UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL RESTORI SOARES

A Test Platform for Criteo's E-mail Product

Trabalho de graduação.

Trabalho realizado no Grenoble INP dentro do acordo de dupla diplomação UFRGS - Grenoble INP.

Orientadora brasileira:
Profa. Dra. Renata de Matos Galante
Orientador francês:
Prof. Dr. Sylvain Bouveret

Porto Alegre
2017

RESUMO

Qualidade e segurança é uma preocupação importante de empresas de TI atualmente. Muitos recursos são investidos nisso e empresas que decidem não levar em conta este tema, rapitamente enfrentam consequências. Esta importância é ainda maior em aplicações que impactam milhões de usuários mundialmente todos os dias. Este é o caso das aplicações da empresa Criteo.

Publicidade on-line é um tema muito polêmico mas também um enorme mercado que define a maior parte da web como a conhecemos. Muitos dos serviços que temos por garantido e usamos diariamente (como Google, Facebook, YouTube, entre outros) dependem de publicidade como sua principal fonte de renda. Estes serviços são responsáveis pela maior parte do tráfego na Internet e sem eles a web não seria a mesma, tamanha a importância da publicidade para a web. Entretanto, propaganda on-line não é usualmente vista com bons olhos. Isto acontece porque muitas vezes é exagerada e desinteressante para os usuários. Portanto, ainda mais importante do que simplesmente exibir propaganda na Internet é a necessidade de direcionar a publicidade correta para o público correto, este é o problema que a Criteo se propõe a resolver.

Assim como qualquer empresa de TI, qualidade e segurança é de extrema importância para a Criteo. Para garantir que seus softwares funcionam adequadamente, diversas medidas devem ser implementadas, pois um pequeno erro pode resultar em muito dano. Para auxiliar a validação de um de seus mais novos produtos – o *Criteo Dynamic E-mail* – uma nova ferramente foi proposta como projeto de estágio. A ferramenta inclui um *website* com o propósito de simular um *website* parceiro da empresa e ser integrada ao conjunto de soluções que compõem o produto *Dynamic E-mail.* A ferramenta é atualmente utilizada por times internos de desenvolvimento de produto para demonstrar e validar novas funcionalidades do *Criteo Dynamic E-mail.* Este trabalho discursa sobre a realização de um estágio, onde o estagiário atuou como desenvolvedor desta ferramenta como parte de um projeto de fim de estudos. Ele inclui uma descrição e avaliação de tudo que foi desenvolvido durante o estágio dentro da própria metodologia de trabalho da empresa, assim como os desafios que surgiram e suas soluções.

Palavras-chave: Publicidade, Web, Big Data, Propaganda, Java, Qualidade e Segurança, Testes, Javascript, Tecnologias distribuídas, E-mail Marketing.

Resumo estendido

Este é um resumo estendido em português para a Universidade Federal do Rio Grande do Sul do trabalho original que segue. O trabalho de conclusão original, em inglês, foi apresentado na escola ENSIMAG do Institut National Polytechnique de Grenoble através do programa de dupla diplomação UFRGS - Grenoble INP.

1 Introdução

O projeto descrito neste trabalho foi desenvolvido durante um estágio de em torno de 6 meses em uma empresa francesa chamada Criteo. Criteo é uma empresa especializada em publicidade online. As suas soluções utilizam uma série de técnicas para decidir se uma determinada publicidade deve ou não ser apresentada para um determinado usuário. Para isso, as empresa rastreia as atividades do usuário na web através do uso de cookies, com a autorização do próprio usuário.

Entre estas soluções está o produto *Dynamic E-mail*, também chamado de *E-mail Retargeting*, o qual utiliza as técnicas de rastreamento para decidir se um e-mail de publicidade deve ou não ser enviado a um usuário. Essa técnica possui uma alta taxa de sucesso por se tratar de uma abordagem mais personalizada.

O projeto desenvolvido durante este estágio se trata de uma ferramenta de testes para testar a plataforma que é responsável pela decisão de mandar ou não um e-mail para um usuário. Ele inclui uma aplicação *web* que funcionará como um site falso de e-commerce onde as ações do usuário são rastreadas e enviadas para a plataforma para serem processadas. O projeto compreende a aplicação *web* e todos os desenvolvimentos que tiveram de ser feitos para integrá-la à plataforma.

Este relatório pretende mostrar uma perspectiva geral do estágio incluindo detalhes sobre o projeto e seus desafios de implementação, assim como as expectativas por parte da empresa e a experiência adquirida pelo estagiário. Ele também arpesenta uma descrição técnica do projeto incluindo as tecnologias que foram utilizadas, a arquitetura da aplicação e os diferentes desafios de implementação que surgiram durante seu desenvolvimento.

2 Contexto

Este capítulo irá apresentar o contexto no qual o estágio esteve inserido. Ele apresentará a empresa e todos os conceitos necessários para entender o seu modelo de negócio e o do produto ao qual o projeto está relacionado.

2.1 A Empresa

Criteo é uma empresa com uma taxa de crescimento impressionante. Seus clientes incluem grandes empresas como Macy's, Walmart e Adidas. Atualmente é negociada publicamente na NASDAQ e possui mais de 2.000 funcionários.

Para compreender melhor como os produtos da Criteo funcionam e qual o papel da companhia na publicidade online é necessário primeiramente definir os diferentes atores deste mercado. Os dois principais atores do mercado são os anunciantes e os editores:

- Anunciantes são empresas que desejam realizar alguma publicidade. Seu objetivo principal é aumentar suas vendas através da propaganda. Elas geralmente escolhem suas campanhas de acordo com o perfil do seu cliente.

- Editores são os responsáveis por mostrar a publicidade ao público. Eles são os recebedores principais do dinheiro gerado pela publicidade. Em geral, no caso da publicidade online, são sites com bastante tráfego como redes sociais, blogs, sites de notícias.

A Criteo age como um intermediário entre anunciantes e editores. A empresa compra oportunidades de publicidade de editores e as vende para anunciantes. Ela faz isso de diversas formas e o valor agregado que a empresa oferece são os algoritmos que visam maximizar a eficiência das publicidades publicadas.

2.2 Criteo Dynamic E-mail

Uma dos produtos que a Criteo oferece é o *Dynamic E-mail*. A ideia por trás do produto é a de enviar e-mails para clientes que visitaram sites de anunciantes, porém não realizaram de fato uma compra. Os editores, neste caso, são parceiros provedores de endereços de e-mail. Uma vez que a Criteo não guarda em seus bancos de dados nenhuma informação pessoal dos usuários, incluindo seus e-mails, ela deve comprar esta informação de parceiros quando finalmente decide enviar um e-mail. Estes parceiros geralmente são provedores de *newsletters* cujos usuários aceitaram termos onde concordam em ser rastreados e receber e-mails da empresa. A empresa paga um valor por estes endereços de e-mail.

2.3 A Plataforma E-mail

A Plataforma E-mail se refere ao conjunto de tecnologias e algoritmos implementados para tratar a massiva quantidade de dados produzidas pelos usuários rastreados na internet. A Plataforma é capaz de cruzar estes dados com um conjunto de regras pré-definido para decidir se um usuário deve ou não receber um e-mail sobre um determinado produto de um determinado anunciante.

A Plataforma possui uma arquitetura composta por diversos componentes descritos abaixo para que se possa compreender melhor os desafios que o projeto envolveu:

- ***Trackers:*** o começo do *pipeline* que compõe a plataforma é o *tracker*. O *tracker* é um servidor web que trata os eventos produzidos pelo usuário no site de um anunciantes através de requisições HTTP(s). Isto inclui eventos de navegação como visualização de produtos, carrinho de compras, página inicial.

- **Topologias *Storm:*** *Storm* é uma tecnologia open-source utilizada para o tratamento distribuído de *streams*. As topologias *Storm* são responsáveis por tratar, filtrar e analisar todas as sessões de usuários produzidas pelos *trackers* para decidir se um e-mail deve ou não ser enviado para um usuário.

- ***Mailrouter:*** uma vez que as topologias decidiram que um usuário deve receber um e-mail, elas enviam uma requisição ao *Mailrouter*, o qual é responsável por contatar o serviço terceirizado que enviará de fato o e-mail.

- ***Backoffice:*** o *Backoffice* se trata de uma plataforma web onde todos os parâmetros da Plataforma E-mail são configurados, tais como as regras que definem se um usuário deve ou não receber um e-mail de acordo com suas ações no website do anunciante e os templates dos e-mails que serão enviados.

3 O Projeto

O objetivo deste projeto foi criar uma plataforma web que seria utilizada para demonstrar e validar novas funcionalidades desenvolvidas no produto *Dynamic E-mail*. Isto inclui um website que funcionará como ambos editor e anunciante, provendo um conjunto de funcionalidades que permite testar a interação entre estes dois tipos de parceiros. O website deve ser capaz de enviar eventos de rastreamento à Plataforma E-mail. O projeto inclui também todos os desenvolvimentos necessários para integrar o website à Plataforma e para verificar o e-mail resultante enviado pela mesma.

3.1 A Plataforma Proposta

Esta seção apresenta de forma breve a plataforma que foi desenvolvida e as implementações extras necessárias para sua integração.

3.1.1 Funcionalidades do Website

Para que o site de e-commerce pudesse agir como ambos editor e anunciante, um conjunto de funcionalidades tiveram de ser implementados para que o usuário pudesse realizar todas as ações necessárias, são elas:

- Inscrever-se em um *newsletter*

- Anular a inscrição de um *newsletter*

- Visitar página inicial

- Visualizar página de um produto
- Procurar por um produto
- Visualizar carrinho de compras
- Executar o *checkout*

3.1.2 Tecnologias utilizadas

O website foi implementado utilizando a tecnologia *Java Servlets*. *Servlets* são classes Java que possuem um comportamento similar ao de um servidor, ou seja, recebe requisições de clientes e as responde. A classe específica de *Servlets* que se utiliza para implementação de websites é a classe de *Servlets* HTTP, que são capazes de receber requisições via protocolo HTTP e respondê-las com, por exemplo, dados em HTML para que um browser possa visualizá-la em forma de site.

3.1.3 Arquitetura Escolhida

A arquitetura escolhida para a implementação do website implementa o padrão de *design* M.V.C. (*model-view-controller*). Seguindo este padrão, cada página do website é composta por três componentes:

- ***Model:*** *models* são responsáveis por armazenar os dados da página. Uma vez que o website não tinha necessidade de armazenamento persistente dos dados de navegação, os *models* foram implementados utilizados utilisando classes Java normais guardadas na memória do servidor. A tecnologia utilizada permite que estes dados sejam guardados durante toda a sessão de um usuário no website.

- ***Views:*** *views* são responsáveis por implementar as interfaces de usuário do website. Estas foram implementadas utilizando páginas *.jsp* que se tratam de páginas HTML com código Java embarcado. As páginas *.jsp* são então compiladas pelo servidor em *Servlets* HTTP que respondem as requisições com o exato código HTML correspondente.

- ***Controllers:*** *controllers* são os responsáveis por tratar requisições feitas pelo envio de formulários e por enviar os eventos de navegação à Plataforma E-mail. Também implementados na forma de *Servlets*, os *controllers* recebem requisições para cada ação executada pelo usuário no website, as interpreta e envia os dados desta ação aos *trackers* da Plataforma.

3.1.4 Configuração do *Backoffice*

Como já mencionado, o *Backoffice* é uma plataforma web onde anunciantes podem configurar todos os parâmetros da Plataforma E-mail tais como templates de e-mails e regras para envio dos e-mails. Para que o website implementado pudesse ser devidamente integrado à Plataforma E-mail, um novo anunciante teve de ser configurado dentro do *Backoffice* e regras tiveram de ser definidas para que os testes pudessem ser feitos. Esta configuração não envolveu nenhum tipo de implementação de código, porém exigiu conhecimento sobre o funcionamento da Plataforma para que o novo anunciante pudesse ser configurado corretamente.

3.1.5 Catálogo de Produtos

Inicialmente, o website apresentava produtos manualmente inseridos para que um usuário pudesse navegar como em um website de e-commerce. Entretanto, para que os dados sobre os produtos que o usuário está visualizando pudesse ser corretamente interpretado pela Plataforma E-mail, estes produtos deveriam corresponder a produtos de catálogos já armazenados pela Plataforma.

A solução para este problema foi utilizar a mesma API de catálogo de produtos que a própria Plataforma utiliza (utilizada também por outros produtos da Criteo), para selecionar um conjunto de produtos a serem apresentados no website. Para isso, quando o website é colocado no ar, ele contacta esta API e recupera dados de um conjunto de produtos a serem exibidos e cujos dados podem ser corretamente interpretados pela Plataforma E-mail.

3.1.6 Provedor de serviço de E-mail *Catch-all*

Finalmente, para que o teste da Plataforma E-mail seja completo, é necessário verificar o e-mail que foi enviado pela mesma.

Para que isso acontecesse, um servidor implementado na linguagem Scala que faz parte do *Mailrouter* teve de ser adaptado para interceptar todas as requisições de envio de e-mail provindas de sessões no website. Este servidor foi implementado utilizando um filtro que analisava o endereço de e-mail do remetente e verificava se era o mesmo do anunciante configurado no *Backoffice* para o website. Quando este era o caso, o servidor enviava o e-mail para uma caixa de e-mails especial para que os testadores pudessem verificar o e-mail resultante enviado.

4 Resultados

Os desenvolvimentos foram feitos de forma incremental, funcionalidade após funcionalidade, o que é muito característico de metodologias ágeis, com as quais o projeto foi realizado. O projeto estava funcional desde sua primeira integração com a Plataforma e já podia ser usado para testes desde que os testadores verificassem os dados internos da Plataforma. Entretanto o objetivo do projeto era de possibilitar testes do tipo *end-to-end*, ou seja: realizar ações no website e verificar os e-mais enviados. Isto foi possível após a última implementação realizada (Provedor de serviço de E-mails *Catch-All*).

