.truckpad.com.br/fretes/de-mg/para-sp/por-ordem-de-data-postagem
Page types: TRUCKLOADS, FROM_MG, TO_SP, TRUCKLOADS_RECENT
```

5.3.1 Page audience

By the nature of TruckPad's URLs and the page types that we had defined, the TruckPad page audience analysis can confuse their administrators. Figure 44 shows the most accessed page types for TruckPad and it is possible to notice that TRUCKLOADS is the most accessed page. During the TruckPad event collection, 3,959 users accessed TRUCKLOADS page, being 2,833 active users, that is, users that is actually accessing this page at the analysis time. However, it is important to notice that the same page URL may be classified as more than one page type. This means that the same user can appear in more than one page type. So, the page audience, in this specific case, is also a TruckPad filter audience. In other words, it is possible to know how many users are filtering truckloads for a specific vehicle type, from one specific state of Brazil.

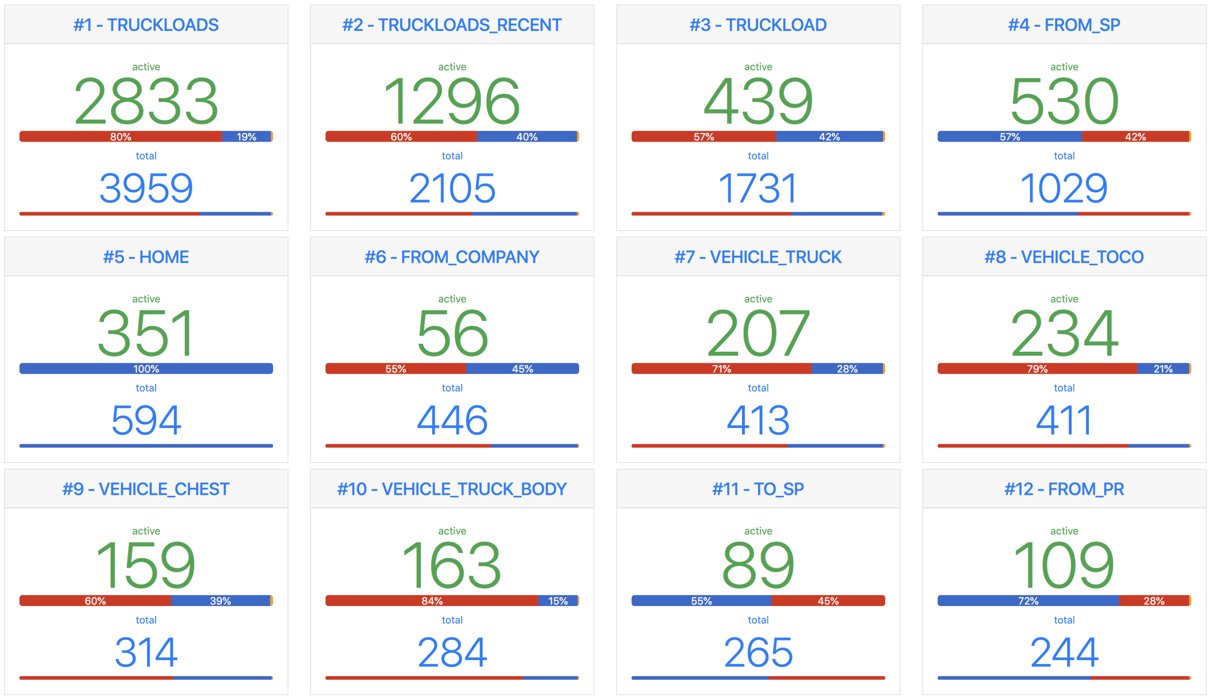


Figure 44 – Pages audience during event collection in TruckPad.

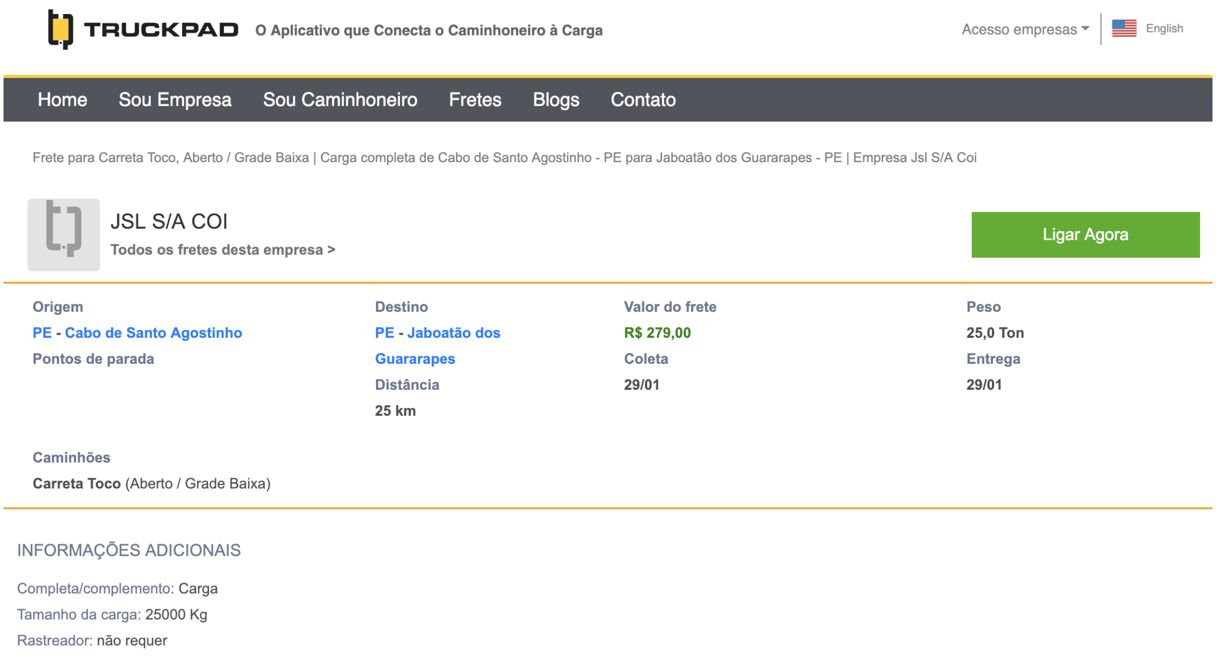
5.3.2 Element audience and conversion points

TruckPad's final purpose is quite simple: connect truckers to truckloads. In terms of Footstep navigation flow, the objective is lead users from TRUCKLOADS pages through filters to TRUCKLOAD page and displays the shipper phone number, allowing the user to interact by clicking on the green *Call Now* button, or in Portuguese *Ligar Agora*. In fact, there are two *Call Now* buttons in the same page for desktop/tablet platforms, as Figures 45 and 46 shows, and only one for the mobile platform, as Figure 47 displays.

