Dedication

To Mom, Dad, family and friends...

Acknowledgment

I would like to express my deepest appreciation to all those who helped to accomplish this project

A special gratitude I give to my academic supervisor, Mr. Samir BEN AHMED, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report.

Furthermore I would also like to acknowledge with much appreciation the crucial role of my supervisor Mr. Wassel MSEHLI, who contributed largely in this project by close supervision, important suggestions, encouragement and patience.

A special thanks goes to Oussama ELKACEH, DevOps manager, for his guidance, supporting this project and proposing ideas to enhance it.

Last but not least, many thanks go to all members of DevOps team Amine, Chaker, Faten and Zied whose have invested their full effort in guiding and helping me achieve the goal.

Contents

List of Figures

List of Tables

General Introduction

Two IT initiatives are currently top of mind for organizations across the globe: Big data analytics and cloud computing. In fact, as organizations have grown, the data associated with them also grow exponentially. In this context, organization that resort to providing Software as a Service solutions are compelled to monitor the efficiency of their applications, mostly by providing a clear performance measurement and guarding audit trail of the performance so changes that affect the behaviour of the application are spotted instantly and the development team can take over and fix it.

However, application become more and more complex and have multiple functionalities that finding a universal benchmark is almost impossible. In fact due to the complexity small changes can lead to dramatical changes in the benchmarks methods. Also traditional benchmark methods that use a limited set of the data cannot be used in application where Tera bytes can be processed.

Predictix company, pioneer in software-as-a-service solutions dedicated to retailers, provides agile solutions designed for the Cloud and supporting Big Data. Today, Predictix aims to improve its productivity and better its distributed applications management by giving its employees the possibility to monitor the performance of each one of their projects, in an easy way.

Our internship goal is to create a multi-tenant application that will allow every employee in Predictix to have the possibility to, in the first place, check the performance history of each one of their projects, and, in the second place, run new benchmarks on one of the applications through simple click of a button, through a simple and easy-to-use user interface.

The remainder of this report contains four chapters. In the first chapter, entitled Overview, we will present an overview of the project that details the project's contexts. We will, also, define the general concepts that the project is related to. Finally, we will,

briefly, state the problematic that our project is considering and the way that our project will treat this issue.

The second chapter, entitled State of the Art, will be devoted to explain general concepts that the project is related to. We will go though each one of the concept so we can give the reader a general understanding of them.

The third chapter, entitled Requirements Definition, will be devoted to the project requirements. Indeed, it details the functional and non-functional requirements, the Product backlog and the requirements specifications.

Finally, the last chapter, entitled Project Realization, describes the implementation, testing, and deployment processes of the achieved work.

Chapter 1

Overview

Introduction

This introductory chapter presents the general context of the project and gives an overview of its scope. In the first section, we will specify pedagogical and professional frame of the project, where we will introduce the host company, its business field and the DevOps team. Then, in the second section, we will present some general concepts of the project. Finally, we will define the project problematic and present the project goals.

1.1 General context

In the part we will put the project in its frames, both scholar and professional. First we will set the pedagogical. Then we will define the professional frame, i.e Predictix and more specifically the DevOps team.

1.1.1 Pedagogical Frame

This project was elaborated in the context of a graduation internship, a final step of obtaining the National Diploma of computer engineering from the Faculty of Science of Tunis. The internship was undertook on period of four months, from February 1st to May 31th, 2017, during which, we integrated the DevOps team at Predictix Tunisia.

1.1.2 Professional frame

Predictix is an American company headquartered in Atlanta, GA, USA, with a center of operations in Tunisia.

1.1.2.1 Predictix: Foundation and activity domain

Predictix is a fast growing consulting and software company founded in 2005, headquartered in Atlanta, Georgia USA with affiliations in Tunis, Tunisia, London, UK and Amsterdam, Netherlands. It merged with LogicBlox in January 2014 which private company that provide the next generation of smart database. This new database permits to present finished products to the clients in the simplest ways, after performing a lot of complicated cloud and multicore computing.

Predictix is specialized in providing software solutions implementing predictive analytic technologies to solve retail problems and improves retailers' profitability. In fact, it helps Tier 1 retailers and brands make better merchandising decisions by eliminating the traditional silos between varrious buying and selling decision, delivering solutions that evolve easily to adapt to their ever-changing business and simplifying the process and lowering the stakes of investing in technology.

As recognition, Gartner, the world's leading information technology research and advisory company, has classified as a leader, in the Magic Quadrant for Merchandise Assortment Management Applications in July 2015.

As a matter fact, because of the leading role of Predictix in Cloud based solutions for retailers. It get acquired by Infor, in 2016.

1.1.2.2 Infor

Infor is the third-largest provider of enterprise applications and services behind Oracle and SAP. Think: over 70,000 customers in 194 countries for its industry-specific applications and suites designed for the cloud, on-premises, or both, that brings in about $2.5 billion in revenue. To handle this massive operation, Infor employs 13,000 people across the globe.

Predictix joined Infor as the base of Infor-retail the Infor division for retailers.

Infor on Predictix: "Atlanta-based Predictix experienced more than 40 percent growth in SaaS subscriptions in 2015 and counts 5 of the top 15 global retailers as customers, managing more than $60bn in weekly forecasts. The company has an engineering- and science-driven culture with deep expertise in retail and a drive to revolutionize the industry. LogicBlox, the company's technology platform underlying all Predictix applications, revolutionizes the development of next-generation predictive and prescriptive applications, and has attracted funding from DARPA, the Defense Advanced Research Projects Agency."

Predictix is providing SaaS applications to big retailers, among which the largest drug retail chain in the United States in sales and profits with annual sales of over US $76 billion in 2015, employs over 240,000 people, with more than 8,000 stores across all 50 U.S States, the District of Columbia, Puerto Rico and the US Virgin Islands. As a matter of act, for forecasting purposes, this client benefited from $125M from inventory reduction alone, with 20% of ongoing. Some of Predictix clients, whose solutions are already on production, are

- Kiabi
- The Home Depot
- Crate and Barrel
- M.Video
- Target

Predictix offers its clients a variety of solutions. The figure 1.1 gives an overview of Predictix business units:

Figure 1.1: Merchandising Predictix Suite

Pricing and Promotions: Retailers, wholesalers and some manufacturers use Predictix services to help them manage their promotions, pricing and markdowns for higher profit bottom lines,by offering a very high flexible and configurable solution that facilitates the user interaction with the results he gets.

- **Forecasting and Replenishment**: The firm provides forecasts and replenishment strategies that enable a retailer to implement the promotional, pricing and assortment strategies established. Additional profit can be gained by minimizing inventory levels and reducing lost sales.
- **Assortment and Space Optimization**: Predictix helps retailers to define the optimal localized assortment for every store and every space. Retailers can expect category margins to improve 100 basis points or more, and to drive higher sales, from optimizing their assortment with Predictix ASO.
- **Planning and Allocation**: Business Consulting at Predictix helps retailers review their current planning and allocation process and design new ones for the future.

In order to provide these solutions , Predictix has forged a number of strategic partnerships with other companies. Some of which are the following companies:

- **LogicBlox**: The platform provider for Predictix' solutions. Predictix use LogicBlox' scalable and declarative database system, programming language and different environment tools to design and build its systems. After a partnership lasting for years, the two companies have merged together.

- **Amazon Web Services**: AWS is a world leader in cloud computing solutions. Predictix is using AWS' IaaS and PaaS solutions to host its different systems. The need for cloud services will be detailed later in the next chapter.

During its life cycle, every client's product goes through a number of steps, from development to testing to deployment. Most of tools and steps of the development pipeline are the responsibility of the DevOps team within which we undertook this internship.