O website foi utilizado, durante o estágio, para um caso de uso real, onde desenvolvedores da empresa que trabalham no produto *Dynamic E-mail* tiveram de demonstrar uma nova funcionalidade desenvolvida e utilizaram o website pois se tratava de uma maneira mais próxima da realidade e mais compreensível para os responsáveis pelo produto. Este caso de uso demonstra o sucesso do projeto pois este foi desenvolvido exatamente para este tipo de teste e demonstração.

5 Metodologia de Trabalho

A implementação do projeto foi de exclusiva responsabilidade do estagiário, entretanto o mesmo foi realizado dentro de um time que trabalha utilizando a metodologia Scrum, então o desenvolvimento seguiu as diretrizes desta metodologia.

5.1 *Scrum*

Scrum é uma metodologia que segue os princípios ágeis. Ele define um conjunto de cargos, responsabilidades e reuniões para guiar o trabalho de uma equipe de desenvolvimento de software. Uma característica comum do Scrum é o pequeno tamanho dos times de desenvolvimento. O trabalho nestes times são divididos em etapas incrementais camadas *Sprints*.

Compostos geralmente por poucas semanas de trabalho, *Sprints* são ciclos que guiam o processo de integração contínua. Os times de desenvolvimento, no *Scrum*, devem entregar ao final de cada *Sprint* um produto utilizável com as funcionalidades desenvolvidas naquele *Sprint*.

Outras características da metodologia *Scrum* são:

- ***Daily Meetings:*** reuniões diárias onde os membros do time reportam o trabalho realizado no dia e definem o que será feito no dia seguinte.
- **Quadro *Kanban*:** mural onde as tarefas a serem feitas são organizadas em diversas etapas definidas pelos próprios times. As etapas mais comuns são ”a serem feitas”, ”em progresso”e ”prontas”.
- **Gráficos de *Burn-down*:** um gráfico atualizado diariamente que estima o total de horas restantes para que todas as tarefas de um *Sprint* sejam completadas.

Ao final de cada *Sprint*, uma série de reuniões é executada para que o time possa realizar uma auto-avaliação e planejar o próximo *Sprint*, são elas:

- **Demonstrações:** reuniões onde o time demonstra aos responsáveis do produto o funcionamento das funcionalidades desenvolvidas durante o *Sprint*.
- **Retrospectivas:** reuniões onde os membros do time realizam uma auto-avaliação do *Sprint* anterior e definem ações a serem realizadas para melhorar a execução no próximo *Sprint*.
- **Planejamentos:** reuniões onde os times avaliam a capacidade de trabalho disponível do time para o próximo *Sprint* considerando todas as férias e feriados, podendo assim definir quantas e quais tarefas e funcionalidades serão executadas no *Sprint* seguinte.

O projeto foi implementado utilizando a metodologia *Scrum* e todas as reuniões e ferramentas apresentadas nesta seção foram utilizadas durante seu desenvolvimento.

6 Conclusão

O projeto proposto se tratou do desenvolvimento de uma aplicação web que simula um website de e-commerce e está integrada à plataforma responsável pela execução do produto *Dynamic E-mail* da Criteo. Sua execução apresentou desafios de diferentes níveis de complexidade e contextos. Envolveu também diversas tecnologias que tratam não apenas do desenvolvimento web mas também de tratamento de dados em grande escala de forma distribuída, o que é uma parte essencial do modelo de negócios da Criteo.

Apesar de sua complexidade, o projeto foi considerado um sucesso pela empresa. Todas as funcionalidades exigidas foram implementadas e devidamente demonstradas aos responsáveis do projeto e validadas pelos mesmos. A aplicação em seu estado final juntamente com os desenvolvimentos extras necessários para integrá-la à Plataforma Email gerou bons resultados e foi validada através de um caso de uso real.

Trabalhos futuros incluem o desenvolvimento de novas interfaces capazes de conectar a diferentes partes da Plataforma Email e prover informações diferentes sobre o processamento das sessões realizadas no website. Como o projeto foi implementado seguindo padrões claros de *design* e utilizou boas práticas de engenharia de software, ele pode ser facilmente estendido para incluir novas funcionalidades ou para alterar as existentes, conforme for necessário de acordo com a evolução da própria Plataforma E-mail.

Grenoble INP – ENSIMAG
Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées

End of Studies' Project Internship Pre-Report

Undertaken at Criteo

A Test Platform for Criteo's E-mail Product

Gabriel RESTORI SOARES
3A – ISI

Company

Criteo

32 rue Blanche

75009, Paris

Internship Responsible

Julien ROS

Internship Tutor

Sylvain BOUVERET

Contents

List of Figures

List of Tables

Glossary

API Application Programming Interface.

CC-BY the most permissive of creative commons' licenses, it allow the usage, redistribution and modification of the content regardless of the purpose, as long as credit is given.

confluence third party team collaboration software, used by Criteo as a documentation repository.

cookie unity of information stored on a user's browser for a specific domain.

e-mail platform the full stack of software that composes the E-mail Retargeting Product.

ESP E-mail Service Provider.

IPO Initial Public Offering.

Java EE Java Enterprise Edition.

jetty Web Server technology that runs on Java.

JMOAB Java Mother-of-All-Builds.

JSP JavaServer Pages.

MariaDB open-source SQL database derived from a fork of MySQL database.

MOAB Mother-of-All-Builds.

mockup prototype or demonstration of a proposed design.

MVC model-view-controller pattern.

OKR Objectives and Key Results.

opt-in the action of subscribing to a newsletter.

opt-out the action of unsubscribing to a newsletter.

PFE *Projet de Fin d'Études.*

retargeting the technique of tracking users' activity in order to expose them publicity concerning products that seem interesting for them.

RPC Remote Procedure Call.

servlet Java class responsible for treating HTTP Requests.

Spring open-source application framework for the Java Platform.

topology graph that defines the processing flow of data streams on Storm.

TrackerRedirect a custom tool created on the project to act as a proxy for requests that are directed to the tracker.

tuple standard data structure used by Storm.

UI User Interface.

Abstract

Quality and assurance is an important concern of IT companies nowadays. A lot of resources are invested on it and the companies that decide not to take it seriously into account rapidly face consequences. This importance is even bigger on applications that impact millions of users worldwide everyday. This is the case of Criteo.

On-line advertising is a very polemical subject but also an enormous market that defines most of the web as we know. Many of the services we take for granted and use daily (such as Google, Facebook, YouTube among others) rely on advertising as their main or only source of revenue. These services are responsible for most of the traffic in the Internet and without them, the web would not be the same, this is the importance of publicity for the web. However, on-line advertising is not usually seen with good eyes. This happens because sometimes it is exaggerated and uninteresting for the users. Therefore, even more important than simply displaying publicity on the Internet is the need of targeting the right advertisement to the right public, this is the issue that Criteo proposes to solve.

Just as in any IT company, quality and assurance is a big part of Criteo. In order to guarantee its software works properly, several measures should be implemented, since one little mistake can result in a lot of damage. To help the validation of one of its newest products - the Criteo Dynamic E-mail - a new tool was proposed as an internship project. The tool includes a website intended to simulate a partner website and is integrated to the software stack that composes the Dynamic E-mail product. The tool is now used by internal development and product teams to demonstrate and validate features of Criteo Dynamic E-mail. This report discourses about my work as the developer of this tool while undertaking a final studies internship at Criteo. It includes a review of everything that was developed in the internship inside the company's own work-flow, as well as all the challenges that arouse and their solutions.

Keywords

Advertising, Web, Big Data, Publicity, Java, Quality and Assurance, Testing, Javascript, Distributed Technologies, E-mail Marketing.

1 Introduction

In the third year of studies at ENSIMAG, the students participate in an internship program called *Projet de Fin d'Études*. During this program, they stay for a period of around six months doing full-time work in a company or a laboratory and they learn about several aspects of their work. This internship is a conclusion for the students' graduation and thus they are required to hand a report relating their experience and the work done. This document is the report of Gabriel RESTORI SOARES about his internship as the developer of a platform that will be used as a testing platform for Criteo's E-mail Retargeting product.

Criteo is a company specialized in on-line advertising. Their solutions include a wide set of techniques to decide whether or not an advertising should be displayed to a user browsing the web. For this purpose, the company tracks the user's activity on the web through the use of Cookies. This tracking is done anonymously and no sensible information about the user is stored by the company. Among these solutions is the E-mail Retargeting Product, which uses the tracking techniques to decide if a personalized e-mail should be sent to a user. It has a high success rate (in terms of percentage of users that actually interact with the publicity), generating more sales for the advertisers. This happens because it is a more personalized approach.

The project developed in this internship is a test tool for the platform that is responsible for the decision of sending an e-mail. It includes a web application which will work as a fake e-commerce website where the actions of its users are tracked and sent to the platform to be processed accordingly. The project comprehends the web application and all developments needed for integrating it with the platform. The tool will be used by internal teams in order to test and validate new features that are being implemented as well as the existent ones.

This project involves various different web technologies - which will be further detailed - and requires consistent knowledge of the existent back-end system so that the integration works as it should. Besides that, the project is fully integrated in the company's continuous integration systems and was developed according to its usual development pipeline.

Criteo's R&D department works according to the agile framework Scrum. Although the project itself is being developed exclusively by the intern, it is integrated inside a Scrum team and is organized just as any other feature development. That means that the development of the project is organized through iterative cycles of two weeks called sprints, and that a demonstration of the implemented features is done at the end of each sprint to the concerned people (all the Scrum teams that work on the e-mail platform and the company's product stakeholders). In addition to that, several other Scrum guidelines (stand-up meetings, kanban charts, planning and retrospective meetings, etc) are part of the regular workflow of this internship.

This report is intended to present a general perspective of the internship including details about the project and its challenges, as well as the expectations on the company's part and the experience acquired by the intern. It also presents a technical description of the project including the technologies that were used, the architecture of the tool and the different challenges that appeared during its development.

2 Context

This chapter will present the context in which the internship was inserted. It will present the company and all the concepts required to understand its business model and the product to which the project is related. It will also describe in the detail how the E-mail Retargeting Product works and how it is implemented, since it is going to be vital to the understanding of the challenges that the project involved.

2.1 The Company

Criteo[22] is a company with an impressive growth rate. It is a global leader in performance display and reported, on the third quarter of 2015, a total revenue of almost 300 million euros, which represented - at the time - a 54% revenue increase[3]. Currently traded publicly on NASDAQ, the company has over 2.000 employees and its clients include big companies of the retail industry such as Macy's, Walmart and Adidas. Its success is publicly attributed to a retention rate of 90%[3]. Criteo has 31 offices in 30 different cities in several different countries around the world, including three R&D offices in Grenoble, Palo Alto and Paris which host over 400 engineers.

Founded in 2005 in Paris by Jean-Baptiste Rudelle, Franck Le Ouay and Romain Niccoli, the company focused its first four years on Research and Development. Initially, it offered mainly a recommendation-engine that tracked users' browsing in order to recommend cinema movies that might interest them. After two big rounds of investment - one in 2006 and another one in 2008 - summing up to more than ten million euros in investment[4], Criteo launched the product that would become its main source of revenue and drive the increasing growth rate that lately led to the company's Initial Public Offering (IPO) on NASDAQ: the display retargeting solution. To better specify how Criteo's products work and what is the company's role in the on-line advertising market, it is necessary to define the different concepts and business-models that this market involves. The first differentiation to consider is the two main actors of any publicity contract: the publishers and the advertisers.

- Advertisers are companies that want to deliver a publicity. Their main goal is to raise visibility for their brand and generate sales for their products. Advertisers are the main source of revenue on the advertising market and it is very important for them to maximize profits while minimizing costs. They usually choose the publicity campaign they are going to deliver according to the customer profile they want to target. For example, it is illogical for an oil company to target the usual Internet user, since it would be inefficient because only a very small part of this public (e.g. oil refineries owners) would actually be interest in it. Because of that, the big majority of advertisers that buy on-line publicity are retailers that sell to the general public (e.g. clothes sellers and travel companies).

- Publishers are responsible for actually showing the advertisements to the public. They are the end receivers of the money that advertising generates. They are also interested in displaying the right publicity to the right public, for two reasons: they are usually paid on a performance basis (e.g. number of users that clicked on publicity coming from a specific blog); they usually don't want to annoy their public, because it would harm their reputation and decrease their public. On the on-line marketing business specifically, publishers are websites with a high amount of traffic (e.g. social networks, blogs and news websites).

The relationship between these two types of actors can happen in several different ways, and can happen directly or passing through an intermediate actor such as Criteo. Besides that, the fashion in which the money from the advertisers are paid to the publishers can also vary depending on the agreement. Historically the usual payment models used by companies in on-line advertising evolved according to the time-line represented on Table 1.[1]

Table 1: Time-line of the evolution of payment models in on-line advertising

2000	**Pay-per-impression:** in this type of payment, the advertiser pays for every impression of its publicity independently of the number of users that actually click on it.
2008	**Pay-per-click:** in this business model the advertiser only pays - and consequently the publisher only gets paid - when a user clicks on the publicity being led to the advertiser's website. It is one of the most common chosen methods and is the main business model used by Criteo's display solution.
2015	**Pay-per-action:** more precise than the pay-per-click method, the PPA model defines that the transaction is only made when a specific action from the user occurs. This action can be a purchase or any other action the both parts of the deal agreed on. It can also be a combination of actions with different price definitions for each action.

Last but not least, advertisers can also choose the way they intend to target their audience. This targeting can take different forms such as contextual targeting, which is the act of choosing what kind of advertising to display according to the publisher's content (e.g. a sports website would display sport-related advertising). Another type of segmentation is the behavioural targeting, which is the act of choosing what publicity to display by anonymously tracking the user's actions (e.g. publicity about travel promotions would be displayed for users that have been looking into travelling information).