Having two identical buttons for the same purpose in the same page is not necessarily a design's mistake, but it might affect usability by confusing the user.

Ideally, for a specific page, the audience of elements present in all platforms should follow their page audience among platforms. For example, according to the TRUCKLOAD page audience, as shown in Figure 44, 57% of events are performed using smartphones (mobile), 42% are performed using the desktop and 4% are performed using tablets. Thus, the percentage share of click events on *Call Now* button among platforms should match the percentage of page access. In other words, the conversion should be the same, no matter the platform.

By analyzing elements audience from TRUCKLOAD page, it is possible to discover if *Call Now* buttons receive the expected attention and are the most accessed element. Figure 48 shows element's audience using the mobile platform. This dashboard confirms that *Call Now* button is the most accessed element in TRUCKLOAD mobile page, with



TRUCKPAD O Aplicativo que Conecta o Caminhoneiro à Carga

Acesso empresas English

Home Sou Empresa Sou Caminhoneiro Fretes Blogs Contato

Frete para Carreta Toco, Aberto / Grade Baixa | Carga completa de Cabo de Santo Agostinho - PE para Jaboatão dos Guararapes - PE | Empresa Jsl S/A Coi

JSL S/A COI

Todos os fretes desta empresa >

Ligar Agora

Origem	Destino	Valor do frete	Peso
PE - Cabo de Santo Agostinho	PE - Jaboatão dos Guararapes	R$ 279,00	25,0 Ton
Pontos de parada	Distância	Coleta	Entrega
	25 km	29/01	29/01

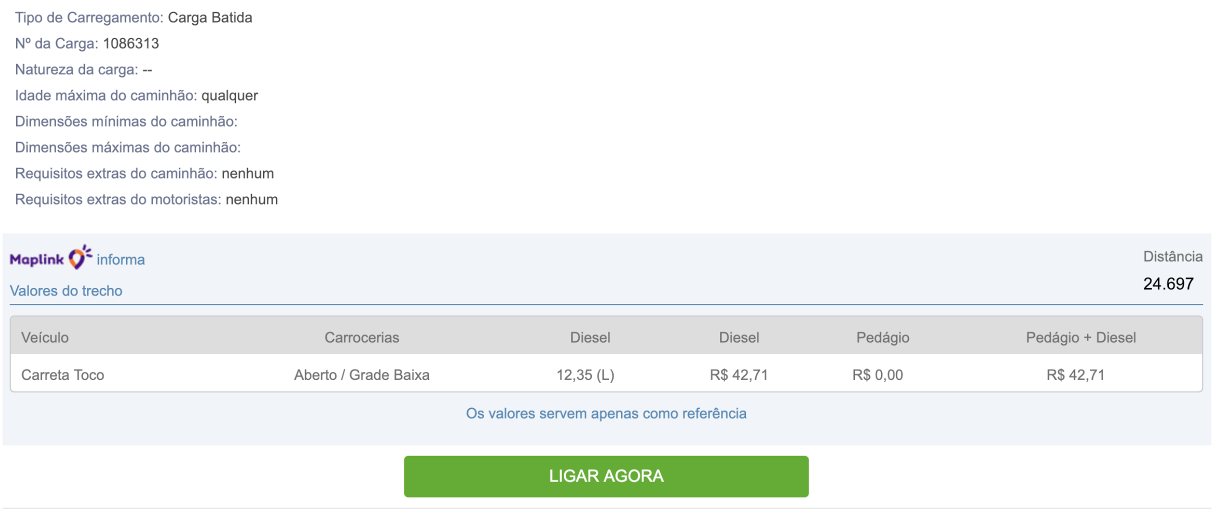
Caminhões

Carreta Toco (Aberto / Grade Baixa)

INFORMAÇÕES ADICIONAIS

Completa/complemento: Carga
Tamanho da carga: 25000 Kg
Rastreador: não requer

Figure 45 – First *Call Now* button in Truckload page from for desktop platform.



Tipo de Carregamento: Carga Batida
Nº da Carga: 1086313
Natureza da carga: --
Idade máxima do caminhão: qualquer
Dimensões mínimas do caminhão:
Dimensões máximas do caminhão:
Requisitos extras do caminhão: nenhum
Requisitos extras do motoristas: nenhum

Maplink informa

Valores do trecho

Distância
24.697

Veículo	Carrocerias	Diesel	Diesel	Pedágio	Pedágio + Diesel
Carreta Toco	Aberto / Grade Baixa	12,35 (L)	R$ 42,71	R$ 0,00	R$ 42,71

Os valores servem apenas como referência

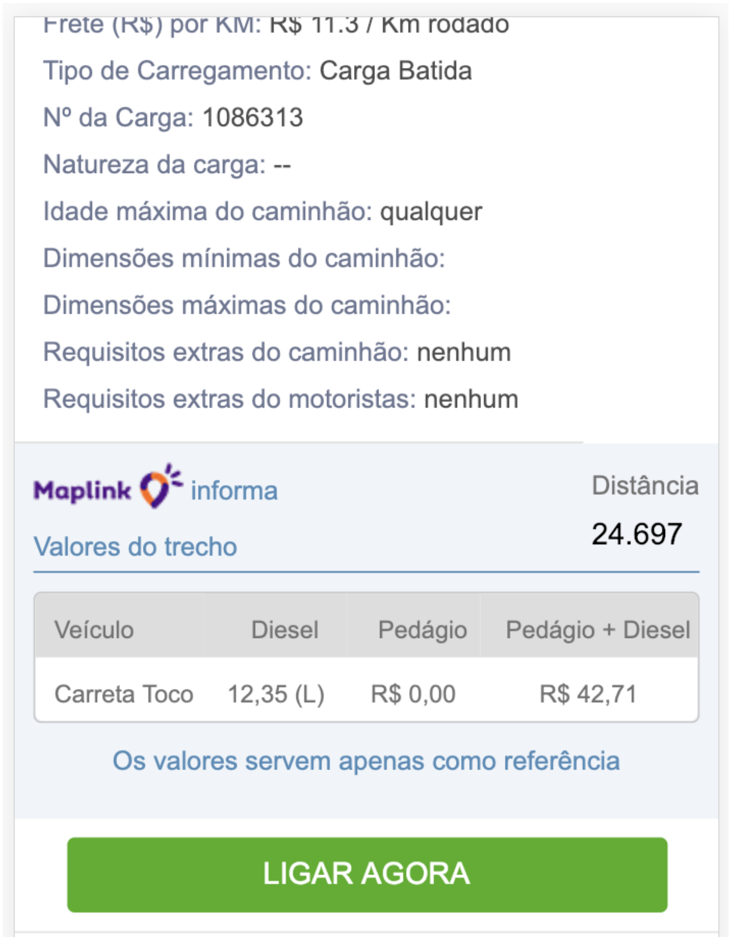
LIGAR AGORA

Figure 46 – Second *Call Now* button in Truckload page from for desktop platform.