1.1.2.3 DevOps team

The DevOps team is needed in every step of the client's product life cycle. In fact, they gather and collect the data, store it, do batch processing or real-time processing on it, and serve it via an API to a data scientist who can easily query it. They have extensive knowledge on databases and best engineering practices. These include handling and logging errors, monitoring the system, building human-fault-tolerant pipelines, understanding what is necessary to scale up, addressing continuous integration, knowledge of database administration, maintaining data cleaning and ensuring a deterministic pipeline.

The main activities of Predictix's architecture and integration team can be summarized to:

• They intervene, in the beginning of the project when the requirements are defined. They play the role of consultant as they are consulted to discuss the data specification, in collaboration with the data science team, with the client. In fact, their role, at this step is to take all necessary measures in order to make sure that the data is normalized and consistent.

• They intervene, also, in the development of the applications, as they are in charge of the applications architecture design. In fact, they translate complex functional and technical requirements into detailed architecture, design and high performing software with the ability to architect highly scalable distributed systems.

• They are, also, in charge of batch implementation thorough data modelling, rules definitions, protocol buffers services, etc.

• Finally they are responsible of the application being automatically build and deployed on the cloud, with extremely high attention to the application performance, as it world affect the application cost.

• One of the main duties of the DevOps team. Is to create internal tools that facilitate

the development of project, deploying and monitoring performance and cost.

lb-job-cost: It's a tool for monitoring lb jobs cost of each client's account. It will give the current actual cost for the month along with the projected cost for the whole month.

lb-workflow-console: It's a web application that helps engineers at Predictix to monitor the batch process in real-time.

Command-history: A web applications that monitor the commands used in all Predictix EC2 machines. It give the user the ability to search the history of command usage.

Nixops-dashboard: SaaS solution that allows users with no technical knowledge, to easily provision Predictix applications on the cloud. Business consultants, testers, and others are now able to deploy complex infrastructures with a click of button.

lamias-servicetester: A DSL for testing web services. It can be used for unit tests and for regression tests.

1.2 Project presentation

Since Predictix works with 1 tier retailers in the US, application data tend to be large enough to face issue in performance. In fact some of Predictix client deliver weekly data of the size of 2 TB of sales records. Developer need to monitor the performance of the application during development phase.

Measuring the performance of an application is an operation that has to be done on daily basis or every time changes are introduced in the application codebase or one its dependencies. To automate the benchmarking process, we need first to understand how developer are doing benchmarks currently.

Compare to traditional 3-tired development stack, Business logic is installed in the LB database itself, Thus simplifying application development by alleviating the need for:

- Fetching data to and from DB
- Integration between OLAP and OLTP
- Simpler language for business logic

So all LB applications expose a set of services to the client (mostly the UI). Each service is a feature in the application or a part of a feature. To test the performance we

measure the time it took for a request to one of the services to complete, Thus measuring both the performance of the application and the LogicBlox database.

The services request are predefined as Lamias tests. The tests can be Relax query or simple HTTP requests. The definition of the Lamias tests used in the benchmarks differ per application. Thus benchmarking is only relevant in the scope of one application. Thus, benchmarks of different application can't be used to compare the performance of the two. Also the tests used to measure the performance of one application can't be reused in the benchmark of another one.

Conclusion

The first chapter provided an overview of the general context of the project and its scope. We introduced the host company and the technical DevOps along with project problematic.

Chapter 2

State of the Art

Introduction

We will introduce in this chapter some useful definitions and concepts that she light on the project. We will also assess the current situation be examining the literature as well as the current solutions adopted by Predictix. Next, we will specify the different proposed solutions and choose the one that fits most our needs.

2.1 Datalog

In computer science, declarative programming is a programming paradigma style of building the structure and elements of computer programsthat expresses the logic of a computation without describing its control flow.

Many languages that apply this style attempt to minimize or eliminate side effects by describing what the program must accomplish in terms of the problem domain, rather than describe how to accomplish it as a sequence of the programming language primitives (the how being left up to the language's implementation). This is in contrast with imperative programming, which implements algorithms in explicit steps.

Declarative programming often considers programs as theories of a formal logic, and computations as deductions in that logic space. Declarative programming may greatly simplify writing parallel programs.

Datalog is a declarative logic programming language that syntactically is a subset of Prolog. It is often used as a query language for deductive databases. In recent years, Datalog has found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.

Its origins date back to the beginning of logic programming, but it became prominent as a separate area around 1977 when Herv Gallaire and Jack Minker organized a workshop on logic and databases. David Maier is credited with coining the term Datalog.

The applications that we will measure the performance of, are all written in commercial implementation of Datalog, LogiQL.

2.2 REST architecture

Rest stands for Representational State Transfer. It is an architecture style for designing networked applications. It permits creating, modifying resources easily. Indeed REST is a lightweight alternative to complex mechanism like RPC, CORBA and SOAP.

Rest is not a 'standard'. In fact, it is a guideline to build an efficient framework for communication between two machines using HTTP protocol. The World Wide Web itself, based on HTTP, can be viewed as a REST-based architecture. REST relies on a stateless, client-server, cacheable communication protocol. It is simple to implement and maintain. In addition, it allows applications to be scalable by supporting multiple backend services

at the same time.

Much like Web Services, a REST service is platform-independent, language-independent standard-based as it runs on top of HTTP, and is easily used in the presence of firewalls.

However, there are a few major concepts which make REST unique from other web services. In fact , its main key principals are the following:

• **Unique URL-Resource mapping**: Every resouce is mapped to a unique URL. That refers to a some logical way to access information.

• **Statelessness**: All information required to process the request by server is contained along with the request. This means that no information of the previous request is maintained by the server. This is inherited from the fact that REST is based on HTTP.

• **Action Verbs**: REST architecture use HTTP verbs to identify the appropriate action. The main HTTP verbs used in a REST architecture are GET, POST, PUT and DELETE. In fact, GET is used by the client to access the resource on the server, PUT to update a resource, POST to create a new one and DELETE to remove resource.

• **Data Exchange formats**: REST architecture does not require any particular encoding for the resource body. JSON [6] and XML are the most used format, but it can be PROTOBUF, YAML etc.

2.3 Cloud

Cloud computing is a type of Internet-based computing that provides shared computer processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (e.g., computer networks, servers, storage, applications and services), which can be rapidly provisioned and released with minimal management effort. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in either privately owned, or third-party data centers that may be located far from the userranging in distance from across a city to across the world. Cloud computing relies on sharing of resources to achieve coherence and economy of scale, similar to a utility (like the electricity grid) over an electricity network.

2.4 SaaS

Software as a service (SaaS) is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. It is sometimes referred to as "on-demand software", and was formerly referred to as "software plus services" by Microsoft. SaaS is typically accessed by users using a thin client via a web browser. SaaS has become a common delivery model for many business applications, including office and messaging software, payroll processing software, DBMS software, management software, CAD software, development software, gamification, virtualization,[4] accounting, collaboration, customer relationship management (CRM), Management Information Systems (MIS), enterprise resource planning (ERP), invoicing, human resource management (HRM), talent acquisition, content management (CM), and service desk management. SaaS has been incorporated into the strategy of nearly all leading enterprise software companies.

2.5 Automation

Automation or automatic control, is the use of various control systems for operating equipment such as machinery, processes in factories, boilers and heat treating ovens, switching on telephone networks, steering and stabilization of ships, aircraft and other applications and vehicles with minimal or reduced human intervention. Some processes have been completely automated.