Criteo acts as an intermediate between publishers and advertisers. That means it buys advertising opportunities from publishers and sells them to advertisers [2]. It does that in a myriad of ways and through different payment methods. A good example of how the Criteo's

[1]This is not an exhaustive list and is rather intended to illustrate the concept of payment models.
[2]In this report, both advertisers and publishers are called the company's 'partners'.

algorithms aggregate value to advertising is their targeting solution through RTB (real-time-bidding) platforms.

When seeing an Internet user on a publisher website, Criteo may choose to buy an advertising space in the website by bidding a price through real-time-bidding. The RTB platforms propose to potential advertisers to display a publicity to a specific user, the advertisers send it a bid-value, the advertiser that proposed the highest value actually buys the publicity space, however the actual payed price is equivalent to the second highest bid[3]. This communication is done automatically and usually the RTB platforms have a time restriction (usually 100 milliseconds) for the advertisers to answer. When this happens Criteo process tracked data from the user to decide whether it is or not a possible profitable user and what would be the best price to bid on this specific case. The aggregate value relies on the fact that the advertising is more efficient and displayed to users that have high probability of engaging on a commercial transaction with the advertiser. Taking in consideration that, in this case, Criteo pays on a CPI (cost-per-impression) basis and gets paid on a CPC (cost-per-click) basis, it is important to calculate well this probability so it can bid the best value and profit from it. Due to the fact that it is done in real time and for millions of Internet users, performance is very important and several big data technologies are employed in the algorithms.

2.2 Criteo Dynamic E-mail

In February of 2014, Criteo did its first acquisition after its IPO and acquired a company called Tedemis[2]. Tedemis offered a solution of e-mail marketing that sends personalized e-mails to users. This platform became a new product on Criteo's portfolio called Dynamic E-mail developed within the company's R&D office at Grenoble.

The way this product works is similar to the display approach, by tracking a user's actions through Cookies, the platform can decide whether or not to send an e-mail to this user about a single product or a group of them. In this case, the publishers do not provide spaces on their websites for displaying publicity but provide the e-mail addresses from the users Criteo is interested in.

More precisely, when a user - who has subscribed to a newsletter and agreed to receive e-mails from Criteo - opens an e-mail from a publisher's newsletter, Criteo drops a Cookie on the user's browser and stores a MD5-encrypted version of the user's e-mail address in its databases. Thanks to the id that is stored in the Cookie and the encrypted address, the company can track the user's activity on partners' websites.

By doing this in real time, Criteo is able to send an e-mail to the user after their session on the partner's website is over. Since Criteo doesn't store the unencrypted version of the user's e-mail address, the company has to contact the publisher to get the actual electronic address of the user. Whenever an e-mail is sent, the company pays a price for the addresses to the publishers that provide them. This type of retargeting has a bigger click rate (around 10% of the e-mails sent are clicked).

[3]This system was introduced by Google to prevent advertisers from spending more than necessary.

2.3 The E-mail Platform

The E-mail platform refers here to the whole set of technologies and algorithms that run on Criteo servers to process the data sent by advertisers and publishers in order to trigger an e-mail send. Currently, there are four scrum teams that work on the e-mail R&D department: the Deliverability team, which works on the tools to improve the chances of the e-mails arriving to the receiver in-box (e.g. to prevent them from being considered as spam); the services team, which works on the tools that communicate with services external to the E-mail platform (e.g. the third party services that send the e-mail); the Targeting team, which works mainly on the tools that manage the decision of sending an e-mail; and lastly the Core team, which deals with the core back-end features of the platform (e.g. databases, statistics) and in which this internship is undertaken.

2.3.1 An Overview

The platform has an architecture composed of several components which will be briefly described here in order to provide a better understanding of the challenges that were faced during the project's development. A better understanding of the whole functioning of the platform may be achieved by consulting Figure 1.

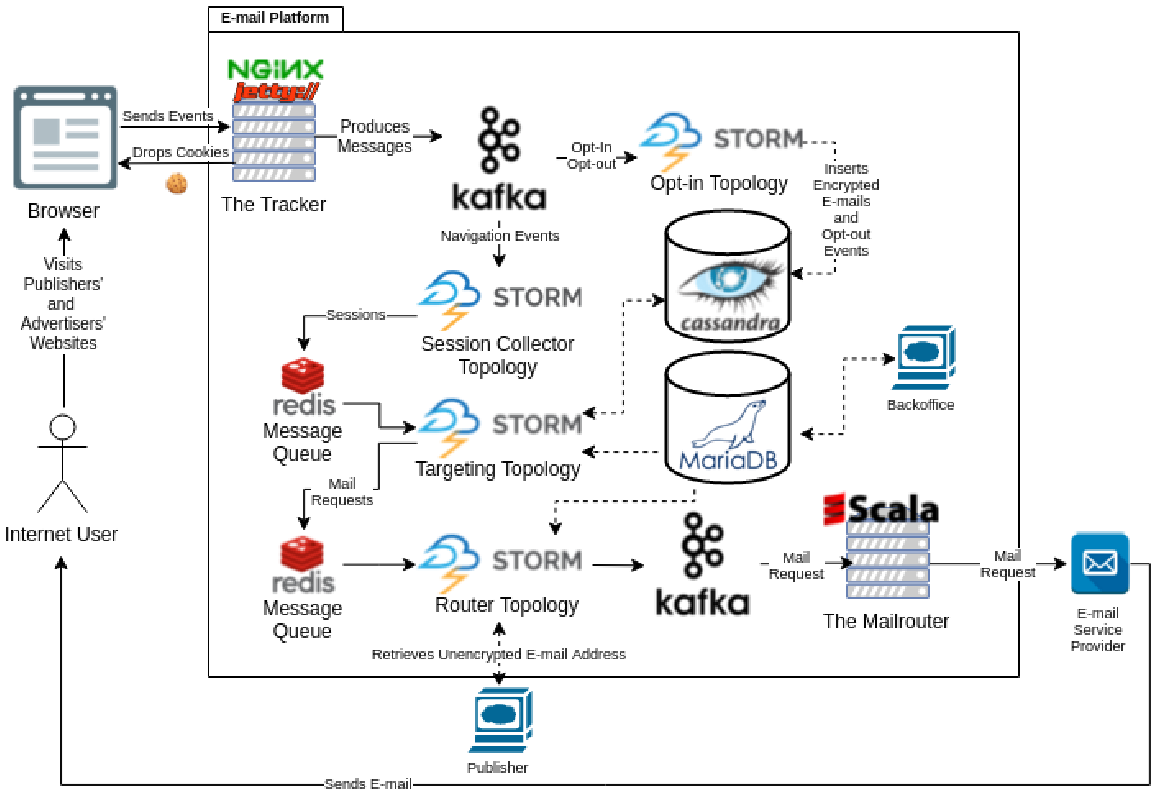


Figure 1: Overview of the e-mail platform's architecture

2.3.2 Trackers

The beginning of the software pipeline that compose the platform are the web-servers. The tracker is a web-server that treats the events coming from the users' web browser in form of HTTP(s) requests. This includes the navigation events that occur on the advertisers' websites (e.g. homepage view, product page view, shopping cart view, etc). This kind of events come from a solution called OneTag, which was Criteo's standard solution for tracking users on the Performance Display product and was integrated to the E-mail product after Criteo acquired Tedemis.

Apart from the OneTag events, the tracker is also responsible for treating events that are specific to the E-mail platform, such as the subscription and unsubscription events as well as events that indicate when a user opened an e-mail or clicked in the link that lead to an advertiser's website. Particularly in the case of subscription and unsubscription events, the web-servers are also responsible for dropping and deleting, respectively, the cookies used for tracking a user's activity on his browser.

The web-servers are currently implemented using two different web-server technologies, Jetty and Nginx[18]. Nginx is a web-server technology implemented in C known for its high performance and is used to treat all the OneTag events. Jetty is a Java based servlet container and is used to treat the events that are specific to the E-mail platform.

When a web-server receives an HTTP request, it prepares and sends the event data forward in the pipeline via the messaging system Kafka[7]. Kafka is a open-source distributed messaging system supported by the Apache foundation. It uses the publish-subscribe pattern and is used in different parts of the E-mail platform to implement message queues. It works through the concept of *topics*[14]. Each topic is a partitioned commit log which is stored on disk and replicated in order to provide fault tolerance. Several *producers* can emit messages to a same topic which can then be consumed by several *consumers* concurrently. Kafka is highly scalable.

2.3.3 Storm Topologies

Once the events are treated by the tracker and the event information is sent via Kafka, they can be consumed and processed in order to decide whether or not to send an e-mail. The processing of the events in the E-mail platform is done using a technology called Storm.

Storm[8] is a real time distributed computation system frequently used for processing heavy streams of data. It employs the topology concept to represent processing units. A topology is basically a graph that defines the flow of processing of a data stream. They are composed by two main components: spouts and bolts[12]. Spouts are responsible for the source of the stream, they capture raw input (e.g. from a Kafka message queue) and transfer this input in the form of tuples. Bolts are responsible for the processing of the tuples on a stream, they usually receive tuples from a spout or another bolt and execute some kind of processing or transformation before transferring the tuple to the next step on the topology. All of it is highly distributable and can be scaled according to the size of the clusters where it is running.

Storm provides an API for several different programming languages; the topologies of the E-mail platform are implemented in Java. Currently, there are eight topologies on the platform; each of the topologies has a different responsibility on the pipeline. The following list describes the main topologies that work on the processing of the events in order to send an e-mail:

- **Opt-in Topology:** it is responsible for treating the subscription (known as an opt-in) and unsubscription (known as an opt-out) events. In the case of an opt-in, the topology is responsible for storing the user's id, also stored in its browser with a cookie, associated to its MD5 encrypted e-mail. This way, the platform can recognize a tracked user and send a request to a publisher for his e-mail address if it decides to send an e-mail. In the case of the opt-out, the topology is responsible for storing the fact that a user should not be considered for receiving e-mails.

- **Session collector Topology:** the platform only processes the user events after he stopped navigating on an advertiser's website. It considers that it happened when it did not receive events for a certain user for 25 minutes, the set of events triggered while the user was navigating is called a session. The job of the session collector topology is to build these sessions by aggregating the received events from a user and, once the 25 minutes period of wait is attained, sending the session to a queue to be consumed by the next step of the pipeline, the targeting topology. Figure 2 displays the architecture of this topology and exemplifies how a Storm topology is built using spouts and bolts.

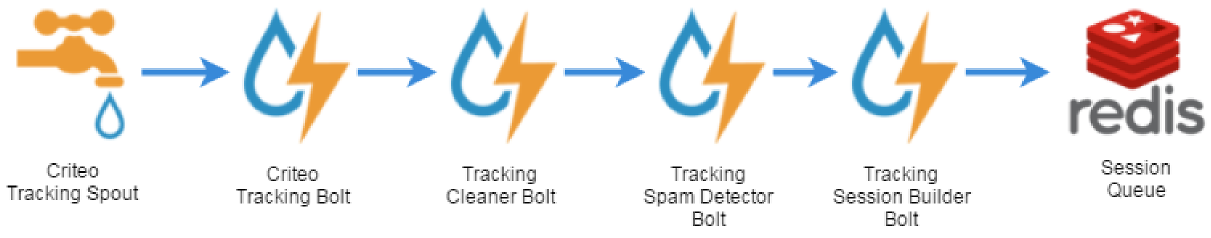


Figure 2: Example of Storm topology (Session Collector Topology)

- **Targeting Topology:** the way the platform decides which sessions should generate an e-mail send is by matching the sessions to rules that the advertisers themselves configured - the configuration is done using a tool called Backoffice, which is further detailed. The targeting topology is responsible for the whole process of deciding if a session should generate an e-mail send, from deciding if a user is eligible to matching the session to a rule and retrieving the information needed to send the e-mail such as the products that the e-mail will contain. If the topology matches a session to a rule and decides to send an e-mail, it sends all the aggregated information of the session to the router topology using yet another queue.

- **Router Topology:** the sessions sent to the router topology are the ones that should generate an e-mail send. This topology is responsible for retrieving the real e-mail address from a user (by sending a request to the publisher that provided its MD5 hash) and building an e-mail according to a template that is also configured on the Backoffice - each rule is associated to a template. Once the address is retrieved and the e-mail is built, the topology sends this information via Kafka to the Mailrouter, which is the tool that contacts third party service responsible for actually sending the e-mail.

The list is not exhaustive and does not include topologies that don't work on the process of deciding to send an e-mail, such as the topologies responsible for storing analytical metrics.

2.3.4 Data Persistence

Regarding the processing that the Storm topologies do, it is also important to mention other technologies involved in it such as Redis[10] and Cassandra[9]:

- Redis is an in-memory database that is used for cache purposes and messages queues, for example it is through Redis that the sessions' information is sent from the session collector topology to the targeting topology and from the targeting topology to the router topology. The session collector and targeting topologies also use Redis as a cache memory.

- Cassandra is the main database used for persistent high volume data in the platform. For instance, all the subscription and unsubscription events are stored in Cassandra, as well as the navigation events and the associations between internal user ids, their cookies and their MD5-encrypted e-mails.

2.3.5 Mailrouter

Once the processing of the router topology has finished, the platform has already the full information needed for sending an e-mail, including the user's unencrypted e-mail address and the e-mail content. It is the job of the Mailrouter sending the request to the contracted E-mail Service Provider (ESP). The architecture of the Mailrouter is presented on Figure 3.

The Mailrouter is implemented in the programming language Scala[24] and uses the open source Remote Procedure Call (RPC) system Finagle, due to its simplicity for implementing high-concurrency distributed systems.

Some of the features of the Mailrouter are: filtering unwanted e-mails (e.g. too old or duplicated e-mails), contacting the ESP for sending an e-mail, acknowledging to the Storm topologies the status of the request, applying small changes in order to improve the chances for the e-mail not to be selected as spam, retry sending the e-mail if it fails to be dispatched.