1,240 events registered during the period of event collection.

Figure 49 displays the elements audience dashboard for the desktop platform, which presents two *Call Now* buttons. The data presented in this dashboard shows that the first *Call Now* button, at the top of the TRUCKLOAD page, is the most accessed element, registering 822 events triggered by users, against 380 events triggered on the second button, at the bottom of the page. As expected, the tablet platform presented a very low number of events: 18 events triggered on the first button and 8 events triggered on the second, as depicted in Figure 50.

The total number of events triggered on *Call Now* buttons, considering all three


Frete (R$) por KM: R$ 11.3 / Km rodado

Tipo de Carregamento: Carga Batida

Nº da Carga: 1086313

Natureza da carga: --

Idade máxima do caminhão: qualquer

Dimensões mínimas do caminhão:

Dimensões máximas do caminhão:

Requisitos extras do caminhão: nenhum

Requisitos extras do motoristas: nenhum

Maplink informa

Valores do trecho

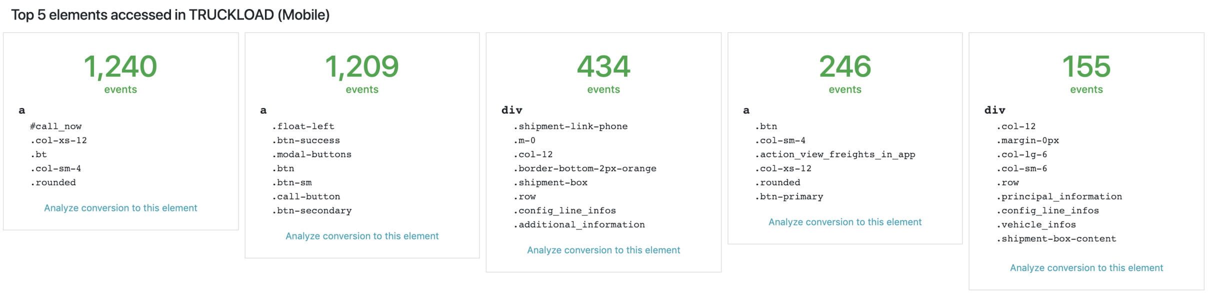
Distância

24.697

Veículo	Diesel	Pedágio	Pedágio + Diesel
Carreta Toco	12,35 (L)	R$ 0,00	R$ 42,71

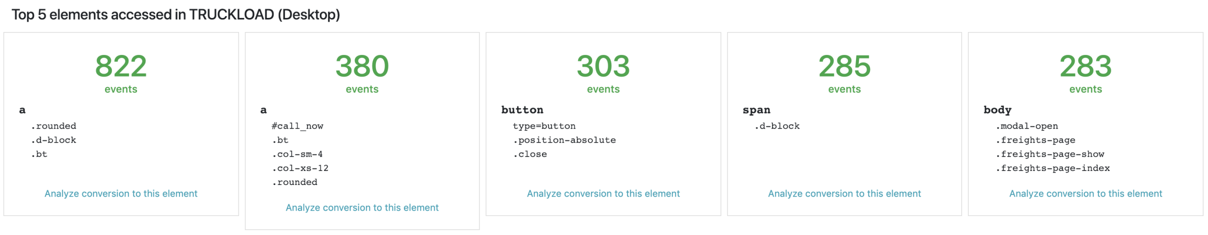
Os valores servem apenas como referência

Figure 47 – *Call Now* button in Truckload page from for mobile platform.


Top 5 elements accessed in TRUCKLOAD (Mobile)

```
1,240
events
a
  #call_now
  .col-xs-12
  .bt
  .col-sm-4
  .rounded
Analyze conversion to this element

1,209
events
a
  .float-left
  .btn-success
  .modal-buttons
  .btn
  .btn-sm
  .call-button
  .btn-secondary
Analyze conversion to this element

434
events
div
  .shipment-link-phone
  .m-0
  .col-12
  .border-bottom-2px-orange
  .shipment-box
  .row
  .config_line_infos
  .additional_information
Analyze conversion to this element

246
events
a
  .btn
  .col-sm-4
  .action_view_freights_in_app
  .col-xs-12
  .rounded
  .btn-primary
Analyze conversion to this element

155
events
div
  .col-12
  .margin-0px
  .col-lg-6
  .col-sm-6
  .row
  .principal_information
  .config_line_infos
  .vehicle_infos
  .shipment-box-content
Analyze conversion to this element
```

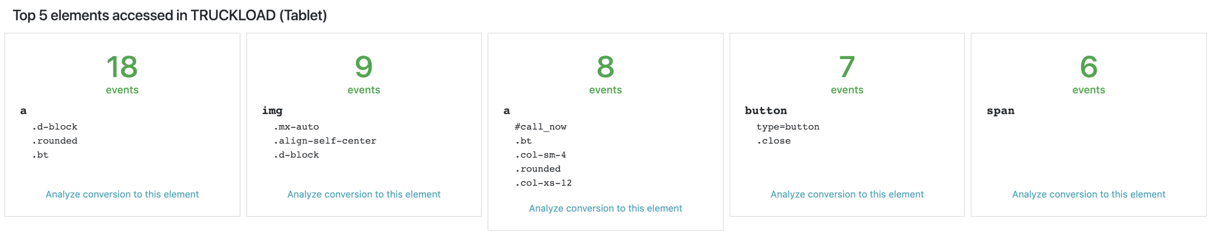
Figure 48 – Elements audience in TRUCKLOAD page, mobile platform.