Automation has been achieved by various means including mechanical, hydraulic, pneumatic, electrical, electronic devices and computers, usually in combination. Complicated systems, such as modern factories, airplanes and ships typically use all these combined techniques. The biggest benefit of automation is that it saves labor; however, it is also used to save energy and materials and to improve quality, accuracy and precision.

The term automation in the software industry has gain a lot of success in the recent years. From the automation of build to the deployment, various steps in the development pipeline can be automated. Including benchmarking the applications.

2.6 Benchmark

In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally

by running a number of standard tests and trials against it.[1] The term 'benchmark' is also mostly utilized for the purposes of elaborately designed benchmarking programs themselves.

Benchmarking is usually associated with assessing performance characteristics of computer hardware, for example, the floating point operation performance of a CPU, but there are circumstances when the technique is also applicable to software. Software benchmarks are, for example, run against compilers or database management systems.

Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures.

In our case the we are going to benchmark applications along with the database system that powers them. Since they are strongly coupled.

2.7 SOA

Service-Oriented Architecture(SOA) is primarily regarded as a technical architecture consisting of tools and service specification to build loosely coupled applications. At another level it is also a means to leverage flexibility and agility to system services as it offers a hierarchical framework to coordinate simultaneous business process design and implementations using loosely coupled service infrastructures. SOA has been debated both in the academy and industry and misinterpretations of its nature impede its adoption. Web service is the most popular technical architecture applied in SOA, among which the most popular and commercially successful platform is SOAP messaging protocol. In SOAP Web service invocations must be carried that are compliant to SOAP messaging standard. Services are described in a service description standard called WSDL. Since SOAP allows multiple message exchange patterns such as request-and-response, broadcasting and sophisticated message correlations, it can be used to integrate almost all kinds of legacy system from different vendors.[5]

Basic technologies such as (XML, SOAP, WSDL) provide means to describe, locate, and invoke services in SOA as an entity in its own right. However, these technologies do not give a rich behavioral detail about the role of the service in more complex collaboration. This collaboration includes a sequence of activities and relationships between activities, which build the business process. There are two ways to build this process: service orchestration and service choreography.

2.7.1 Service orchestration

Service orchestration represents a single centralized executable business process (the orchestrator) that coordinates the interaction among different services. The orchestrator is responsible for invoking and combining the services.

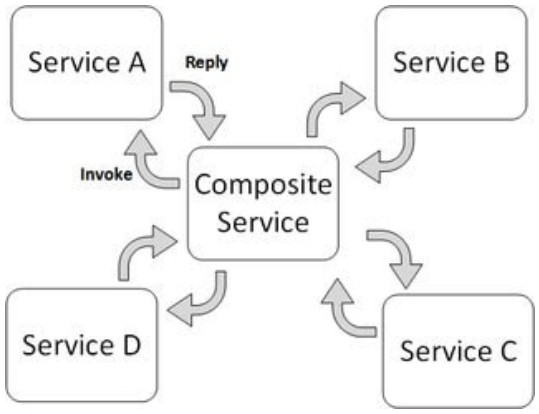

Figure 2.1: Service orchestration

The relationship between all the participating services are described by a single endpoint (i.e., the composite service). The orchestration includes the management of transactions between individual services. Orchestration employs a centralized approach for service composition.

2.7.2 Service Choreography

Service choreography is a global description of the participating services, which is defined by exchange of messages, rules of interaction and agreements between two or more endpoints. Choreography employs a decentralized approach for service composition.

The choreography describes the interactions between multiple services, where as orchestration represents control from one party's perspective. This means that a choreography differs from an orchestration with respect to where the logic that controls the interactions between the services involved should reside.

2.8 BPMN

The Business Process Management Initiative (BPMI) has developed a standard Business Process Modeling Notation (BPMN). The BPMN 1.0 specification was released to the

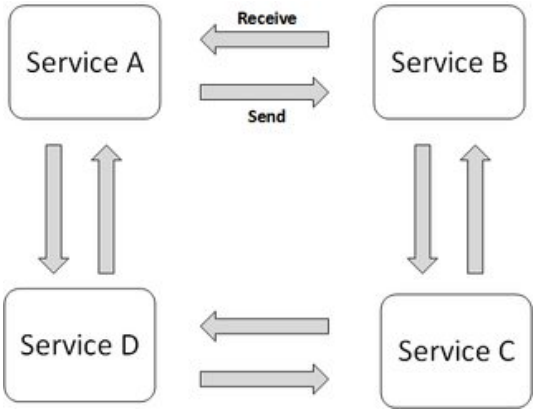

Figure 2.2: Service choreography

public in May 2004. This specification represents more than two years of effort by the BPMI Notation Working Group. The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and, finally, to the business people who will manage and monitor those processes. BPMN will also be supported with an internal model that will enable the generation of executable BPEL4WS. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation. BPMN defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance.

2.9 Agile

The Agile Method is a particular approach to project management that is utilized in software development. This method assists teams in responding to the unpredictability of constructing software. It uses incremental, iterative work sequences that are commonly known as sprints. Were a sprint is a period of time allocated for a particular phase of a project. Sprints are considered to be complete when the time period expires. There may be disagreements among the members of the team as to whether or not the development is satisfactory; however, there will be no more work on that particular phase of the project. The remaining phases of the project will continue to develop within their respective time frames.

Scrum is just one of the many iterative and incremental agile software development method. In the SCRUM methodology a sprint is the basic unit of development. Each sprint starts with a planning meeting, where the tasks for the sprint are identified and an estimated commitment for the sprint goal is made. A Sprint ends with a review or retrospective meeting where the progress is reviewed and lessons for the next sprint are identified. During each sprint, the team creates finished portions of a product.

Conclusion

In This chapter we introduced general concepts and general solution a specific problems that are the core of this project. The reader now have a global overview on the problem domain that we are facing and a good understating to some of the tools that are used it to solve it. In the next chapter we will dive deep to the project requirement and client needs.

Chapter 3

Requirement definitions

Introduction

After imbibing the general context of the project, we are going to focus in the is chapter on providing a full description of out project, that we will be calling Benchmarks Dashboard. To do so, we will start by analyzing the requirements and extracting the functional and non-functional specifications. Then we will define the Product backlog of the project to achieve. Finally we will present the general use cases of the project.

3.1 Requirements analysis

3.1.1 Design session

After the beginning of the internship, we had a on-month training where we were trained about the different technologies that Predictix is using, such as LogiQL, Nix and LB. Also, as interns, we had the opportunity, first, to get familiar with the company's working environment, second, understand better the project's goal and its added value to its users, and third, were able to collect the functional and non functional requirements of the project.

3.1.2 Functional requirements

The purpose of this project is to automate the benchmarking process of the applications developed in Predictix. For that there are two main requirements that need to be addressed.

3.1.2.1 Automating "benchmark runs"

The application should give the ability to the clients, the developers, to run benchmark with the click of a button. The application must run all the repeated process without user's interactions. For that 3 operations must be automated:

• **Deploy the application**: The applications developed in house are SaaS applications, thus they must be deployed to the cloud. The user choose what the application that want to run benchmark for. Then we need to automatically provision a server in the cloud, install the application on that server along with all dependencies that are needed to run the benchmark.

• **Load test data**: To have a real and concise measure of the performance of the application, a data sample should be loaded into the database. This sample should be the same across all benchmarks runs so it can make sense to compare them. The data should be loaded as soon as the application is installed.

• **Run The benchmark**: After loading test data we will run a series of service test and measure the time each service took to complete the request. Each service test will be run 3 times to unveil any issues with database warmup.

Finally after the benchmarks completed the result of the test should be stored for the developer to check and the instance should not be provisioned. The figure 3.1 shows the BPMN diagram of "New Benchmark" process.