The Mailrouter uses a Couchbase[20] database as a cache memory in order to filter duplicated e-mails. Couchbase is a NoSQL distributed database that features different data models such as JSON document and key-value models.

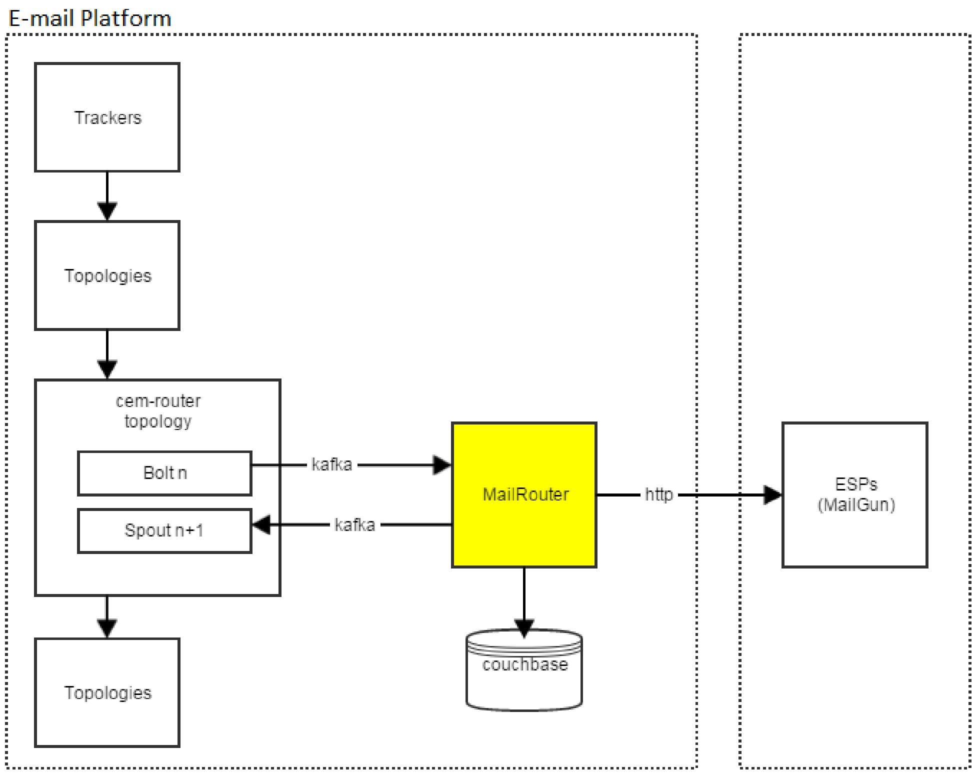


Figure 3: Mailrouter architecture overview

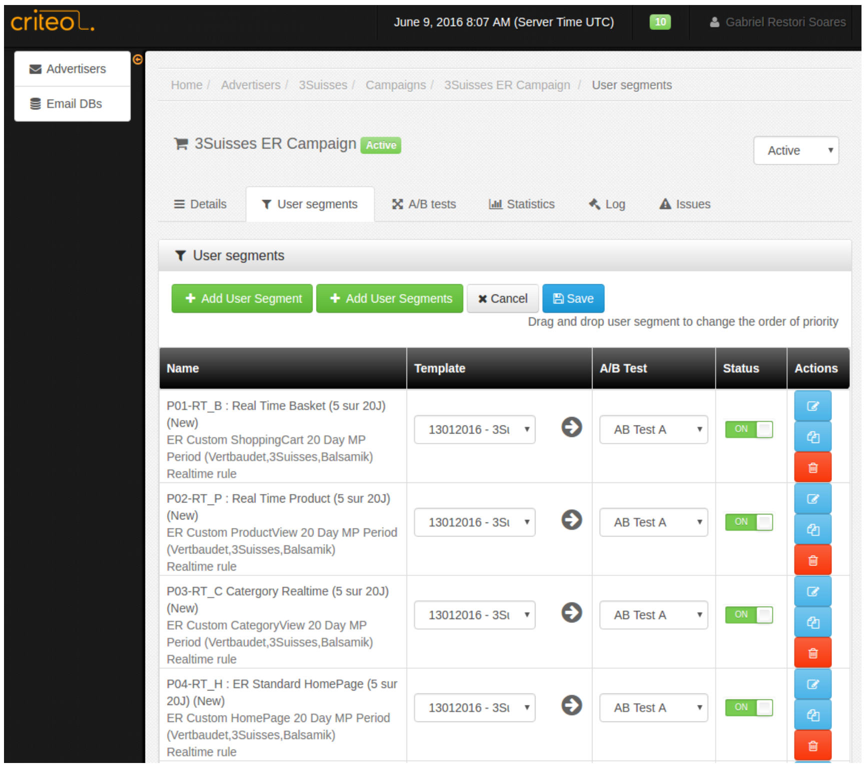
2.3.6 Backoffice

The Backoffice is the front-end application where the parameters of the platform are configured. On it the advertisers are able to configure the public that should be targeted as well as the rules that define when an e-mail should be sent and the appearance of the e-mail. Another example of configuration that can be done on the Backoffice is the definition of the web-service that should be called in order to retrieve a user's unencrypted e-mail. A screenshot of the Backoffice user interface is presented on Figure 4.

It is a web platform implemented in PHP[25] using the framework Symfony[26]. It stores these configurations on a MariaDB database which can be then consulted by the Storm topologies while processing the users' sessions.

2.4 Final Considerations

The context in which this project is inserted is not a simple one. However, all the details of the E-mail platform had to be explained so all the aspects of the project can be understood. Besides that, the context of the company is important to understand the work-flow in which the project was developed and the objectives behind it.


criteo
June 9, 2016 8:07 AM (Server Time UTC)
10
Gabriel Restori Soares
Advertisers
Email DBs
Home / Advertisers / 3Suisses / Campaigns / 3Suisses ER Campaign / User segments
3Suisses ER Campaign Active
Active
Details
User segments
A/B tests
Statistics
Log
Issues
User segments
Add User Segment
Add User Segments
Cancel
Save
Drag and drop user segment to change the order of priority

Name	Template	A/B Test	Status	Actions
P01-RT_B : Real Time Basket (5 sur 20J) (New) ER Custom ShoppingCart 20 Day MP Period (Vertbaudet,3Suisses,Balsamik) Realtime rule	13012016 - 3Su	AB Test A	ON	
P02-RT_P : Real Time Product (5 sur 20J) (New) ER Custom ProductView 20 Day MP Period (Vertbaudet,3Suisses,Balsamik) Realtime rule	13012016 - 3Su	AB Test A	ON	
P03-RT_C Catergory Realtime (5 sur 20J) (New) ER Custom CategoryView 20 Day MP Period (Vertbaudet,3Suisses,Balsamik) Realtime rule	13012016 - 3Su	AB Test A	ON	
P04-RT_H : ER Standard HomePage (5 sur 20J) (New) ER Custom HomePage 20 Day MP Period (Vertbaudet,3Suisses,Balsamik) Realtime rule	13012016 - 3Su	AB Test A	ON	

Figure 4: *Screenshot* of the Backoffice containing an advertiser's rules configuration

3 Project

The project described in this chapter was developed during an internship inside the developer team responsible for Criteo's Dynamic E-mail product. The internship worked as a final studies project for ENSIMAG. One of the main criteria for the internships to be validated by the school is for it to include the development of an actual project from scratch such as this. This chapter will cover the details of a web platform that was implemented by the intern to work as a test platform for Dynamic E-mail product, including a description of its architecture, all of its components, its architecture and some extra developments that were needed for the platform work as intended.

3.1 Overview

The objective of the project is to create a platform employed to demonstrate and validate Dynamic E-mail's features. This includes a website that will work both as an advertiser and a publisher, providing a set of features that allows the testing of the interaction with these two kinds of partners. This website should be capable of sending tracking events to the Dynamic E-mail's main back-end platform, here called the E-mail platform. The project also includes all the developments needed for integrating the website to the E-mail platform and for verifying the resulting e-mail sent by it.

One of the ways that the E-mail platform is tested is by deploying it in a controlled environment called the pre-production environment. The pre-production acts as a replica of the production environment that treats the real web traffic and triggers the send of an actual e-mail. The pre-production receives a part of the real traffic that is being sent to the production and treats it equally, but the main difference is that the e-mails that are triggered are not actually sent, this way Criteo can verify statistics to assure the functioning of the platform in a restricted environment.

Every time a new development occurs on the E-mail platform, a new version as a whole is created by building and packaging the whole software suite and its dependencies. This creates a new version number which can then be used to retrieve the software from the company's repositories and deploy them (the version control system Criteo uses is further detailed). Before a new version is deployed on the production environment it can be tested on pre-production. So, in order to use the website as a testing and validation platform, it should be running and integrated to the pre-production environment. It can then be accessed by anyone who wishes to use it for testing the platform, including the stakeholders of Criteo e-mail (product owners and product managers).

3.2 Proposed Platform

Once the overall functioning of the E-mail platform is understood, it gets easier to understand how the website can be used for testing it. Since, most of the tests of the platform are done automatically, the website isn't intended to test the fine details of technical solutions. It will mostly serve the product and business areas of the company that constantly have to test the platform in a case-by-case basis. The following sections will present a technical view of all that was implemented, including the website itself and other external developments that were made for the test platform to work as intended.

3.2.1 Website's Features

In order to correctly simulate an advertiser's website, the application should allow the user to execute a defined set of actions when interacting with the website, so the E-mail platform can track these actions and process them accordingly. The features that were implemented on the website are the following:

- **Subscribe to a newsletter:** the e-mail platform only tracks and sends e-mails to users that actively subscribed to a partner's website and agreed on receiving e-mails from Criteo. In order to integrate to the application the full scenario of a user's test case, a subscribe page had to be implemented on the application to represent the *opt-in* event. This event should send the pertinent information about the user (i.e. his encrypted e-mail address) to the platform, so it can start to track the user's actions.

- **Unsubscribe from a newsletter:** not less important than the *opt-in* event, the *opt-out* event occurs when a user unsubscribes from a newsletter, this means an active decision of not receiving anymore e-mails related to a partner's website including e-mails from Criteo. In order to comply with both ethical and governmental rules, when an *opt-out* event occurs, Criteo erases the user's information from its databases and removes the Cookies from the user's browser, in order not to track their actions anymore.

- **View homepage:** since a user's session usually starts with this user viewing the homepage of a partner's website, it is necessary to develop in the application a homepage that sends the corresponding event to the platform.
- **View product:** a strong indicator of whether a user is interested on a product is when he consults the detailed information of a specific product. Some of the e-mails Criteo sends uses this information in order to fill the template with products in which the user was interested. Therefore, a product page had to be implemented to represent this case.
- **View category:** usually, on e-commerce websites, products belong to one or several categories of similar products. It is common for Internet buyers to browse through categories while searching for the right product to them, this gives the platform a lot of information on the user's preferences, so it tracks this kind of actions. Therefore, the category system should also be implemented on the application.
- **Search product:** following the same principle as the category view, a product search also contains a lot of information about the user's intentions. Consequently, the web application should also include a search mechanism.
- **Shopping Cart:** a good example of how the shopping cart information is important to the platform's decision, is when a user adds a certain product to their cart but actually don't end up buying it. This usually indicates that the user is yet uncertain on whether buying the product or not, and in that cases a targeted e-mail can make a great difference.
- **Checkout cart:** although the buy event may seem unimportant, since the main intention is targeting users that still want to buy a product, it is exactly for that reason that it needs to be tracked, so the platform doesn't send e-mails about products that the user has already bought. The buy event is also important to indicate whether or not a purchase had the Criteo's influence, which is very important for the company's metrics.

3.2.2 Technologies

For the application to work as it should and also be easily integrated to Criteo's software factory work-flow, some choices were made regarding the server-side technology. The most used programming language in the E-mail platform is Java. Criteo uses various tools to integrate Java projects and dependencies in their software factory. Therefore, the natural choice for server-side language was Java EE. In addition to that, the web-server that runs it is a Jetty web-server.

Java Enterprise Edition (Java EE) is a platform that provides a set of APIs (Application Programming Interfaces) and a runtime environment for developing several different types of applications including distributed architectures and web services.

It is mainly used because of its servlet API. Servlets are classes that extend the capabilities of a server, this means it can treat and respond to requests. Although the servlet paradigm can be used to treat several types of requests, the most common ones are the HTTP Requests. In order for a class to treat a HTTP Request using Java EE it should extend the *HttpServlet* class; it can override the methods that correspond to the different types of requests (POST, GET, PUT, etc) and manipulate the parameters of request and response to provide functionality. A simple example of servlet is one that responds with a HTML text to a GET request, so it can be accessed by a browser and display a web page.

On the application, the servlets are used to manage the flow of web pages on the site and to trigger events (e.g. adding or removing products from the shopping cart). They are also responsible for handling JSP pages, which are basically HTML pages with embedded Java code that are compiled into servlets that respond the proper HTML page after the execution of the code.

In order to be able to easily deploy and run the website, the Jetty Web Server is used. Jetty can be used as an embedded web server; this means it is able to run directly from the packaged *.jar* file that contains the application.

In order to make Jetty compatible with the usually used directory structure for Java web applications[15], two other modules were added to the project: *jetty-webapp*, *jetty-jsp*. The first one allows Jetty to interpret the standard folder structure and read the configuration file that maps URLs to servlets. The second one enables Jetty to compile JSP pages into servlets.

3.2.3 Architecture

The high level view of the application's functioning is very simple, according to the actions of the user on the website, it will send the necessary events (*opt-in*, tracking events, etc) to the e-mail platform, as presented of Figure 5.