Top 5 elements accessed in TRUCKLOAD (Desktop)

```
822
events
a
  .rounded
  .d-block
  .bt
Analyze conversion to this element

380
events
a
  #call_now
  .bt
  .col-sm-4
  .col-xs-12
  .rounded
Analyze conversion to this element

303
events
button
  type=button
  .position-absolute
  .close
Analyze conversion to this element

285
events
span
  .d-block
Analyze conversion to this element

283
events
body
  .modal-open
  .freights-page
  .freights-page-show
  .freights-page-index
Analyze conversion to this element
```

Figure 49 – Elements audience in TRUCKLOAD page, desktop platform.

platforms, was 2,468 – 50% of this amount correspond to events performed on smartphones, 48.7% correspond to events performed on the desktop, and 1.3% correspond to events


Top 5 elements accessed in TRUCKLOAD (Tablet)

18 events
```
a
  .d-block
  .rounded
  .bt
```
Analyze conversion to this element

9 events
```
img
  .mx-auto
  .align-self-center
  .d-block
```
Analyze conversion to this element

8 events
```
a
  #call_now
  .bt
  .col-sm-4
  .rounded
  .col-xs-12
```
Analyze conversion to this element

7 events
```
button
  type=button
  .close
```
Analyze conversion to this element

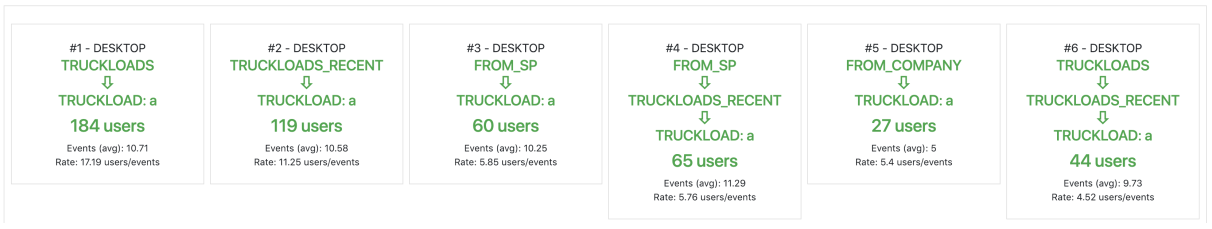
6 events
```
span
```
Analyze conversion to this element

Figure 50 – Elements audience in TRUCKLOAD page, tablet platform.

performed on tablets.

By cross-referencing this result with the TRUCKLOAD page audience, it is possible to notice that mobile and tablet element audience lag behind page audience, and desktop presents a better element audience. Despite mobile access represents 57% of total access, desktop *Call Now* buttons have received more clicks than mobile *Call Now* button.

In order to leverage the analysis of *Call Now* buttons audience, the conversion point dashboard can show how they are being accessed. Analyzing the first desktop *Call Now* button as conversion point, 184 users took in average 10.71 steps to go from TRUCKLOADS page to TRUCKLOAD page and hit the *Call Now* button at the top of page, presenting a rate of 17.19 users/events, as Figure 51 shows.


#1 - DESKTOP
TRUCKLOADS
⇩
TRUCKLOAD: a
184 users
Events (avg): 10.71
Rate: 17.19 users/events

#2 - DESKTOP
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a
119 users
Events (avg): 10.58
Rate: 11.25 users/events

#3 - DESKTOP
FROM_SP
⇩
TRUCKLOAD: a
60 users
Events (avg): 10.25
Rate: 5.85 users/events

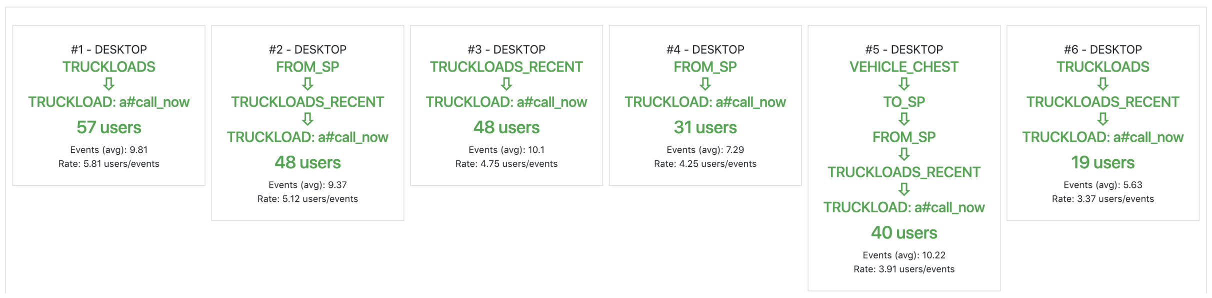
#4 - DESKTOP
FROM_SP
⇩
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a
65 users
Events (avg): 11.29
Rate: 5.76 users/events

#5 - DESKTOP
FROM_COMPANY
⇩
TRUCKLOAD: a
27 users
Events (avg): 5
Rate: 5.4 users/events

#6 - DESKTOP
TRUCKLOADS
⇩
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a
44 users
Events (avg): 9.73
Rate: 4.52 users/events

Figure 51 – First *Call Now* button conversion in desktop platform.

In the second desktop *Call Now* button analysis as conversion point, 57 users took in average 9.81 steps to go from TRUCKLOADS page to TRUCKLOAD page and hit the *Call Now* button at the bottom of page, presenting a rate of 5.81 users/events, as Figure 52 shows.