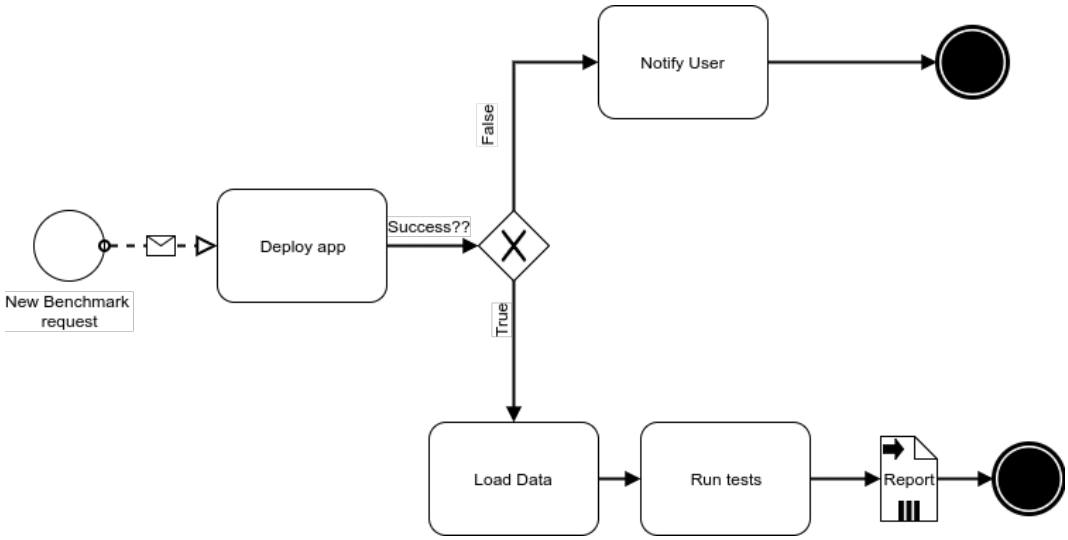

Figure 3.1: BPMN diagram: New benchmark

3.1.2.2 Visualize the previous benchmarks

The developer should be able to see and investigate the result of the benchmarks. He must be also be able to:

• **Compare benchmarks**: In this case the user can see difference in the overall performance of the application, between two different dates.

• **Test history**: The user can see the test history. Thus the state of the test between all benchmarks, or specify two bencharmks.

• **Benchmark status**: The user can see the status of the benchmark. Whether they competed successfully or not or they still running. Also they can see who started the benchmark run, and when he started it.

3.2 Product Backlog

From the previous requirements and some other related to user management we can extract the product backlog described in table 3.1

Once the Product backlog was established and validated by the Product Owner, the scrum team has broken down the Product Backlog into four sprints of a duration of 20 days each. The sprint duration was suggested by the Scrum Master as sprint ceremonies were planned every 20 days at the end of each sprint.

Table 3.1: Product Backlog

ID	User Story	Priority	Estimation
1	As a user, I want to login using my username and password and start using the application right away.	2	14
2	As a user, I want to visualize accounts applications and select one to run new benchmark	1	20
3	As a user, I want to visualize the list of accounts.	4	3
4	As a user, I want to visualize the status of accounts benchmarks.	7	8
5	As a user, I want to be able to choose an account and visualize its history	5	14
6	As a user, I want to be able to choose a benchmark and visualize the tests duration.	6	6
7	As a user, I want to be able to choose a test in an account and visualize the history over all benchmarks.	8	8
8	As an account administrator, I want to be able to grant Run benchmark right to users also be able to revoke it.	9	4
9	As an administrator, I want to be able to manage all accounts.	3	3

	User stories	Estimation
Sprint 1	As a user, I want to visualize accounts applications and select one to run new benchmark	20
Sprint 2	As a user, I want to login using my username and password and start using the application right away.	14
	As an administrator, I want to be able to manage all accounts.	3
	As a user, I want to visualize the list of accounts.	3
Sprint 3	As a user, I want to be able to choose a test in an account and visualize the history over all benchmarks.	14
	As an account administrator, I want to be able to grant Run benchmark right to users also be able to revoke it.	6
Sprint 4	As a user, I want to visualize the status of accounts benchmarks.	8
	As a user, I want to be able to choose a test in an account and visualize the history over all benchmarks.	8
	As an account administrator, I want to be able to grant Run benchmark right to users also be able to revoke it	4

Table 3.2: User Stories through each sprint

The table 3.2 defines the user stories that will be achieved during each sprint.

3.3 Requirement specifications

3.3.1 Actors identification

The intended users of the Benchmarks Dashboard are Predictix employees, The AppDev team (Application development team). They must be able to, easily Interact with the system, run new benchmarks and check the status and result as soon as possible.

- **AppDev team**: The AppDev members are the developer that implement new feature in the application, write the necessary steps to build the application and monitor the performance of the application after every new feature.

- **Account admin**: The account admin will be responsible for managing account users. Grant run benchmark rights and revoke them.
- **Administrator**: the administrator is a person who should be chosen, carefully. in fact, s/he will be creating, editing and deleting users, accounts. The administrator should also be able to assign accounts to users.

3.3.2 Use case diagram

The figure 3.2 introduces the general use cases which present the functionalities of the application.