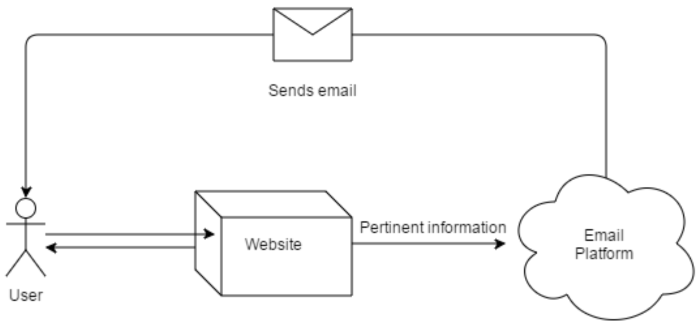


Figure 5: *Black-box view* of the interaction between the website and the user and platform

During the design of the website's architecture, it was decided that the best way to implement it was to use the model-view-controller pattern (MVC) pattern. Following this pattern, each page is composed by three main components:

- **The controllers:** in the application, the controller corresponds to a Java Servlet that will manage the HTTP requests to a given page, the controller may execute any kind of pre-processing according to the request of the page (e.g. previously filter the products when a search happens). The controller is also responsible for managing the request triggered by forms (e.g. the newsletter subscription form). After treating the requests, the controller can then redirect the user to the pertinent view, implemented here by the JSP pages. On the project structure, the controllers correspond to the Java classes that extend the HttpServlet class.

- **The views:** The views correspond to the front-end of the pages. They are implemented using JSP pages. Although the JSP pages are compiled into servlets that respond the requests with the proper HTML, it is much easier to use their HTML's syntax to configure the user interface. They are usually called by a request redirection from their respective controllers after the back-end processing is finished.

- **The models:** they are responsible for storing and managing the application data. They are normal Java classes containing pertinent data. Since this project's scope concerns mainly tests, few models are required (basically only for the list of products and the user's shopping cart). Further, on the data persistence section, it will be explained how models are stored and what is their persistence scope.

This architecture is represented by the diagram on Figure 6.

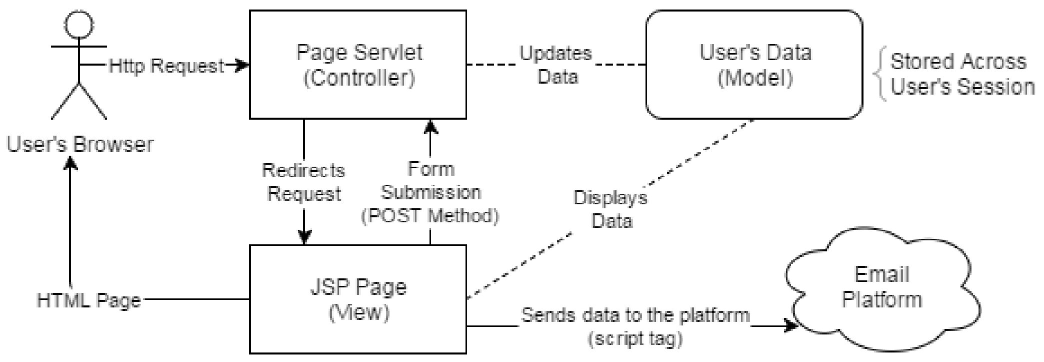


Figure 6: Model-view-controller architecture on the application

An example of how a controller class may be used to treat requests in this architecture can be seen in the following code piece:

```java
public class RemoveProductFromCart extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    HttpSession userSession = request.getSession();

    ShoppingCart userCart = (ShoppingCart) (userSession.getAttribute("shoppingCart"));

    int productIndex = Integer.parseInt(request.getParameter("productIndex"));

    userCart.removeProduct(productIndex);

    response.sendRedirect("./viewShoppingCart?productRemoved=true");
  }
}
```

In this example the servlet class *RemoveProductFromCart* is used to treat HTTP POST requests. When the request is made, the servlet manipulates the model class *ShoppingCart* in order to remove a product from the user's shopping cart. It then redirects the user's browser to the shopping cart page.

3.2.4 Data Persistence

There are two kinds of data on the application, data that is common to all users (i.e. the catalog of products) and the data that is different for each user (i.e. the shopping cart).

The product catalog is stored using the Java EE context mechanism. This means that the data contained in the catalog is available and is the same to every user making requests to the servlets. The context mechanism works by saving the data on a key-value pattern on the web-server's memory, this way it can always be accessed from a servlet.

The shopping cart data is stored on a per user basis using the Java EE session mechanism. This means that the association to the user will be done using a Cookie that is dropped on the user's browser and contains an id that identifies the set of data of each user on the server's memory. Two constraints come with the session approach:

- The data is lost after the session is expired (i.e. the user has not done anything during a predefined time [default: 30 min])
- The data is lost after the user closes the browser (the Cookie containing the session id is deleted by the browser)

3.2.5 User Interface

The UI of the website was defined previously to its implementation. To do that, mockups were created using a diagram tool. After defining a desired UI it becomes simple to write the front-end code of the website.

The screen-flow defined for the usual use cases is presented on Figure 7, it includes the actions that can be executed without triggering a page change (i.e. add or remove product from cart, checkout cart).

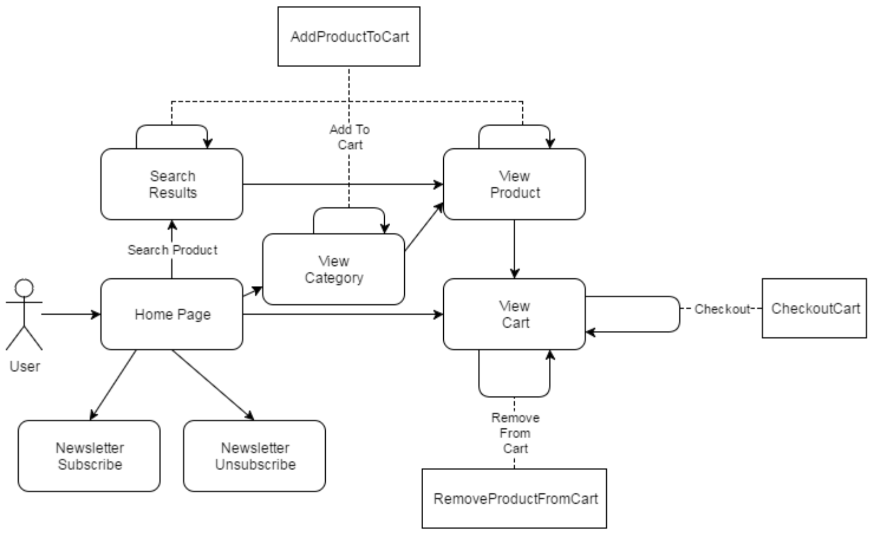
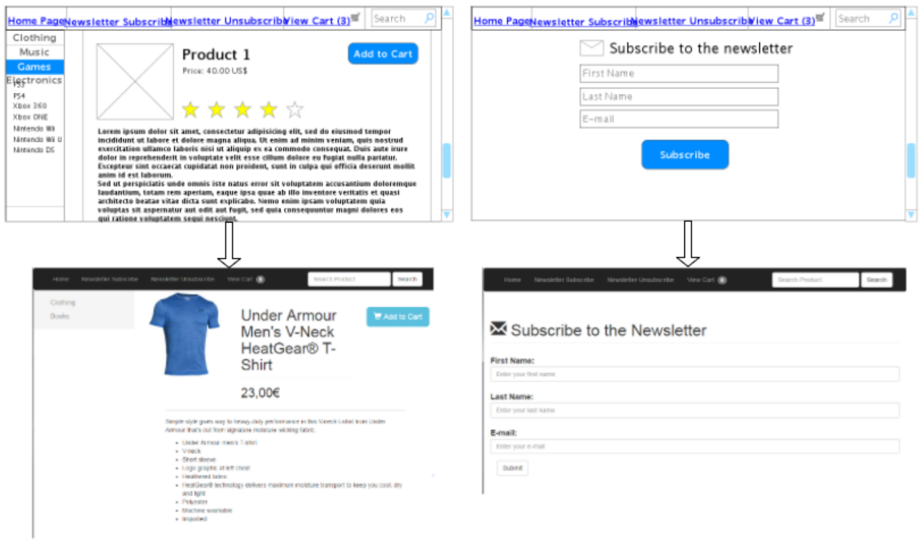


Figure 7: Screen-flow of the web application with user's actions

Due to the increase of the mobile market and Criteo's efforts to include this market on their solutions, it is only reasonable that the application has a responsive UI so it can be tested on mobile devices as well. In order to do that easily and with high maintainability the Bootstrap framework was used. Bootstrap is a open-source framework created on Twitter and licensed under the creative commons license CC-BY 3.0[19]. A comparison between the proposed mockups and the implemented UI is presented on Figure 8.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

Figure 8: Comparison between the initial mockups and the final results of the user interface

3.2.6 Tracker Communication

In order to make this project functional and actually be able to test the e-mail platform with it, it is necessary to integrate it with the running platform according to the environment where it is deployed. This implicated in a series of developments that had to be made on the project for the integration to work as it should.

On the production environment, the integration between advertisers or publishers and the platform is made mainly by Criteo itself, the websites only need to send requests to the platform with the information of each event that happens. For that reason, the tracker, which receives and handle this events is configured to accept only requests from Criteo's e-mail retargeting domain name (er.criteo.com). It also implies that the Cookies used for tracking the users' activities are dropped for this same domain name. This raises some challenges for the integration of the test platform, since it has to run in the pre-production environment and the machines of the pre-production environment do not have this domain name.

The proposed solution for these problems was to implement a redirection tool to "trick" the tracker into accepting the sent requests and to handle the Cookies. This tool is called the "TrackerRedirect" servlet.

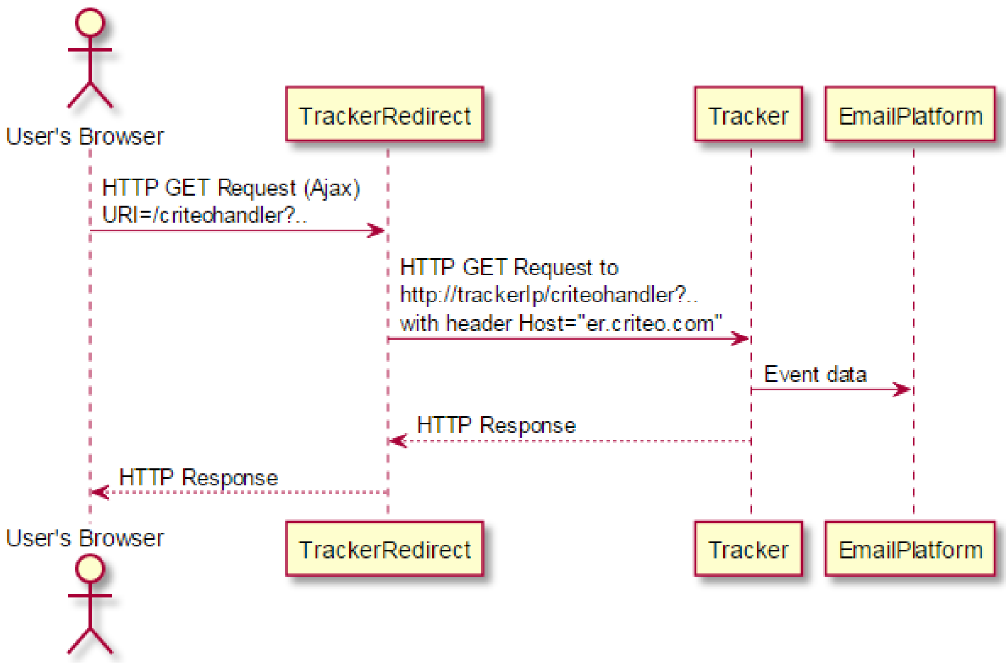


Figure 9: Sequence diagram of the interaction between the website and the tracker

All the events that should be sent to the platform's tracking tool are first sent to the TrackerRedirect with a parameter containing the actual URI that should be requested from the tracker. The TrackerRedirect is responsible for sending the request to the configured tracker's ip address, receive its response and send it back to the user's browser. In doing so, it manipulates the Host header from HTTP request in order for the tracker to believe the request is coming from the right domain name. This sequence is represented on the diagram on Figure 9.

Besides the domain adaptation, the TrackerRedirect tool is also responsible for sending to the tracker the Cookies contained on the user's browser so it can identify the user. It also drops the Cookies sent by the tracker on the user's browser.

3.2.7 Backoffice Configuration

As already mentioned, the Backoffice is the front-end application where advertisers can configure all the settings that define how the platform will process the navigation of their users in their website. Just like for any other advertiser, for the test platform to generate events that can be used for sending e-mails, some settings had to be configured in the Backoffice.

Among the configurations related to a certain advertiser, the most remarkable are the campaigns and their rules. Each campaign may contain one or several rules and can target different audiences (between customers, non-customers and both). The rules define when exactly an e-mail should be sent to a user (e.g when he added products to the cart but didn't buy them) as well as which e-mail template should be used in each case.

For the test platform, a new advertiser entity was defined in the Backoffice and its own campaigns, rules and templates were configured. This way, whenever a feature of the E-mail platform needs to be tested (a real life case is later presented), the tester needs only to configure a rule that will match the specific case of that feature and use the website to confirm that the platform behaves as it should.

3.2.8 Profile API

As previously explained, the Router Topology in the E-mail platform is responsible for retrieving the actual e-mail address of a user once his actions triggered the decision of sending an e-mail. It does that by contacting a publisher's user base through a web-services API.

When an opt-in event occurs, the user's data (encrypted e-mail, internal id, etc) is stored in the platform together with a reference to a customer base. This customer base is configured through the Backoffice and contains the information about the web-services that should be contacted for retrieving the unencrypted e-mails. This is how the platform is able to know which publisher to contact for retrieving the e-mail for a specific user.