#1 - DESKTOP
TRUCKLOADS
⇩
TRUCKLOAD: a#call_now
57 users
Events (avg): 9.81
Rate: 5.81 users/events

#2 - DESKTOP
FROM_SP
⇩
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a#call_now
48 users
Events (avg): 9.37
Rate: 5.12 users/events

#3 - DESKTOP
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a#call_now
48 users
Events (avg): 10.1
Rate: 4.75 users/events

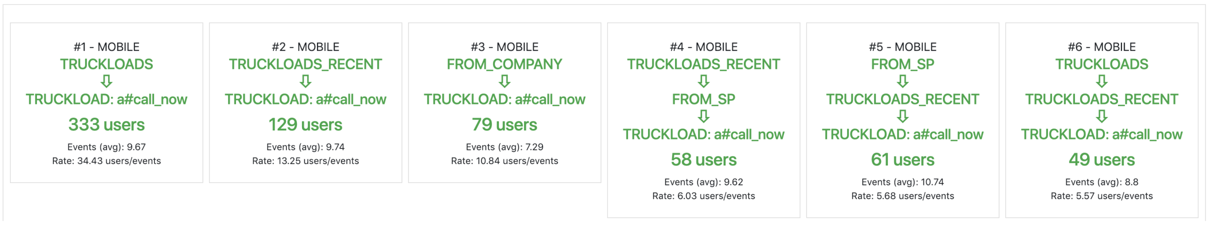
#4 - DESKTOP
FROM_SP
⇩
TRUCKLOAD: a#call_now
31 users
Events (avg): 7.29
Rate: 4.25 users/events

#5 - DESKTOP
VEHICLE_CHEST
⇩
TO_SP
⇩
FROM_SP
⇩
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a#call_now
40 users
Events (avg): 10.22
Rate: 3.91 users/events

#6 - DESKTOP
TRUCKLOADS
⇩
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a#call_now
19 users
Events (avg): 5.63
Rate: 3.37 users/events

Figure 52 – Second *Call Now* button conversion in desktop platform.

In the mobile platform, the conversion of the *Call Now* button shows 333 users coming from TRUCKLOADS page, taking in average 9.67 events to trigger a click (tap) on this button, a rate of 34.43 users/events. In terms of the number of users converting to *Call Now* button, mobile represents 58% and desktop corresponds to 42%, which is very approximate from pages audience share between the mobile and the desktop platforms. Also, smartphone users spent less events to convert than desktop users.



#1 - MOBILE
TRUCKLOADS
⇩
TRUCKLOAD: a#call_now
333 users
Events (avg): 9.67
Rate: 34.43 users/events

#2 - MOBILE
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a#call_now
129 users
Events (avg): 9.74
Rate: 13.25 users/events

#3 - MOBILE
FROM_COMPANY
⇩
TRUCKLOAD: a#call_now
79 users
Events (avg): 7.29
Rate: 10.84 users/events

#4 - MOBILE
TRUCKLOADS_RECENT
⇩
FROM_SP
⇩
TRUCKLOAD: a#call_now
58 users
Events (avg): 9.62
Rate: 6.03 users/events

#5 - MOBILE
FROM_SP
⇩
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a#call_now
61 users
Events (avg): 10.74
Rate: 5.68 users/events

#6 - MOBILE
TRUCKLOADS
⇩
TRUCKLOADS_RECENT
⇩
TRUCKLOAD: a#call_now
49 users
Events (avg): 8.8
Rate: 5.57 users/events

Figure 53 – *Call Now* button conversion in mobile platform.

Comparing and analyzing the results between mobile and desktop platforms, we can extract some information and insights:

- *Call Now* button at the top of page presents a better audience/conversion;
- Having two *Call Now* buttons in the same page may improve the conversion, but also might the usability of the page;
- Although element audience on the mobile platform presented less events than expected, the conversion is better than desktop buttons conversion;
- Placing *Call Now* button fixed on top and always visible on the screen might improve its audience and conversion.

5.4 Groupon Brazil

Groupon is the first daily deals e-commerce company in the world, founded in 2008 in United States of America and further widespread abroad, including Brazil and some countries from South America, Europe and Asia. In 2017, Groupon Brazil and Peixe Urbano had announced a fusion in operation and, since then, both websites share the same infrastructure and are maintained and developed by the same engineer team. However, Groupon Brazil and Peixe Urbano serve different offers, have different users and their page design have some differences.

The event collection in Groupon Brazil occurred at the same time of event collection in Peixe Urbano in order to assess the Footstep capability to collect and process several events from highly accessed and different websites. The pages instrumentation with Footstep JavaScript was also easy due to the fact that Groupon also uses Google Tag Manager.

In September 30, from 3:45 PM to 5:45 PM, BRT, a total of 92,701 events were collected, with 5,942 unique users, 6,084 unique pages and 164,034 elements. The page types of Groupon Brazil are very similar to the page types defined for Peixe Urbano.

5.4.1 Page audience

Section 5.2.3 identified a probable failure flow between DEAL and CHECKOUT page in mobile platform, presenting only 16.87% of total flow. This is considered as failure flow because is expected that users access a DEAL page and then access CHECKOUT page. For desktop platform, the result was 39.8% of flow going from DEAL to CHECKOUT page, being verified as best flow between DEAL and CHECKOUT pages.

The same analysis of flow between DEAL and CHECKOUT pages for Groupon Brazil presented a worst result for both platforms. Mobile platform presented a flow of 6.49% of users from DEAL to CHECKOUT page behind from FILTERED (30.8%), SEARCH (28.59%), DEALS (19.75%) and HOME (8.84%) page flows. Desktop platform presented a flow of 23.36% of users from DEAL to CHECKOUT page, behind from FILTERED page with 24.06% of flow. These results reinforce the alert to engineers about the design and user experience for DEAL page type in mobile platform, whose platform audience is 71.93% of users against 26.09% of desktop audience.

5.5 Final Remarks

This chapter explored three websites using the Footstep System to analyze their user behavior between pages and elements. The performance of event collection and processing was good by the fact of two websites (Peixe Urbano and Groupon Brazil) with a great audience being tracked at the same time, it was not detected any saturation between NiFi processors.

Great insights were extracted from Footstep results for e-commerce websites, such as a possible lack of user experience in mobile platform design of offers pages, presenting a sub optimal flow to checkout page. Groupon Brazil also presented this sub optimal flow between offer and checkout page for desktop platform.