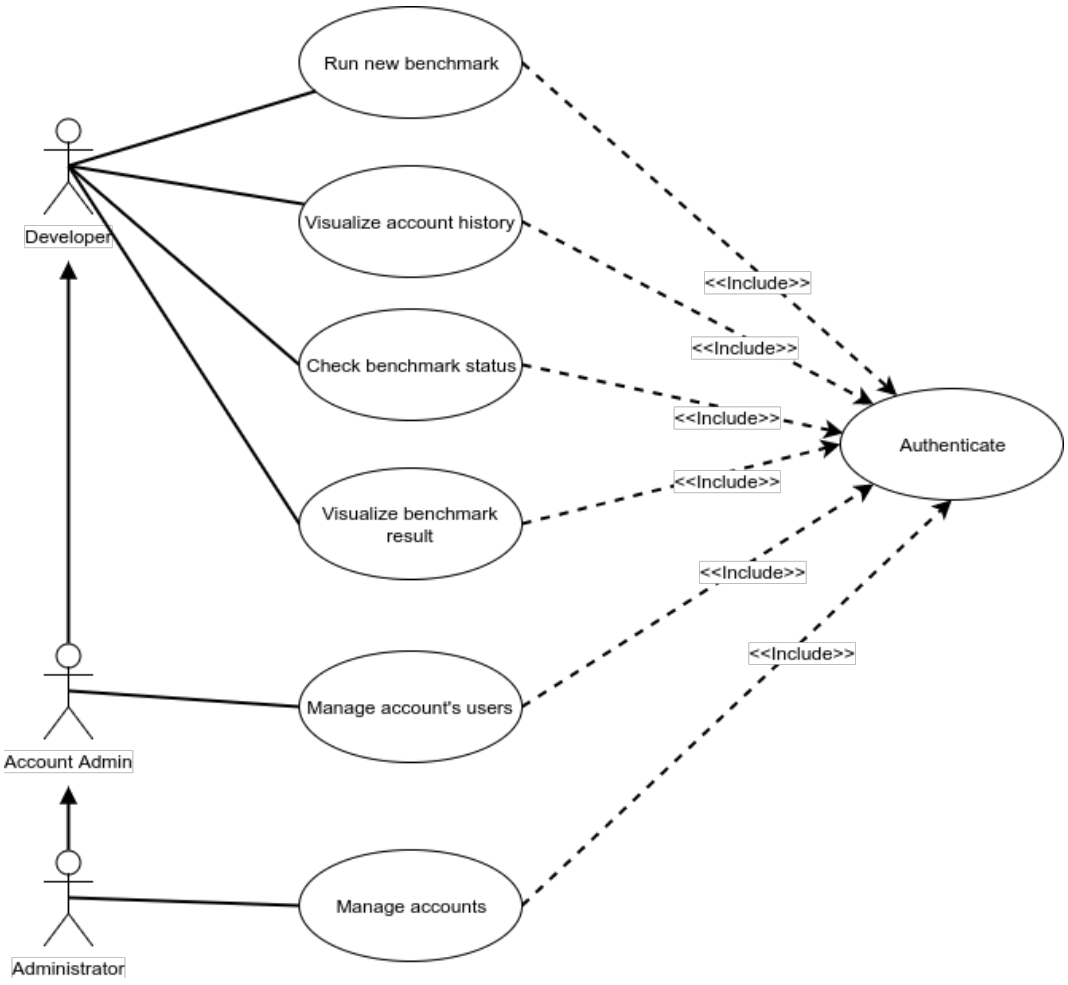

Figure 3.2: General use case diagram

3.4 Mockups

In this part we are going to describe and validate the project business requirements in every UI views using mockups. First, we are going to present the Pencil tool. Then we are going to define the project's mockups.

3.4.1 Pencil

Predictix development teams suggested to use Pencil, which is a small graphical tool to sketch out use interfaces, for websites and web / desktop / mobile application. Mockups provide enough interactivity to replace prototypes, and make it easy to collaborate and get feedback on the wireframes.

3.4.2 Project mockups

In this section, we are going to describe our project's components mockups.

UI general Description

The Benchmarks dashboard is composed of five main screens which are the account choice, Account's info, Tests choice, Test, New Benchmarks. In the following, we are going to proceed by describing the mockups and the need required in every view.

3.4.3 Account choice

The figure 3.5 represents the landing page of our application, for a administrator user, after the authentication screen which the account's choice. The page should contain a list of the different accounts that the user have access to. In fact, the user should have the possibility to select a certain account, in order to view the all benchmarks run and the list of tests of that project.

3.4.4 Account overview

3.4 represents the Account overview screen mockup. This page is generated after a user selected an account. This page will contain a chart that represent the overall performance of the account's tests. The chart will contain the total duration it took every benchmark

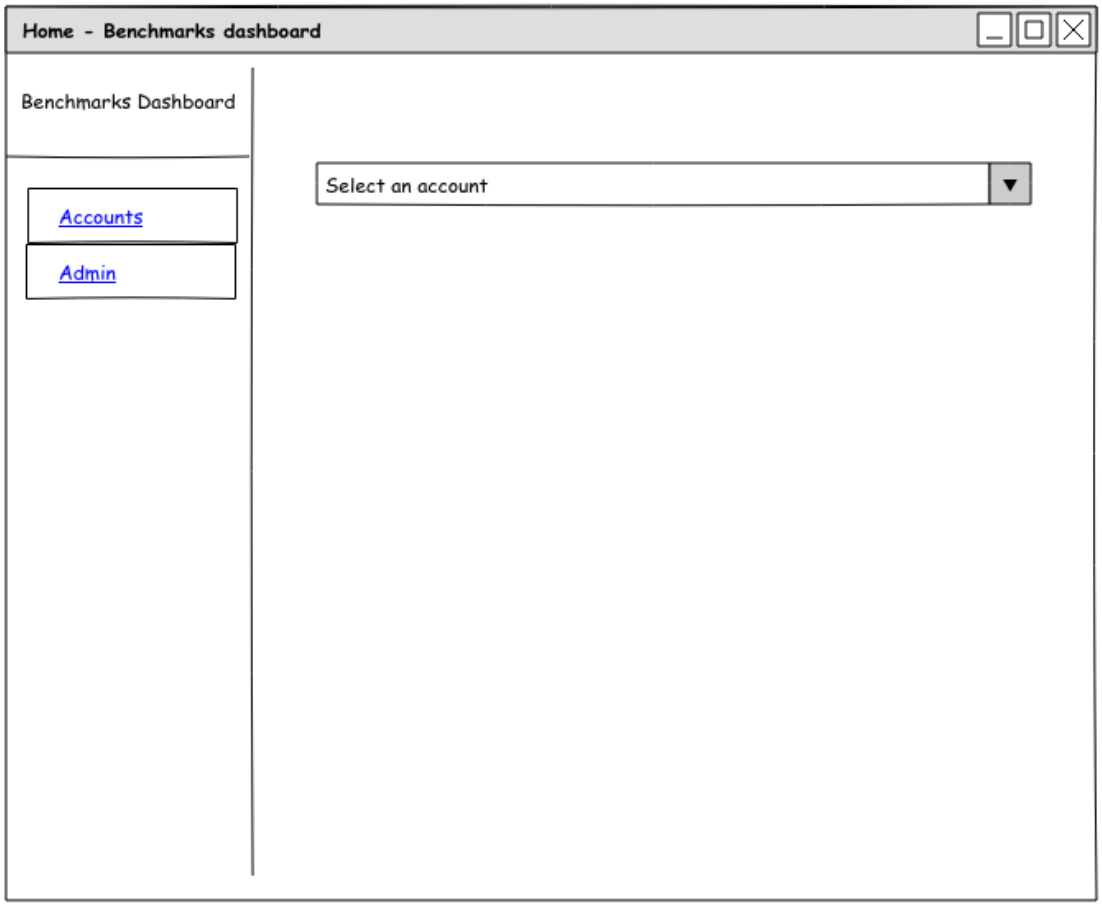

Figure 3.3: Benchmarks dashboard Home

run to finish. It will give a general overview on the state of the account's applications, the general trend for its tests and also any alerts about the the current benchmarks status.

3.4.5 New deployment

The figure 3.5 represents the landing page of our application, for a administrator user, after the authentication screen which the account's choice. The page should contain a list of the different accounts that the user have access to. In fact, the user should have the possibility to select a certain account, in order to view the all benchmarks run and the list of tests of that project.