Since the test platform simulates both an advertiser and a publisher, a user database should be configured for it and an API should be implemented so the E-mail platform can retrieve the subscribed e-mail addresses, this API was called the Profile API. The communication between the router topology and the Profile API works through HTTP requests containing data encoded in a JSON format. The topology sends a list of MD5 hashes to the API and the API answers with the list of corresponding e-mail addresses and names for the users. This architecture is represented on Figure 10.

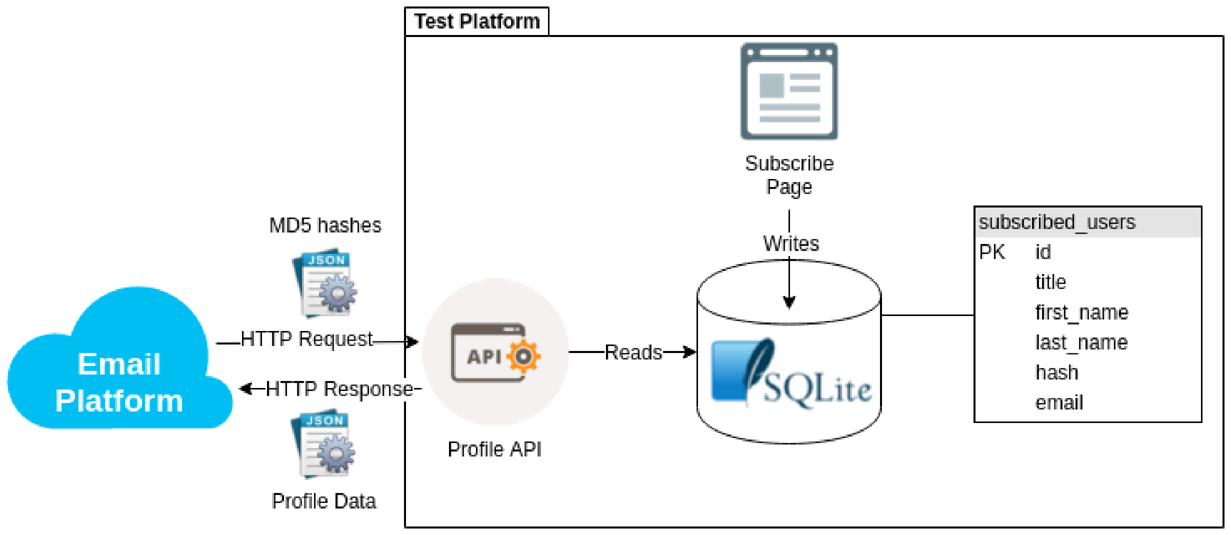


Figure 10: Profile API's architecture

In order to implement the API, the test platform should store the information of every subscription it receives, as well as the encrypted addresses so it can associate to the unencrypted ones. This storage was implemented using the SQLite database, which allowed the application to remain a self-contained Java project that could be easily deployed from its packaged *.jar* file. SQLite is a library that implements a self-contained relational database so it can be used to configure an application-specific database with no other dependencies. A single relational table is used to store the users' data (MD5-encrypted e-mail address, e-mail address, title, first name and last name). The API was implemented using a servlet that uses the request data to consult the SQLite database and sends a response containing the users' e-mail addresses to the E-mail platform.

3.2.9 Product Catalog

Initially, in order to display the aforementioned product, category and search pages a fake catalog of products was created with product image s taken from the Internet, so it would be possible send the events to the tracker. However, in order to fill the e-mail templates configured in the Backoffice, a real catalog would have to be associated.

Criteo has a common solution for product catalogs for both Display and E-mail, where the advertisers can import their list of products containing all the information the company needs (categories, product ids, names, descriptions, images, etc). When filling a template with products' information, the E-mail platform is able to consult this information using the product ids that were sent during a session and the catalog that is associated to an advertiser. Therefore, not only an existing catalog had to be associated to test platform's advertiser entity in the Backoffice, but also a solution had to be implemented to send events with pertinent ids for this catalog.

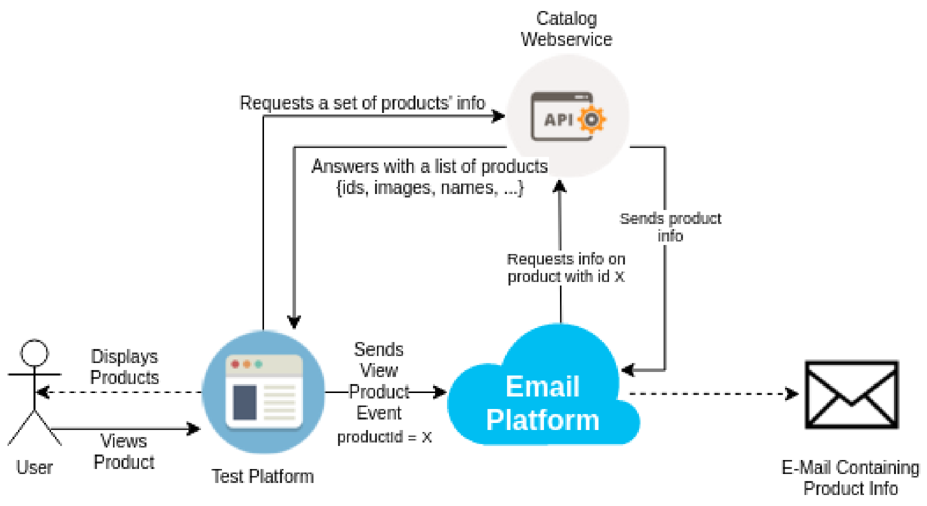

Catalog Webservice
API
Requests a set of products' info
Answers with a list of products {ids, images, names, ...}
Sends product info
Requests info on product with id X
Displays Products
Views Product
User
Test Platform
Sends View Product Event
productId = X
Email Platform
E-Mail Containing Product Info

Figure 11: Communication scheme for retrieving Test Platform's product catalog

The solution was to contact the same API that the E-mail platform uses to retrieve products' information in order to retrieve the information needed to display products in the test platform's website. This way, by retrieving products from the same catalog that was associated to the test platform's advertiser entity, the website would be able to always send pertinent product ids to the tracker. This communication scheme is presented on Figure 11.

3.2.10 Catch-All E-mail Service Provider

As previously explained, the main difference between the production and pre-production environments is that on the pre-production, the emails that are triggered by the platform aren't actually sent. This happens because in pre-production the Mailrouter is configured in a way that instead of contacting the real E-mail Service Provider (ESP), it contacts a fake ESP. This fake ESP is an application that receives the Mailrouter requests and sends back an acknowledgement of the send without actually sending an e-mail. It was implemented in Scala using the same technology as the Mailrouter - Finagle.

However, the whole objective of the test platform is for the user to be able to see the e-mails that it triggered in the pre-production environment. Therefore, a new type of fake ESP had to be implemented, the Catch-All E-mail Service Provider. The Catch-All ESP works very similarly to the original fake ESP. It acknowledges the send of every request it receives, but it actually sends some of them to a special mailbox (called the catch-all box) which can then be accessed by the testers so they can verify the content of the sent e-mail. It decides which of the e-mails should be sent by filtering the e-mail's sender address, which is configured on the e-mail templates in the Backoffice. This way, it is possible to configure the Catch-All ESP to send only e-mails triggered by the advertiser corresponding to the test platform.

Since the Catch-All ESP was just an extension of functionality of the original fake ESP, it was implemented using the same technology and the same Scala classes, as presented on

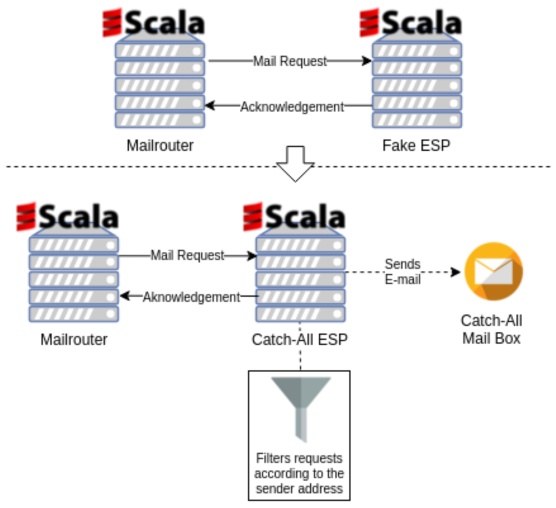


Figure 12: Extension made to the fake ESP

Figure 12. Scala is a programming language that runs on the Java Virtual Machine and for that reason has a high integrability with Java. So, for implementing the send of the e-mail the standard Java e-mail API was used. Scala also presents built-in functionality for dealing with regular expressions, so the way the application filters the e-mails is by matching its sender with a configurable regular expression, this way it is possible to configure it to send e-mails coming from more than only one advertiser.

3.3 Final Considerations

This project included more than just the development of the web platform. It involved a series of extra developments and configurations that were required from the intern for the platform to work as intended. It is often the case of applications in the real-world such as this. Hardly ever it will be possible to create a project from scratch in an isolated environment. However, if the good-practices of software-engineering and systems design are followed, it gets clearly easier to propose solutions for integration problems. Since this was the case for both the E-mail Platform and the Test Platform that was developed and integrated to it, the project was implemented and worked as intended was considered a success by the host company. More details about the evaluation of the solution are presented in the next chapter.

4 Evaluation

Since the project was implemented inside a company with a clear objective and added value, it had to be tested on a real environment for it to be considered a success. This chapter will present a few details about some statistics of the project and how it was tested, it will include a small description of a real use case that occurred during the internship.

4.1 Project Statistics

All the developments that were necessary for the functioning of the test platform (i.e. all the ones presented in the previous session) were implemented by the intern. They consisted in five months of research of the existent systems and development of new ones. The test platform consisted of roughly 1300 lines of code in Java classes, 500 lines of HTML code in JSP pages and about 50 lines of CSS code. The ESP extension consisted in about 100 lines of Scala code, since the biggest part of the functionality was already implemented.

4.2 Experiments

The developments were made in an incremental way, feature by feature, which is very characteristic of the agile methodologies in which these developments were made. The project was already functional for integration tests since its first integration with the E-mail platform; for example, a tester could subscribe in the website and verify that the tracking cookies were dropped in his browser which demonstrates the well functioning of the tracker and verify that its subscription were registered in the databases which demonstrates the well functioning of the opt-in topology. However, these kinds of tests are not very useful since there are already several integration tests that verify the functioning of the platform automatically. The real value of the test platform relies on the fact that, once all the aforementioned features were implemented, it was possible to test the whole E-mail platform without having to access its internal components. The tester is able to execute actions on a website and verify the generated e-mail in a mailbox. This kind of tests are called end-to-end tests and being able to execute them instantly, like a real tracked user, may, for example, facilitate the work of product stakeholders when validating features.

By the end of the project it was possible to use it to perform these end-to-end tests on the platform. this kind of test was used to validate not only the Test Platform itself during a demonstration with the stakeholders of the project. The platform was also used to test and demonstrate another feature that was being developed by the team in parallel, more details about this are presented in the next section.

4.3 Real Use Case

Another advantage of the test platform is that it allows a more visual way of interacting with the platform, this is specially useful during demonstrations of features. Since Criteo works using Scrum, a demonstration of what was developed is done at the end of each sprint. The test platform allows these demonstrations to be more visual and less technical, this was the case of a recently new feature implemented in the E-mail product, the recommendation.

Criteo has a recommendation engine that can choose products to be recommended to a user according to the products he has previously seen. The engine used to be only used by the Display product, but recently, one of the backlogs of the e-mail core team was to integrate it in the e-mail product, this way Criteo could attach recommended products to its e-mails as well. A screenshot of an e-mail using during the demonstration is presented on Figure 13.

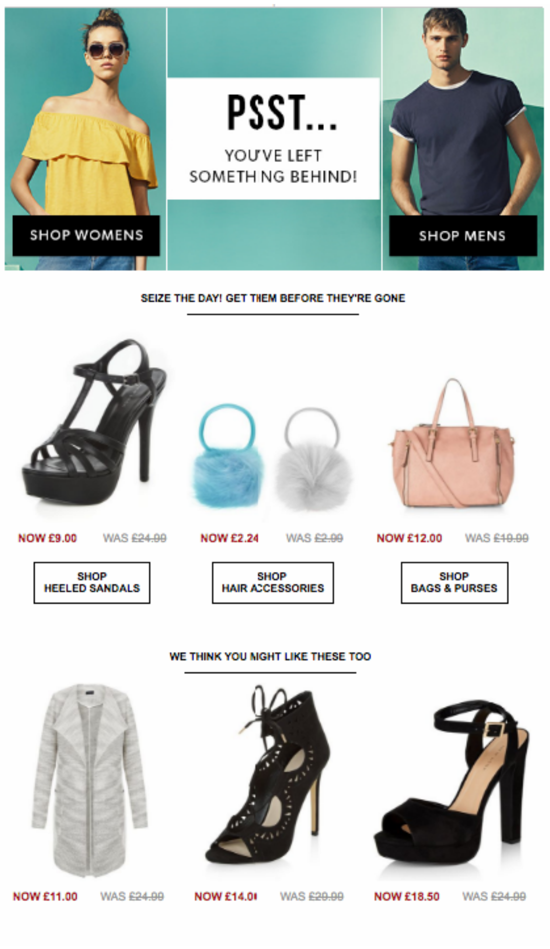


Figure 13: Example of e-mail using the recommendation feature

Once the product catalog was integrated to the test platform, the team was able to demonstrate the recommendation feature using it. This happened during an actual demonstration meeting with the company's stakeholders and the test platform behaved as intended. It was possible to visualize some products in the platform and verify that an e-mail was sent to the catch-all e-mail box containing the product recommendations related to the products that the user visualized. This kind of test is useful not only for validating features but also for demonstrating in a more comprehensible fashion the way the E-mail platform works. It was an interesting use case for the platform that required from it exactly what it was implemented to do: allow a user to subscribe, trigger a series of events and verify the e-mail it generated.