Another valuable insight extracted from conversion point dashboard for e-commerce websites is the fact of most of users placing orders, in other words, hitting submit purchase button, come from textual search results page, showing the importance of this page type for business.

This chapter also presented the versatility of Footstep System by the analysis of other kind of business website: TruckPad, a transportation marketplace that connects truckers to truckloads. It was possible to measure the element audience of button that shows the phone number of shipper to truckers. The results showed that mobile platform maybe is affected by the place of *Call Now* button, by the difference of the platform audience and element audience. Mobile platform audience is higher than desktop platform audience, but desktop truckload contains two *Call Now* buttons, presenting the same element audience than mobile platform.

6 Conclusion

Understanding the user behavior in web applications is paramount to website owners and administrators. Existing solutions fail to provide detailed and useful information regarding the interaction of users with web page elements. The Footstep system was developed to tackle this problem, providing an all-in-one approach designed to collect events along with page elements in client side and processes these events in a fast and reliable way. Moreover, thanks to its graph-based data model, Footstep is able to provide relevant information about the behavior of users as they browse web applications.

In order to achieve the objectives set out on Section 1.2, Footstep was build upon a modular architecture. The separation into modules provides asynchronous processing of events, separating concerns and responsibilities. The Gathering Events Module is responsible for collecting events efficiently and transparently to the end user. The Consumer & Parser Module is responsible for processing and parsing events to the UsaGraph model. The Generalization Module is responsible for identifying similar pages and elements, providing flexibility to the analysis, allowing website administrators to label similar pages in order to cluster different and dynamic URLs into the same page type. Finally, the Analytics Module offers several dashboards that allows the visualization of the audience on pages and elements, of navigation flows and of conversion rates.

Technologically speaking, important choices were made to implement the Footstep System. RabbitMQ, an open source message broker, was used in conjunction with Apache NiFi, a tool specialized in data transformation and interchange, to provide an efficient solution for processing large volumes of events. Parsed events are finally transformed into nodes and edges according to the UsaGraph model and stored in Neo4J, a native graph-based database.

In order to evaluate the proposed system, experiments with different web applications were conducted. The results show that the proposed system present versatility to deal with different types of e-business applications, supports the collection of events in high traffic websites containing thousands of different pages and offers both near real-time and historical analysis.

The development of the Footstep System produced relevant contributions to user behavior analysis, being distinguished among them:

- Automatic event collection with near zero code injection;
- Automatic page elements evaluation;

- Fast and asynchronous events processing, decoupling event collection from event processing and analysis;
- The development of the UsaGraph model (SIQUEIRA; BALDOCHI, 2018), which brings the power of graph theory to web analytics;
- The Footstep page labeling, giving flexibility and customization to the analysis.

By the obtained results, Footstep System has potential to be adopted by companies with conversion and user behavior analysis concerns.

6.1 Future Work

Aiming at encouraging research in user behavior analysis and to contribute for the development of existing work, the following suggestions are presented for future work:

1. Footstep validation in high traffic website for long time, at least a week, in order to stress test the Footstep architecture
2. Automatic identification of page labels based on element appearance in different pages;
3. Evolution of Footstep to take pages' screenshots to enrich analysis of element audience and conversion;
4. Extend Footstep JavaScript library to collect more than click and tap events, in order to enrich user behavior analysis;
5. Development of a prediction model based on artificial intelligence in order to infer user behavior based on past events;
6. Development of a dashboard for comparing navigation and conversion over time.

Bibliography

AKKUS, I. E. et al. Non-tracking web analytics. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2012. (CCS '12), p. 687–698. ISBN 978-1-4503-1651-4. Disponível em: <http://doi-acm-org.ez38.periodicos.capes.gov.br/10.1145/2382196.2382268>. Cited in page 23.

ALQURASHI, T.; WANG, W. A graph based methodology for web structure mining-with a case study on the webs of uk universities. In: ACM. *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*. [S.l.], 2014. p. 40. Cited in page 25.

APACHE NIFI. *An easy to use, powerful, and reliable system to process and distribute data*. 2006. Available: <https://nifi.apache.org/>. Cited in page 46.

APACHE NIFI. *Apache NiFi in Depth*. 2018. Available: <https://nifi.apache.org/docs/nifi-docs/html/nifi-in-depth.html>. Cited in page 46.

BERNASCHINA, C. et al. A big data analysis framework for model-based web user behavior analytics. In: SPRINGER. *International Conference on Web Engineering*. [S.l.], 2017. p. 98–114. Cited in page 20.

BROOKMAN, J. et al. Cross-device tracking: Measurement and disclosures. *Proceedings on Privacy Enhancing Technologies*, De Gruyter Open, v. 2017, n. 2, p. 133–148, 2017. Cited in page 21.

CATTUTO, C. et al. Time-varying social networks in a graph database: a neo4j use case. In: ACM. *First international workshop on graph data management experiences and systems*. [S.l.], 2013. p. 11. Cited in page 27.

CHEN, J.; LIU, W. Research for web usage mining model. In: IEEE. *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*. [S.l.], 2006. p. 8–8. Cited in page 25.

CHEN, W.; WANG, H.; ZHANG, X. An optimized distributed olap system for big data. In: IEEE. *Computational Intelligence and Applications (ICCIA), 2017 2nd IEEE International Conference on*. [S.l.], 2017. p. 36–40. Cited in page 28.

CLARK, S.; BLAZE, M.; SMITH, J. M. Smearing fingerprints: Changing the game of web tracking with composite privacy. In: SPRINGER. *Cambridge International Workshop on Security Protocols*. [S.l.], 2015. p. 178–182. Cited in page 23.

COOLEY, R.; MOBASHER, B.; SRIVASTAVA, J. Web mining: Information and pattern discovery on the world wide web. In: IEEE. *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*. [S.l.], 1997. p. 558–567. Cited 2 times in pages 23 e 25.

COOLEY, R.; MOBASHER, B.; SRIVASTAVA, J. Data preparation for mining world wide web browsing patterns. *Knowledge and information systems*, Springer, v. 1, n. 1, p. 5–32, 1999. Cited in page 23.