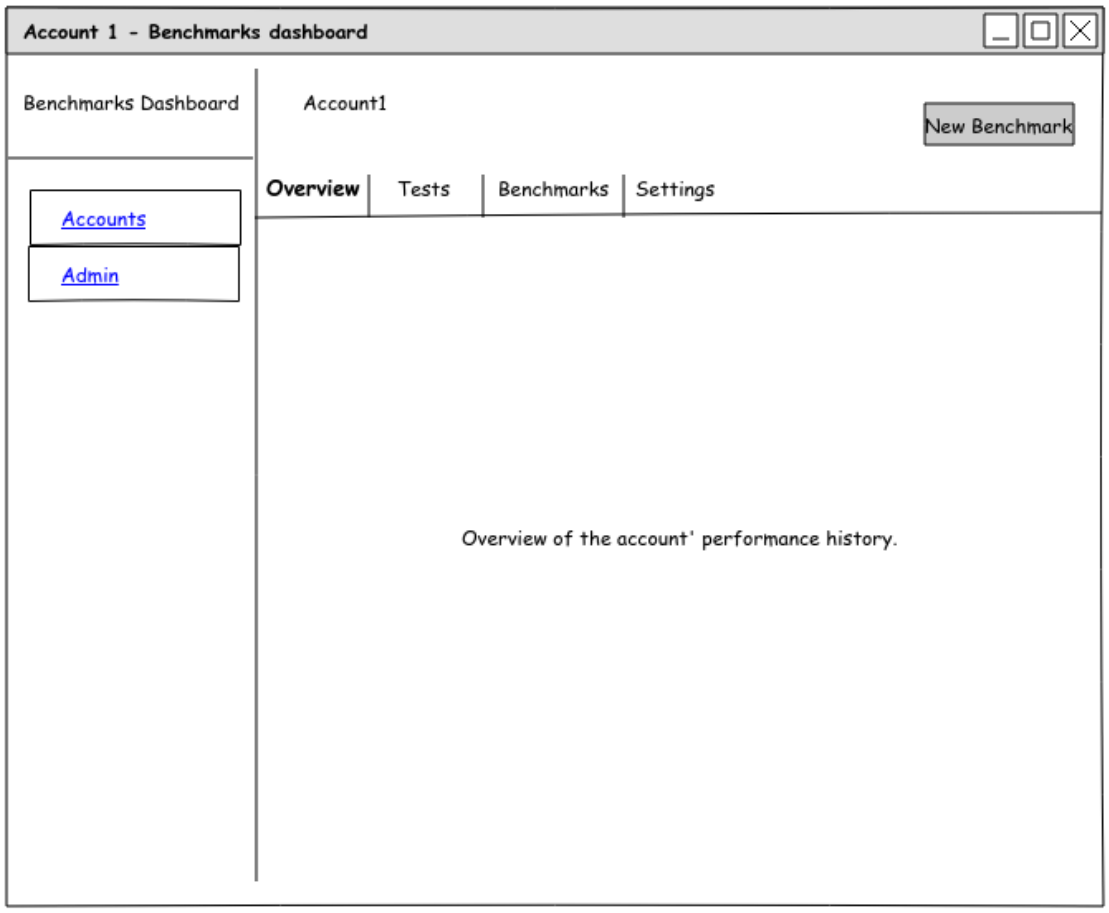

Figure 3.4: Account overview

3.4.6 Account' tests

3.6 represents the account's tests mockup. This screen will present the list of all account's tests. This list will show the test name, the percentage of time difference between the latest two runs for that test and the difference in seconds. The list tests that exceed 15 seconds in difference will be highlighted in red and those that exceed 5 seconds will be highlighted in yellow. This will help the user to identify malfunctioning tests. Th

Conclusion

In this chapter, we have established the requirements analysis and have defined the requirements specifications. This have led us to clearly identify the system's actors and the

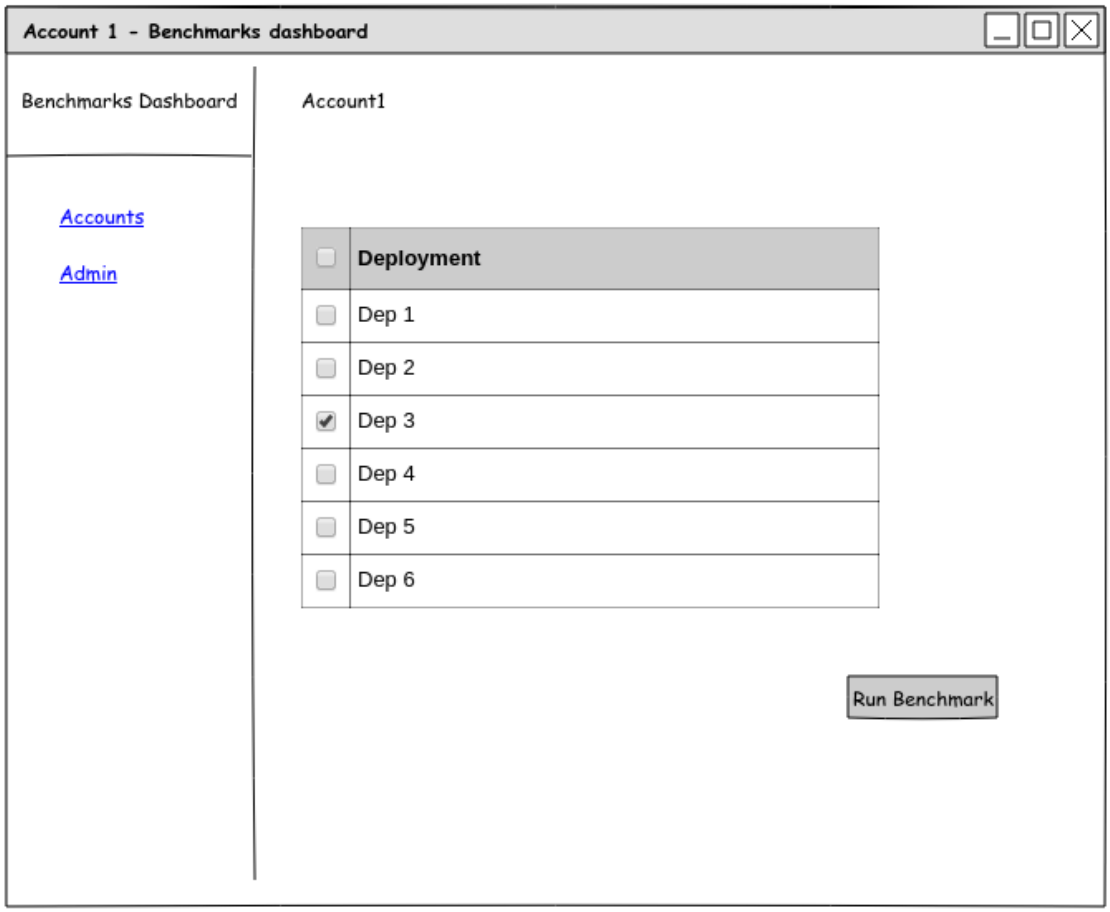

Figure 3.5: New benchmark screen

features it should implement. Presented system's mockups, and described the functionalities needed by the user