4.4 Limitations and Future Works

Although the initial objective of the platform is already fulfilled, there is still space for extensions. Currently the platform is able to test the case where everything is fine and an e-mail is actually generated, but there are several times when this is not the case. During the processing of the sessions there is a series of verifications that are made until an e-mail is sent. For example, a rule may state that no e-mail should be sent to a user if this user has already received an e-mail in the last twenty four hours, this is called the marketing pressure. A possible extension for that use case - which the product team has already shown interest in - is the possibility of verifying the exact reason for which a session didn't generate an e-mail. Currently this is done by verifying directly the databases and statistics. A development of this kind may not be developed inside the test platform project itself but certainly is within its scope and extends the possibilities of the platform.

A software stack as complex as the E-mail platform has several different components that need to be tested and more components are being constantly added as the platform evolves. Testing it may be very complicated and demanding. Tools such as the test platform ease the process and save important time for the people that need to test it. Since the E-mail platform is always evolving, the test platform will always be able to evolve as well in order to include more information about it. For that reason, in order to improve the maintainability and extensibility of the project, one of the developments that will be done in the near future is its adaptation to the Spring framework.

Spring is an open-source Java framework compatible with Java EE that is intended to facilitate the development of Java-based enterprise applications[21]. It is most known for its implementation of the concepts of Dependency Injection, which helps the independence of the Java classes, and Aspect Oriented Programming which consists in separating the business logic of the application from unrelated concerns such as security and logging[13]. An application built using the Spring framework has the characteristic of being more modular and easier to unit test.

5 The Work-flow

The implementation of the project itself was exclusive responsibility of the intern, however it was implemented inside a Scrum team. This was done as a way to integrate the intern in the usual work-flow of the company and in its collaboration ambient. Another advantage of integrating the intern in one of the teams that work in the E-mail platform was that it facilitates the contact between him and people that have an extensive knowledge of the platform and that may eventually use the test platform as well. Besides that, the project had to be integrated to Criteo's software factory, which means integrating it to all the tools responsible for the continuous integration flow at the company.

5.1 Scrum

Scrum[27] is a well known methodology for software development that emerged from the Agile movement. The so called Agile methodologies began to emerge after several failures were spotted in the classical waterfall methodology. The Agile[16] movement started in 2001 when 17 pioneers of similar methodologies met to craft the Agile manifesto, the text that stated the values and principles that should guide software development in Agile methodologies. It was strongly inspired by the lean manufacturing system that started with the Toyota production system.

Just like many other Agile methodologies, Scrum values transparency in the business and continuous improvement, not only of the development teams, but also of the methodology itself. One of its most remarkable characteristics is its capability in providing a continuous integration and continuous delivery work-flow, which opposes to the classic waterfall method that generally used for one single delivery of a big project.

Scrum defines a simple set of roles, responsibilities and meetings to guide the work of a software development team. Another characteristic of Scrum is the small size of the development teams in order to ease the process organization. The work in these teams is divided in cycles called Sprints.

5.1.1 Sprints

Usually composed of a few weeks of work, Sprints are the cycles of work time that guide the continuous integration process in Scrum. Teams using Scrum are supposed to have a deliverable product at the end of each Sprint, implementing one or a few features during the Sprint.

During the development of these features, a set of tasks previously defined have to be executed by the team. These tasks are usually small and compose the steps in order to develop a feature. These tasks are organized in a Kanban chart, which divide them in three categories: to do, in progress and done.

Everyday, a fast meeting with the whole team occurs when each member states what was done in the previous day and what will be done in the current day. These meetings are called the stand-up meetings, due to the fact that the members are usually standing during it. In these meetings, the teams also update the status of each task in the Kanban chart and recalculate the time left for all the tasks to be completed which are then plotted in another chart called the Burn-down charts.

By the end of each sprint, a series of meetings is executed in order to evaluate the work done in the Sprint and prepare the next one:

- **Demos:** the first of these meetings is the demo meeting. During the demo, the teams are supposed to demonstrate the features that were implemented in the previous Sprint to all interested parts, specially the stakeholders of the product that is being developed.

- **Retrospectives:** these meetings are the main drivers for the continuous improvement characteristic of Scrum. There are several different techniques for handling retrospective meetings, but the main objective is for the team to make a critical analysis of the last Sprint. The team is supposed to have a clear idea of what worked well and what didn't, and propose actions to improve in the next Sprint.

- **Plannings:** these are the meetings where the tasks of the next Sprint are defined and sized. By estimating the amount of work capacity the team will have in the next Sprint and how much time each task will take, the team is able the choose the tasks that should be done according to the main features to be implemented (also called backlogs or histories).

5.1.2 Roles

The Scrum guidelines also define a set of roles that each member of a team take:

- Product Owner: is the main stakeholder from the view of the development team. He is responsible for defining the backlogs that should be worked upon and their priority order. He is the main reference on non-technical subjects of each feature. In other words, he defines and validates the final result of the product in development.

- Scrum Master: the Scrum Master is usually a member of the development team that acts as a facilitator of the Scrum process. He is the responsible for maintaining the team under the Scrum work-flow. His responsibilities may include scheduling the meetings, updating the charts, etc.

- Development Team: they are the main developers of the project. They are responsible for executing the tasks of each sprint and collaborating in the work-flow of continuous integration, delivery and improvement.

- DevLead: although this is not a standard role of the Scrum methodology, it is a very important role at Criteo. The DevLead is the leader of the development team and main reference on technical issues. He acts as well as a manager for the Scrum team.

5.1.3 Scrum at Criteo

It is very common in companies that are adept of Agile Methodologies to modify a certain methodology so it can fit better in the company's culture. The adaptation of the methodologies themselves according to each specific need is one of the main values stated in the Agile Manifesto [16].

A part from the DevLead role already mentioned, there are other specificities to Criteo's development work-flow. The most notable of them is the Objectives and Key Results (OKR) planning. The OKRs compose a method for setting goals created inside Intel and became well known for its use at Google [17]. The methodology describes a way of setting mid-term goals (the objectives) and analysing the outcome as a way of validating the accomplishment of the goals (the key results).

Criteo uses OKRs as a compliment for Scrum, and defines every three months new objectives for the products in development, as well as analyzing if the key results previously planned were attained. This facilitates the process of creating backlogs on the Scrum methodology. Although OKRs weren't used for the project developed during this internship, since it is a smaller project than usual, it was a very interesting concept to learn from while working at the company.

5.1.4 Scrum in the Internship

As previously stated, this project was entirely developed under the Scrum methodology and inside a Scrum team. However, the project was developed entirely by the intern. Therefore, as a matter of organization, neither the work capacity of the intern nor the weight of the tasks developed by him were taken into account in the standard team tools (burn down charts, task planning).

Nevertheless, every Sprint inside the team corresponded as a Sprint for project. A new feature development or deliverable task (e.g. platform integration) had to be done at every Sprint. Moreover, during the demo meetings with all the E-mail teams and product stakeholders, a demo of the test platform - when applicable - also happened. This way, the whole team was aware of the current state of the development of the project.

I personally believe that this integration was crucial for the success of the project. It not only helped the organization of the development but also fomented the communication between the intern and the team members (e.g. during stand-up meetings). This allowed, for example, a smooth interaction when technical information of the E-mail platform was needed and when the objectives of the project were discussed.

5.2 Continuous Integration

Since this was a project like any other inside the company, it is natural that it follows Criteo's usual work-flow for R&D development. This means that it should be integrated to the tools that the company uses for maintaining the quality and implementing continuous integration.

In order to assure not only the quality of the software crafted on its R&D department but also its maintainability and the readability of its code, Criteo uses a set of tools that guarantee that the code will be reviewed and tested before it is deployed on production while maintaining the fast cycle of development that Agile Methodologies propose. The first of them is Gerrit Code Review.

5.2.1 Git and Gerrit

On the words of its own project leader, Gerrit is a "... web based code review system, facilitating online code reviews for projects using the Git version control system"[5]. On Gerrit, a change to the code is only merged to the repository if it has passed a set of verifications which may include a review from another developer. It is an open-source project that was developed from the code of Rietveld (another code review tool) and intended for the Android project.

The company's R&D department uses it on every project and has its own set of rules of how using it. When a change on the code is pushed by a developer, it creates a new review page on the company's Gerrit instance. This triggers a compilation job (managed by Jenkins, which will be further explained) that verifies that the new code can be merged without conflicts and that it compiles as it should. If the change passes this first phase, it gets attributed the *+1* value, the developer that created it can then indicate it is ready to submit which will trigger the actual merge of the change on the main repository. However, it is the common culture of the company that a change be reviewed by another developer before being merged. Reviewers can comment on specific lines of code and attribute another value to the change, the values are interpreted on the following way:

- **+1**: *the code looks fine to me but I can not give an opinion on whether it should be merged or not.* This type of review is not encouraged by the company and usually means that the developer has not enough expertise to approve the change.

- **-1**: *the code has some problems that should be fixed before the merge.* This type of review is usually accompanied of comments on specific sections of the code that should be changed.

- **-2**: *the code has a bug or will implicate in a major blocking problem.* This value can only be changed again by the same person that attributed it. It is unusual and blocks the merge until it gets changed.

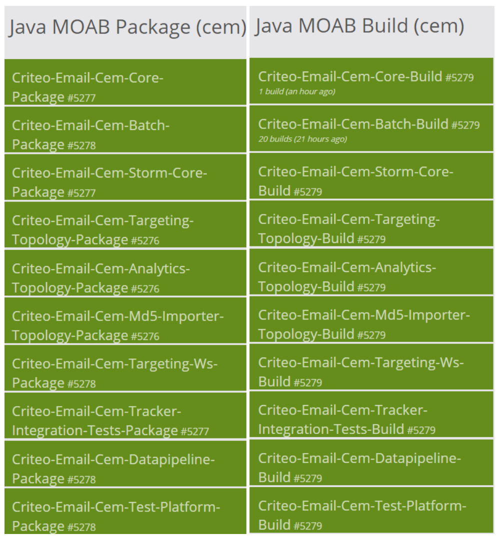
- **+2**: *the code looks perfectly fine, feel free to merge.* After this type of review, the developer that created the change has the approval of another developer and can submit the code for merging.

After the code is reviewed and merged, another tool gets in action and starts the next steps of verification of the code: the Java Mother-of-All-Builds (JMOAB).

5.2.2 The Java Mother-of-All-Builds

The JMOAB is a implementation of Criteo's MOAB paradigm for Java code. The MOAB paradigm dictates that at any given point in time, all software artifacts shall have been compiled altogether with the same level of source code (the last one at that moment). The way this works is by grouping software projects - in this case, the group is composed by every project of the e-mail platform - and for a given point at time, attribute an id to this group that represents the current state of the committed code. This id, also called the MOAB number, represents the state of the code at that moment which can be retrieved and checked out. If at that MOAB number, all projects compiled, packaged properly and passed the automatic tests, it represents a point in time where everything worked as it should and can be a candidate for a release.

The JMOAB is implemented using the open source continuous integration tool called Jenkins. On Jenkins, a wide range of jobs can be configured to be executed on servers, these jobs can be triggered in several different ways (e.g. when a source code is committed or in a periodical basis). The JMOAB is actually a series of Jenkins jobs that compile, package and test the current state of the repositories altogether, creating a new MOAB number, every time it happens. Two different web-pages were implemented to verify the state of every project at the current MOAB number, one for the building and another one for the packaging. If a project passes all the tests on one of these two phases, its name appears on green on the correspondent monitor, otherwise it appears on red.



Java MOAB Package (cem)	Java MOAB Build (cem)
Criteo-Email-Cem-Core-Package #5277	Criteo-Email-Cem-Core-Build #5279 *1 build (an hour ago)*
Criteo-Email-Cem-Batch-Package #5278	Criteo-Email-Cem-Batch-Build #5279 *20 builds (21 hours ago)*
Criteo-Email-Cem-Storm-Core-Package #5277	Criteo-Email-Cem-Storm-Core-Build #5279
Criteo-Email-Cem-Targeting-Topology-Package #5276	Criteo-Email-Cem-Targeting-Topology-Build #5279
Criteo-Email-Cem-Analytics-Topology-Package #5276	Criteo-Email-Cem-Analytics-Topology-Build #5279
Criteo-Email-Cem-Md5-Importer-Topology-Package #5276	Criteo-Email-Cem-Md5-Importer-Topology-Build #5279
Criteo-Email-Cem-Targeting-Ws-Package #5278	Criteo-Email-Cem-Targeting-Ws-Build #5279
Criteo-Email-Cem-Tracker-Integration-Tests-Package #5277	Criteo-Email-Cem-Tracker-Integration-Tests-Build #5279
Criteo-Email-Cem-Datapipeline-Package #5278	Criteo-Email-Cem-Datapipeline-Build #5279
Criteo-Email-Cem-Test-Platform-Package #5278	Criteo-Email-Cem-Test-Platform-Build #5279

Figure 14: Web page that monitors the MOAB status

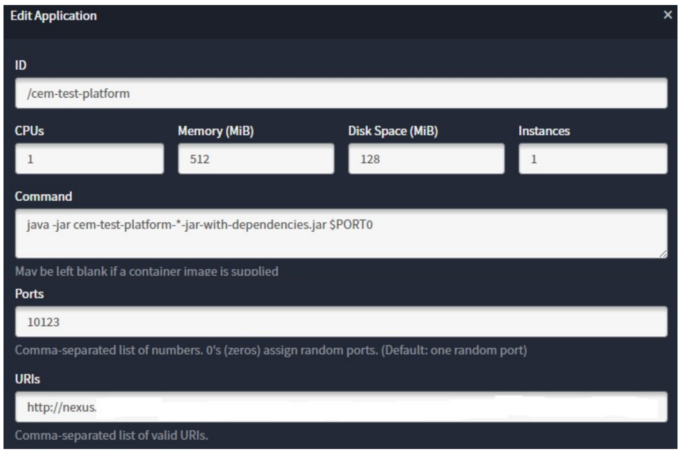
As a project belonging to the e-mail product, the test platform had to be integrated to the JMOAB, and is represented in the Figure 14 as "Criteo-email-cem-test-platform-build" and "Criteo-email-cem-test-platform-package". After a project is compiled and packaged by the JMOAB, the *.jar* file resulting from it is then transferred to a repository managed by the Nexus Repository Manager, created by Sonatype. Once on Nexus, the file can be retrieved and deployed on production and pre-production environments. The deployment of the packages that have arrived until Nexus on the pre-production and production environments is done through the cluster manager Apache Mesos and its framework Marathon.