COOPER, A. et al. *Privacy considerations for internet protocols*. [S.l.], 2013. Cited in page 22.

CRAZYEGG. *Crazy Egg Website Optimization | Heatmaps & A/B Testing*. 2005. Available: <https://www.crazyegg.com/>. Cited in page 16.

DIGITAL ANALYTICS ASSOCIATION. *Digital Analytics Association*. 2012. Available: <https://www.digitalanalyticsassociation.org>. Cited in page 26.

EIRINAKI, M.; VAZIRGIANNIS, M.; KAPOGIANNIS, D. Web path recommendations based on page ranking and markov models. In: ACM. *Proceedings of the 7th annual ACM international workshop on Web information and data management*. [S.l.], 2005. p. 2–9. Cited in page 27.

ERMAKOVA, T. et al. Web tracking-a literature review on the state of research. 2018. Cited 4 times in pages 9, 20, 21 e 23.

FALAHRASTEGAR, M. et al. Tracking personal identifiers across the web. In: SPRINGER. *International Conference on Passive and Active Network Measurement*. [S.l.], 2016. p. 30–41. Cited in page 22.

FAYYAD, U. M. et al. Advances in knowledge discovery and data mining. the MIT Press, 1996. Cited in page 24.

FOURIE, I.; BOTHMA, T. Information seeking: an overview of web tracking and the criteria for tracking software. In: EMERALD GROUP PUBLISHING LIMITED. *Aslib Proceedings*. [S.l.], 2007. v. 59, n. 3, p. 264–284. Cited in page 20.

GAUR, L. et al. Google analytics: A tool to make websites more robust. In: ACM. *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*. [S.l.], 2016. p. 45. Cited in page 28.

GILL, P. et al. Follow the money: understanding economics of online aggregation and advertising. In: ACM. *Proceedings of the 2013 conference on Internet measurement conference*. [S.l.], 2013. p. 141–148. Cited in page 23.

GOOGLE ANALITICS. *Analytics Tools & Solutions for Your Business*. 2005. Available: <https://www.google.com/analytics/>. Cited 2 times in pages 16 e 41.

GOOGLE TAG MANAGER. *Google Tag Manager*. 2012. Available: <https://tagmanager.google.com/>. Cited in page 63.

GROEF, W. D. et al. Flowfox: a web browser with flexible and precise information flow control. In: ACM. *Proceedings of the 2012 ACM conference on Computer and communications security*. [S.l.], 2012. p. 748–759. Cited in page 23.

GROUPON BRAZIL. *GROUPON Serviços Digitais, LTDA*. 2017. Available: <https://www.groupon.com.br>. Cited in page 62.

GÜNDÜZ, Ş.; ÖZSU, M. T. A poisson model for user accesses to web pages. In: SPRINGER. *International Symposium on Computer and Information Sciences*. [S.l.], 2003. p. 332–339. Cited 2 times in pages 15 e 27.

HASSAN, N.; HIJAZI, R. *Digital Privacy and Security Using Windows: A Practical Guide*. 1st. ed. Berkely, CA, USA: Apress, 2017. ISBN 1484227980, 9781484227985. Cited in page 22.

HERNÁNDEZ, S. et al. Analysis of users' behavior in structured e-commerce websites. *IEEE Access*, v. 5, p. 11941–11958, 2017. Cited in page 16.

HOTJAR. *Hotjar*. 2014. Available: <https://www.hotjar.com/>. Cited in page 16.

IITSUKA, S. et al. Inferring win-lose product network from user behavior. In: ACM. *Proceedings of the International Conference on Web Intelligence*. [S.l.], 2017. p. 426–433. Cited in page 27.

IKRAM, M. et al. Towards seamless tracking-free web: Improved detection of trackers via one-class learning. *Proceedings on Privacy Enhancing Technologies*, De Gruyter Open, v. 2017, n. 1, p. 79–99, 2017. Cited in page 23.

KAUR, K.; SINGH, H. Click analytics: What clicks on webpage indicates? In: IEEE. *Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on*. [S.l.], 2016. p. 608–614. Cited in page 29.

KAUSHIK, A. *Web Analytics: An Hour a Day*. [S.l.]: John Wiley & Sons, 2007. Cited 2 times in pages 9 e 26.

KELLER, M.; NUSSBAUMER, M. Menuminer: revealing the information architecture of large web sites by analyzing maximal cliques. In: ACM. *Proceedings of the 21st International Conference on World Wide Web*. [S.l.], 2012. p. 1025–1034. Cited in page 25.

KLEFTODIMOS, A.; EVANGELIDIS, G. An overview of web mining in education. In: ACM. *Proceedings of the 17th Panhellenic Conference on Informatics*. [S.l.], 2013. p. 106–113. Cited in page 24.

LAMBERTI, F. et al. Supporting web analytics by aggregating user interaction data from heterogeneous devices using viewport-dom-based heat maps. *IEEE Transactions on Industrial Informatics*, IEEE, v. 13, n. 4, p. 1989–1999, 2017. Cited 2 times in pages 26 e 29.

LI, C.-h.; KIT, C.-c. Web structure mining for usability analysis. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*. [S.l.], 2005. p. 309–312. Cited in page 25.

LIU, G. et al. Repeat buyer prediction for e-commerce. In: ACM. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.], 2016. p. 155–164. Cited in page 15.

LU, Z.; YAO, Y.; ZHONG, N. Web log mining. In: SPRINGER. *Web Intelligence*. [S.l.], 2003. p. 173–194. Cited in page 25.

MASSEGLIA, F.; PONCELET, P.; CICCHETTI, R. An efficient algorithm for web usage mining. *Networking and Information Systems Journal*, Citeseer, v. 2, n. 5/6, p. 571–604, 2000. Cited in page 25.

MAYER, J. R.; MITCHELL, J. C. Third-party web tracking: Policy and technology. In: IEEE. *Security and Privacy (SP), 2012 IEEE Symposium on*. [S.l.], 2012. p. 413–427. Cited 4 times in pages 20, 21, 22 e 23.

MELICHER, W. et al. (do not) track me sometimes: users' contextual preferences for web tracking. *Proceedings on Privacy Enhancing Technologies*, De Gruyter Open, v. 2016, n. 2, p. 135–154, 2016. Cited in page 23.