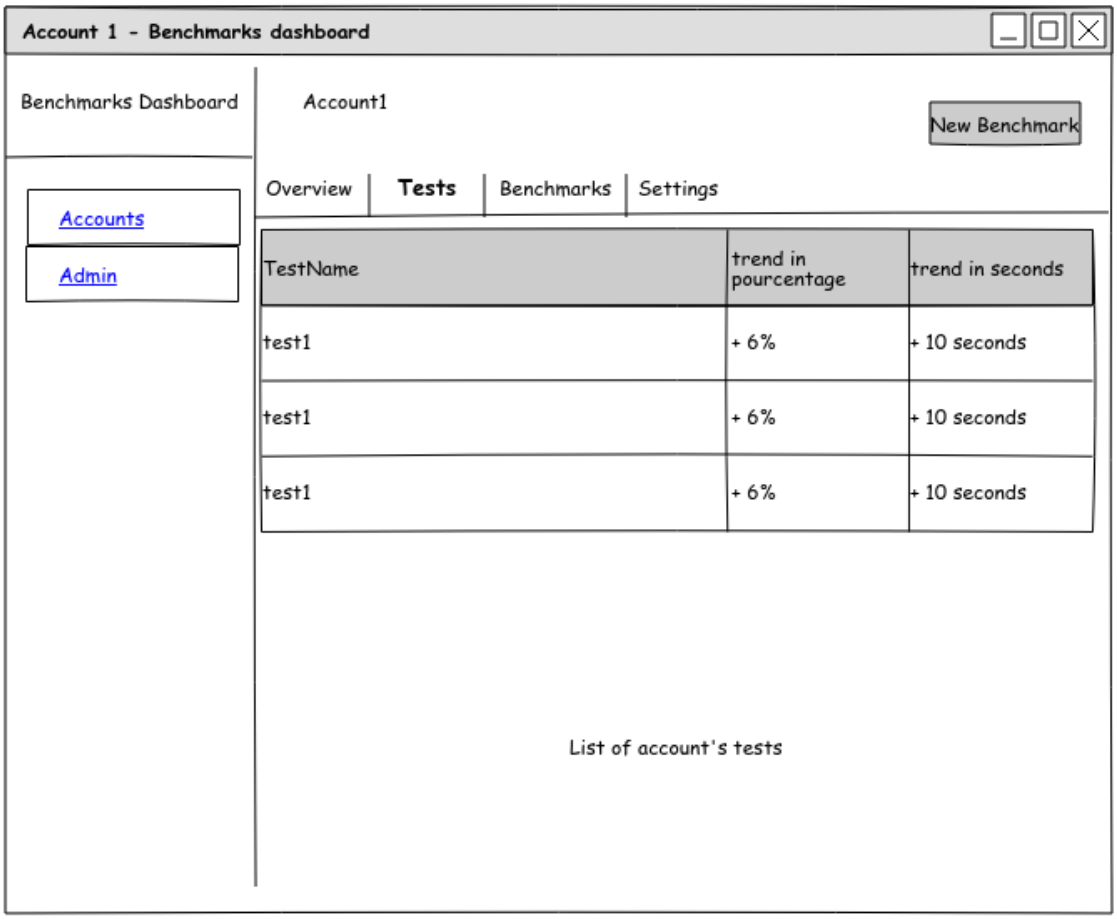

Figure 3.6: Account overview

Chapter 4

Project realization

Introduction

This chapter covers the implementation, testing and the deployment processes of the Benchmarks dashboard project. To deal wit each of these processes, we will introduce the technologies used and describe the way we used them. In addition, we will present the project planning.

4.1 Used technologies

4.1.1 Programming languages

4.1.1.1 Python

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2.

Python is a high-level general-purpose programming language that can be applied to many different classes of problems. The language comes with a large standard library that covers areas such as string processing (regular expressions, Unicode, calculating differences between files), Internet protocols (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming), software engineering (unit testing, logging, profiling, parsing Python code), and operating system interfaces (system calls, filesystem, TCP/IP sockets). Look at the table of contents for The Python Standard Library to get an idea of what's available. A wide variety of third-party extensions are also available.[2]

4.1.1.2 Javascript

JavaScript (shortened to JS) is an interpreted, object-oriented, weakly typed, programming language with first-class functions most known as scripting language for Web pages. It was originally implemented so that client-side scripts could interact with the user, control the browser, communicate asynchronously and alter the web page content that was displayed, and now it has evaluated to involve game development and applications creation. JavaScript's main purpose was the creation and embedding of scripts in the HTML client side in order to guarantee more interactivity with the user. It was created to offer dynamic tasks such as the transformation of the page elements, visual effects and immediate reactions to the user actions. But now JS for server-side web is gaining popularity with the appearance of JavaScript frameworks for that purpose. JavaScript was a purely interpreted language. This means that scripts execute without preliminary compilation (without conversion of the script text into machine code). The user's browser interprets

the script, that is, analyzes and immediately executes it. In modern implemented, The user's browser interprets the script, that is, analyzes and immediately. [3]

4.1.2 Frameworks

4.1.2.1 Pyramid

Pyramid is a fast and low-level Python web framework. It is developed as part of the Pylons Projects. Even Pyramid isn't famous like Flask and Django, it minimal, unobtrusive and agnostic design helps in developing more robust and fast web applications.

4.1.2.2 ReactJS

React is an open-source front end library developed by Facebook. It's used for handling view layer for web and mobile apps. ReactJS allows us to create large web applications that use data which can change over time, without reloading the page, using reusable UI components. Its main goal is to be fast, simple and scalable. It is currently one of the most popular JavaScript libraries and it has strong foundation and large community behind it. React components are typically written in JSX, a JavaScript extension syntax allowing quoting of HTML and using HTML tag syntax to render sub-components.

4.1.2.3 Grommet

Grommet [10] is a UX framework for enterprise applications developed by HP, on top of ReactJS. It was created to align the design and developer workflow into one seamless experience. The Grommet components [11] library is designed to be perfect complement to the design system. With little setup, developer will be able to create a Grommet application from scratch in minutes and take the pain out of translating design files into running application code.

4.1.3 Tools and services

4.1.3.1 AWS

Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms. These services operate from many global geographical regions

including 6 in North America. They include Amazon Elastic Compute Cloud, also known as "EC2", and Amazon Simple Storage Service, also known as "S3". As of 2016, AWS has more than 70 services, spanning a wide range, including compute, storage, networking, database, analytics, application services, deployment, management, mobile, developer tools and tools for the Internet of Things. Amazon markets AWS as a service to provide large computing capacity quicker and cheaper than a client company building an actual physical server farm.

4.1.3.2 Nix

Nix [1] is a purely functional package manager. This means that it treats packages like values in purely functional programming languages such as Haskell they are built by functions that dont have side-effects, and they never change after they have been built. Nix stores packages in the Nix store, usually the directory /nix/store, where each package has its own unique sub-directory such as

`nix/store/b6gvzjyb2pg0kjfwrjmg1vfhh54ad73z−firefox−33.1/`

Where `b6gvzjyb2pg0...` is a unique identifier for the package that captures all its dependencies (its a cryptographic hash of the packages build dependency graph). This enables many powerful features.

Nix was created by Eelco Dolstra as PhD project. Eelco is, along with other main contributors to the Nix ecosystem, employed as DevOps engineer at Predictix.

4.1.3.2.1 Nix language The Nix expression language is a pure, lazy, functional language. Purity means that operations in the language don't have side-effects (for instance, there is no variable assignment). Laziness means that arguments to functions are evaluated only when they are needed. Functional means that functions are normal values that can be passed around and manipulated in interesting ways. The language is not a full-featured, general purpose language. Its main job is to describe packages, compositions of packages, and the variability within packages.

4.1.3.2.2 NixOS Nixos is Linux distribution built on top of the Nix package manager. It uses the Nix language for declarative configuration which allows reliable system upgrades. Although NixOS started as a research project, it is a fully functional and usable operating system.

4.1.3.2.3 NixOps NixOps [7]is a tool for deploying sets of NixOS Linux Machines, either to real hardware or to virtual machines. It extends NixOS's declarative approach to system configuration to networks and adds provisioning.

The NixOps Dashboard is a built-in house extension to NixOps and offers the same content and functionality via a Web UI and a rich RESTful API. Core features are:

- User-friendly web-based interface
- Improved security
- Scheduled operations (deployments, backups, ...etc)
- Permanent traceability
- Clear audit trail
- Role based access

We will be using NixOps Dashboard API to provision servers and deploy the applications in the cloud in order to benchmark them in a production-like systems.

4.2 Development environment

Below are the characteristics of the development machine we used during the project implementation.

- **Processor**: Intel Core i7-3540M CP
- **RAM**: 16 GO DDR3
- **System**: Ubuntu Linux 14.04 LTS

4.3 Project management

This section gives an overview of the project management process of the Benchmarks Dashboard application. At first, we will start by introducing the project team and then we will describe the tracking process of the different implementation tasks.

Scrum role	Name
Product owner	Oussama Elkaceh
Scrum Master	Wassel Msehli
Team members	Chaker Benhamed

Table 4.1: Project members and their role

4.3.1 Project team

The table **??** introduces the project team members and their roles.