5.2.3 Mesos and Marathon

Mesos[6] is the tool Criteo uses for managing the clusters that run some of its software. According to its main website it "...abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively"[6]. It is an open-source cluster manager software maintained by the Apache Foundation.

Marathon[11] is a framework for Apache Mesos that works as an orchestrator for the applications running on a cluster. Using its REST API or its web interface it is possible to scale the resources allocated for a specific application and manage fault tolerance on these applications. A screenshot of Marathon's web interface for running applications is presented on Figure 15.

In this project, Marathon is used to launch the main Java application that contains the website and its own embedded web-server. It is possible to configure the link for downloading the pertinent *.jar* file from Nexus and the command that should be executed in a Linux machine to launch the website, as well as the resources needed for it to run.


Edit Application

ID: /cem-test-platform

CPUs: 1

Memory (MiB): 512

Disk Space (MiB): 128

Instances: 1

Command:
```
java -jar cem-test-platform-*-jar-with-dependencies.jar $PORT0
```
Mav be left blank if a container image is supplied

Ports: 10123

Comma-separated list of numbers. 0's (zeros) assign random ports. (Default: one random port)

URIs: http://nexus.[redacted]

Comma-separated list of valid URIs.

Figure 15: Marathon web interface for running applications

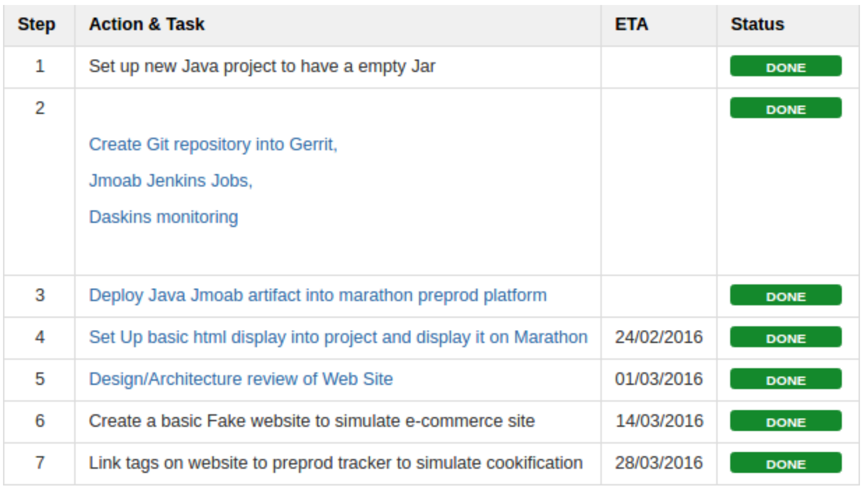
Once the website is running, it is possible to access it via web browser with a link that Marathon configures. If any problem occurs while running the application (e.g. the instance ran out of memory), Marathon is able to restart the application using the parameters that were configured. This way the website is always available to anyone who needs to use it for testing the platform. It is also possible to configure jobs in Jenkins in order to update the launched application every time a new version is commited.

5.3 The Schedule

Initially, the detailed scope of the capabilities of the testing platform was not strictly defined. The general objective of the project was to create a website where someone could simulate the actions of an Internet user browsing an e-commerce website similar to Criteo's partners' websites and verify the resulting e-mail generated by the E-mail platform. Some of the implemented features were defined during the advancement of the internship by meeting with the product stakeholders and gathering information to better address their needs. However, an initial planning for the development of the application was already crafted since the beginning. This planning can be seen on Figure 16, extracted from the project's page on the company's standard documentation repository, Confluence.

The initial weeks of the internship followed as planned above. However, as the platform evolved, new features were planned and implemented each Sprint. Besides that, other activities were also part of this internship experience, such as the trainings that the company provides to its new employees. The overview of what was done in the internship from the beginning date to the final presentation[4] is shown in the Gantt diagram presented on Figure 17, which presents a view in a Sprint-by-Sprint basis.

[4]The "Future Works" session is not detailed due to the fact that it was not entirely defined at the moment this report was written.



Step	Action & Task	ETA	Status
1	Set up new Java project to have a empty Jar		DONE
2	Create Git repository into Gerrit, Jmoab Jenkins Jobs, Daskins monitoring		DONE
3	Deploy Java Jmoab artifact into marathon preprod platform		DONE
4	Set Up basic html display into project and display it on Marathon	24/02/2016	DONE
5	Design/Architecture review of Web Site	01/03/2016	DONE
6	Create a basic Fake website to simulate e-commerce site	14/03/2016	DONE
7	Link tags on website to preprod tracker to simulate cookification	28/03/2016	DONE

Figure 16: Testing platform's initial planning taken from its Confluence page

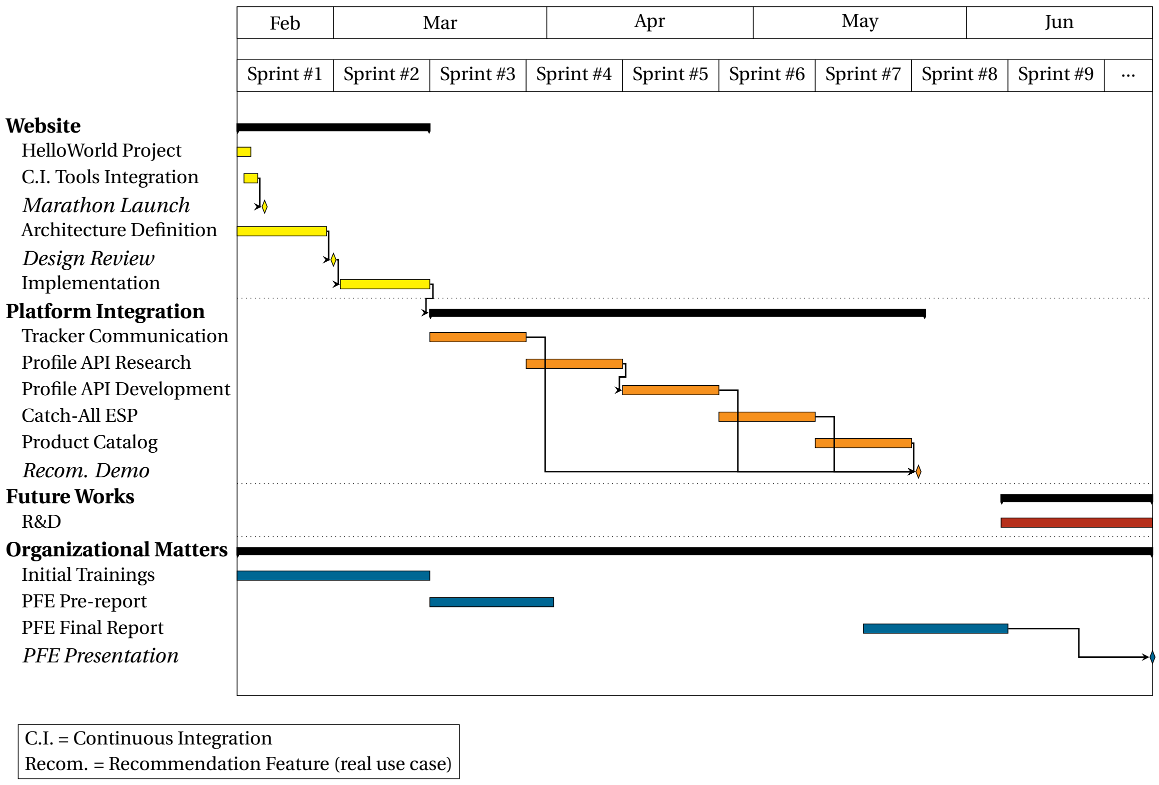


Figure 17: Gantt chart presenting the schedule accomplished during the internship

6 Conclusions

This report presented a description of a project that was implemented at the heard of a high tech company with real stakes. It was developed during an internship that works as a final studies' project. The final studies' project (*PFE*) at ENSIMAG provides the student who is finishing his course with a way of *kick-starting* his professional career by putting him in an environment with real challenges and expectations.

The proposed project was the implementation of a web application that simulates an e-commerce website. The website should be integrated to an existing system which is part of Criteo's technology called the E-mail platform in order to test it. It posed challenges in several different levels of complexity and contexts. It involved several different technologies not only concerning web development but also distributed technologies used to treat a high volume of data, which is an essential part of Criteo's core business.

Besides its complexity, the project was considered a success by the host company. Every feature that was implemented was properly demonstrated to the stakeholders of the project and validated by them. The final state of the application together with all the extra developments that were made to integrate it to the E-mail platform generated good results and was even validated with a real use case.

Future works include the development of new interfaces capable of connecting to different parts of the E-mail platform and provide different information about the processing that occurred with each session of actions performed on the Test Platform. Since the project was implemented following clear design patterns and used good software-engineering practices, it can be easily extended to include more features or change the current ones if eventually the E-mail platform changes in a way that requires it to do so. It might be even more easily extensible once it has been adapted to the Spring framework.

6.1 Personal Take

Although, I've already participated in other professional experiences, I feel that working at Criteo provided me with a complimentary view of the professional world that will aid me when deciding which paths to take in my professional development.

I had already participated in two very different internships before working at Criteo. My previous internships were very different from this one. One of them was undertook in a huge multinational company, which has over 50.000 employees and the other one was undertook in a small company that had no more than 20 employees. I consider Criteo to be in between these two definitions; it is a fast growing company with more than 2.000 employees and operates in the fast market of on-line advertising. Therefore, the experience I've had there allowed me to have a completely different view from the previous ones in terms of business processes and organization.

Criteo is a tech company that deals with very up to date technology that is constantly advancing in order to deal with more and more complex problems. To be able to work with and learn from such technologies was a great opportunity to me. I believe the knowledge I acquired during this internship will be of great use in my next endeavors. That being said, it is important to note that my experience at Criteo not only taught me specific technical skills but also changed my perspective towards the idea of knowledge in informatics. In a domain that evolves as fast as this does, it is imperative to stay up to date and continue learning even after graduation.

It is very rewarding to know that a project such as the one that was developed can be useful to the company and have a real impact in the process of testing and validating the technologies that are being developed, as it was exemplified in this report. Because of that, I am grateful to ENSIMAG and Criteo for giving me this opportunity. I'm specially grateful to Julien Ros and Benjamin Barnel, who specified this internship's project scope and helped me during the whole process of development. I am also grateful to all employees from the E-mail team who were always available for sharing their knowledge when I needed help.

References

[1] Douglas Quenqua. French retargeting company descends on silicon valley. URL https://www.clickz.com/clickz/news/1708656/french-retargeting-company-descends-silicon-valley.

[2] Liam Boogar. Criteo acquires tedemis for €21 million in its first acquisition post-ipo. URL http://www.rudebaguette.com/2014/02/20/criteos-first-post-ipo-acquisition-tedemis-forecasts-email-advertising/.

[3] Zak Stambor. Digital marketing vendor criteo's revenue jumps 54 URL https://www.internetretailer.com/2015/11/04/digital-marketing-vendor-criteos-revenue-jumps-54-q3.

[4] Erick Schonfeld. Criteo raises $10 million from index ventures. URL http://techcrunch.com/2008/01/15/criteo-raises-10-million-from-index-ventures/.

[5] Andy Singleton. Interview with gerrit project leader shawn pearce. URL http://blog.assembla.com/assemblablog/tabid/12618/bid/40855/Interview-with-Gerrit-project-leader-Shawn-Pearce.aspx.

[6] Apache Software Foundation. Apache mesos, . URL http://mesos.apache.org/.

[7] Apache Software Foundation. Apache kafka, . URL http://kafka.apache.org/.

[8] Apache Software Foundation. Apache storm, . URL http://storm.apache.org/.

[9] Apache Software Foundation. Apache cassandra, . URL http://cassandra.apache.org/.

[10] Redis Labs. Redis. URL http://redis.io/.

[11] Mesosphere Inc. About marathon. URL https://mesosphere.github.io/marathon/.

[12] Tutorials Point Staff. Apache storm - core concepts, . URL http://www.tutorialspoint.com/apache_storm/apache_storm_core_concepts.htm.

[13] Tutorials Point Staff. Spring framework - overview, . URL http://www.tutorialspoint.com/spring/spring_overview.htm.

[14] Kafka introduction. URL http://kafka.apache.org/documentation.html#introduction.

[15] Jakob Jenkov. Java web app directory layout. URL http://tutorials.jenkov.com/java-web-apps/directory-layout.html.

[16] Manifesto for agile software development. URL http://www.agilemanifesto.org/.

[17] Felipe Castro. Agile goal setting with okr - objectives and key results. URL `https://www.infoq.com/articles/agile-goals-okr`.

[18] About nginx. URL `http://nginx.org/en/`.

[19] About bootstrap. URL `http://getbootstrap.com/about/`.

[20] About couchbase. URL `http://www.couchbase.com/about`.

[21] About spring. URL `https://projects.spring.io/spring-framework/`.

[22] About criteo. URL `http://www.criteo.com/`.

[23] About jetty. URL `http://www.eclipse.org/jetty/`.

[24] About scala. URL `https://www.scala-lang.org/`.

[25] About php. URL `http://php.net/`.

[26] About symfony. URL `https://symfony.com/`.

[27] What is scrum. URL `https://www.scrum.org/resources/what-is-scrum`.