MOE, W. W.; FADER, P. S. Dynamic conversion behavior at e-commerce sites. *Management Science*, INFORMS, v. 50, n. 3, p. 326–335, 2004. Cited 2 times in pages 15 e 24.

MORE, S. Modified path traversal for an efficient web navigation mining. In: IEEE. *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on*. [S.l.], 2014. p. 940–945. Cited in page 27.

NANNONI, N. *Message-oriented middleware for scalable data analytics architectures*. 2015. Cited in page 45.

NEO4J. *Importing CSV Data into Neo4j*. 2007. Available: <https://neo4j.com/developer/guide-import-csv/>. Cited in page 52.

NEO4J. *Neo4j Graph Platform – The Leader in Graph Databases*. 2007. Available: <https://neo4j.com>. Cited in page 53.

NEO4J'S GRAPH PLATFORM. *The Internet-Scale Graph Platform*. 2007. Available: <https://neo4j.com/product/>. Cited 2 times in pages 9 e 53.

NIKIFORAKIS, N.; JOOSEN, W.; LIVSHITS, B. Privaricator: Deceiving fingerprinters with little white lies. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 24th International Conference on World Wide Web*. [S.l.], 2015. p. 820–830. Cited in page 23.

PATIL, U. M.; PATIL, J. Web data mining trends and techniques. In: ACM. *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*. [S.l.], 2012. p. 961–965. Cited in page 25.

PEIXE URBANO. *Peixe Urbano Web Serviços Digitais, LTDA*. 2010. Available: <https://www.peixeurbano.com.br>. Cited in page 62.

PEIXE URBANO. *Sobre a empresa*. 2010. Available: <https://sobre.peixeurbano.com.br/institucional/sobre-o-peixe-urbano/>. Cited in page 63.

PHANIKANTH, K.; SUDARSAN, S. D. A big data perspective of current etl techniques. In: IEEE. *Advances in Computing and Communication Engineering (ICACCE), 2016 International Conference on*. [S.l.], 2016. p. 330–334. Cited in page 27.

PIERRAKOS, D. et al. Web usage mining as a tool for personalization: A survey. *User modeling and user-adapted interaction*, Springer, v. 13, n. 4, p. 311–372, 2003. Cited in page 24.

PIVOTAL SOFTWARE, INC. *Understanding When to use RabbitMQ or Apache Kafka*. 2017. Available: <https://content.pivotal.io/rabbitmq/understanding-when-to-use-rabbitmq-or-apache-kafka>. Cited in page 45.

PUGLIESE, G. Web tracking: Overview and applicability in digital investigations. *it-Information Technology*, De Gruyter Oldenbourg, v. 57, n. 6, p. 366–375, 2015. Cited 3 times in pages 21, 22 e 23.

RABBITMQ. *RabbitMQ – Messaging that just works*. 2007. Available: <https://www.rabbitmq.com/>. Cited in page 45.

ROESNER, F.; KOHNO, T.; WETHERALL, D. Detecting and defending against third-party tracking on the web. In: USENIX ASSOCIATION. *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. [S.l.], 2012. p. 12–12. Cited in page 20.

RUFFO, G. et al. Walty: A user behavior tailored tool for evaluating web application performance. In: IEEE. *Network Computing and Applications, 2004.(NCA 2004). Proceedings. Third IEEE International Symposium on*. [S.l.], 2004. p. 77–86. Cited in page 24.

SABTU, A. et al. The challenges of extract, transform and loading (etl) system implementation for near real-time environment. In: IEEE. *Research and Innovation in Information Systems (ICRIIS), 2017 International Conference on*. [S.l.], 2017. p. 1–5. Cited in page 27.

SANCHEZ-ROLA, I. et al. The web is watching you: A comprehensive review of web-tracking techniques and countermeasures. *Logic Journal of the IGPL*, Oxford University Press, v. 25, n. 1, p. 18–29, 2017. Cited 3 times in pages 20, 21 e 23.

SCHAFER, J. B.; KONSTAN, J. A.; RIEDL, J. E-commerce recommendation applications. *Data mining and knowledge discovery*, Springer, v. 5, n. 1-2, p. 115–153, 2001. Cited in page 15.

SIQUEIRA, W.; BALDOCHI, L. Leveraging analysis of user behavior from web usage extraction over dom-tree structure. In: ______. [S.l.: s.n.], 2018. p. 185–192. ISBN 978-3-319-91661-3. Cited 4 times in pages 18, 31, 84 e 92.

SRIVASTAVA, J. et al. Web usage mining: Discovery and applications of usage patterns from web data. *Acm Sigkdd Explorations Newsletter*, ACM, v. 1, n. 2, p. 12–23, 2000. Cited in page 24.

SRIVASTAVA, T.; DESIKAN, P.; KUMAR, V. Web mining–concepts, applications and research directions. In: *Foundations and advances in data mining*. [S.l.]: Springer, 2005. p. 275–307. Cited in page 24.

SUDHAMATHY, G. Mining web logs: an automated approach. In: ACM. *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*. [S.l.], 2010. p. 57. Cited in page 25.

TRUCKPAD. *O aplicativo que conecta o caminhoneiro à carga*. 2015. Available: <https://www.truckpad.com.br>. Cited in page 62.

VASCONCELOS, L. G. de; JR, L. A. B. Towards an automatic evaluation of web applications. In: ACM. *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. [S.l.], 2012. p. 709–716. Cited in page 27.

VELÁSQUEZ, J. D.; PALADE, V. *Adaptive web sites: A knowledge extraction from web data approach*. [S.l.]: Ios Press, 2008. v. 170. Cited 2 times in pages 24 e 25.

W3TECHS. *Usage of traffic analysis tools for websites*. 2009. Available: <https://w3techs.com/technologies/overview/traffic_analysis/all>. Cited in page 28.

Appendixes

Appendix A – Publications

This work contributed to academia with an article publication: Leveraging analysis of user behavior from web usage extraction over DOM-tree structure (SIQUEIRA; BALDOCHI, 2018) in the 18th International Conference on Web Engineering.

We are also working on a future journal publication focused in Footstep System architecture.