4.3.2 Jira

Predictix development teams use Jira for managing the projects. Jira is a commercial software product developed by Atlassian. It is used for bug tracking, issue tracking and project management. Jira allow prioritizing, assigning, tracking, reporting and auditing issues. Indeed, it improves productivity by cutting down on time wasted on tracking issues and coordination. In fact, it keeps the team on track and allows the project manager to monitor the progress on projects. Besides, it improves quality be ensuring that all tasks are recorded down with all details and followed up until accomplishment. Moreover, Jira is an extensible platform which means that it offers workflow customization to match more the business process. [4]

4.4 Implementation

4.4.1 Achieved work

This section is a description of the four sprints and the results we achieved at the end of all of the sprints.

4.4.2 Sprint 1

In the first sprint, our goal is to start by automating the benchmark of one of Predictix's applications, thus we can understand how to generalize for all other project and solve issue of consistency. The application we worked on is THD-AP. We started by installing the application locally understand how to build it, This included how to compile the application, install it and run the tests. We also deploy it in the cloud and run the

benchmark tests in cloud server. After getting acquainted to the benchmark process. We started by automating the benchmark process by adding systemd services to Install the application, Load the data and run the benchmarks

4.4.3 Sprint 3

In the second sprint, our goal is to model the web application that the users will use to both start benchmarks and visualize the benchmarks runs. We stated by looking the available frameworks for web development. Finally we developed the workflow that will run new benchmark.

After we get acquainted to the benchmark process, we started righting defining the steps in the benchmark pipeline.

4.4.3.1 Deploying

To be tested in production-like environment, the application need to be deployed to the cloud. In order to compare between benchmarks runs the server in which the application will be deployed need to be consistent across all the runs The user mustn't have any interaction with the provisioning of the server nor the deployment of the application.

• **Provisioning**: Since Predictix application are memory-intensive, we chose to deploy them in a r3.4xlarge EC2 instances [8] from Amazon AWS. This type of instances offer lower price of GiB of RAM. r3.4xlarge has 122 GB of RAM with 16 cores CPU and 320 GB of SSD Storage.

• **Installing**:After provisioning the server. We need to deploy the application to that server. The closure of the application will be already built in Hydra. We will be using an existent nix configuration to install the application. It uses a systemd service called *install-app*. This service will start the database, the web-server, install workspaces and it will make sure that the application is up and running.

For starting the deployment we use NixOps Dashboard, it will take some arguments like the type of the instance, AWS account to use to deploy the application, The application configuration (Users, Whether to run workflows or not...) and application closure (package). And start deploying the application with those configurations. After NixOps finish creating the instance in EC2 cloud. It will install the packages using predefined instructions to install LB applications.

4.4.3.2 Load data

Loading data is project specific, each application need to provide the steps to download data from S3, extract them and run the necessary workflows to load the data into the database. Those steps will be written in a script called *load-perf-data.sh* by the application developers, the script will be ran as soon as the application in installed. It will be expected to be at the script directory within the application root directory. The figure 4.1 shows the definition of the systemd service *load-perf-data.sh*. And the figure 4.2 Shows the log of systemd service and how the service is invoked automatically.

```
systemd.services.load-perf-data = {
  description = "Load Perf data";
  after = [ "install-app.service" ];
  script = "source /etc/profile ; sh /data/lb_deployment/installed-app/scripts/load-perf-data.sh";
  serviceConfig = {
    Type = "oneshot";
    RemainAfterExit = true;
    User = "logicblox";
  };
};
```

Figure 4.1: Load perf data nix expression

```
load-perf-data-start[5513]: [load-perf-data] Import workspace - BEGIN
load-perf-data-start[5513]: imported workspace '/thd-ap'
load-perf-data-start[5513]: [load-perf-data] Import workspace - END
load-perf-data-start[5513]: [load-perf-data] run workflow - BEGIN
load-perf-data-start[5513]: 2017-06-20 09:12:13,53800+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > lb.JsonService (maste
load-perf-data-start[5513]: 2017-06-20 09:12:14,00400+00:00  INFO Driver            - Executing task 'dev.deploy_initial_data > master.deploy_initial_data > lb.JsonService (master.wf, line
load-perf-data-start[5513]: 2017-06-20 09:12:15,25400+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > lb.JsonService (maste
load-perf-data-start[5513]: 2017-06-20 09:12:16,00900+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > lb.StartServices (mas
load-perf-data-start[5513]: 2017-06-20 09:12:16,22100+00:00  INFO Driver            - Executing task 'dev.deploy_initial_data > master.deploy_initial_data > lb.StartServices (master.wf, li
load-perf-data-start[5513]: 2017-06-20 09:12:21,89800+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > lb.StartServices (mas
load-perf-data-start[5513]: 2017-06-20 09:12:22,51600+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_hierarc
load-perf-data-start[5513]: 2017-06-20 09:12:22,74900+00:00  INFO Driver            - Executing task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_hierarchies > mas
load-perf-data-start[5513]: 2017-06-20 09:12:22,75800+00:00  INFO lb.tdx.v2.Import  - 'dev.deploy_initial_data > master.deploy_initial_data > master.import_hierarchies > master.TdxImport >
load-perf-data-start[5513]: 2017-06-20 09:12:22,97400+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users >
load-perf-data-start[5513]: 2017-06-20 09:12:23,27800+00:00  INFO Driver            - Executing task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users > master.Td
load-perf-data-start[5513]: 2017-06-20 09:12:23,27900+00:00  INFO lb.tdx.v2.Import  - 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users > master.TdxImport > lb.td
load-perf-data-start[5513]: 2017-06-20 09:12:33,73400+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_hierarc
load-perf-data-start[5513]: 2017-06-20 09:12:34,18800+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users >
load-perf-data-start[5513]: 2017-06-20 09:12:34,60200+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users >
load-perf-data-start[5513]: 2017-06-20 09:12:34,79200+00:00  INFO Driver            - Executing task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users > master.Td
load-perf-data-start[5513]: 2017-06-20 09:12:34,79300+00:00  INFO lb.tdx.v2.Import  - 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users > master.TdxImport > lb.td
load-perf-data-start[5513]: 2017-06-20 09:12:45,47000+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_users >
load-perf-data-start[5513]: 2017-06-20 09:12:45,83800+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_measure
load-perf-data-start[5513]: 2017-06-20 09:12:46,08600+00:00  INFO Driver            - Executing task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_measures > master
load-perf-data-start[5513]: 2017-06-20 09:12:46,10300+00:00  INFO lb.tdx.v2.Import  - 'dev.deploy_initial_data > master.deploy_initial_data > master.import_measures > master.TdxImport > lb
load-perf-data-start[5513]: v, /tmp/dev_data/initial/measures/COGS_LY_C_AP.csv, /tmp/dev_data/initial/measures/Inv_TY_U_AP.csv, /tmp/dev_data/initial/measures/MkdStrPos_LY_R_AP.csv, /tmp/de
load-perf-data-start[5513]: 2017-06-20 09:12:57,10000+00:00  INFO Driver            - Processing request: task 'dev.deploy_initial_data > master.deploy_initial_data > master.import_measure
load-perf-data-start[5513]: 2017-06-20 09:12:57,24100+00:00  INFO Driver            - Terminating driver due to 1 queries without work to do.
load-perf-data-start[5513]: 2017-06-20 09:12:57,25400+00:00  INFO Driver            - Scheduler terminated.
load-perf-data-start[5513]: [load-perf-data] run workflow - END
```

Figure 4.2: load-perf-data log

4.4.3.3 Running benchmark tests

For running tests we use a generic script called *run-perf*. This script will define the instruction for running the tests, collect the logs, generate the report and upload it to S3. The figure 4.3 shows the definition of the systemd service *perf-runner* responsible for running the script, it will be launched as soon as the data is loaded with the *load-perf-data* service.

The figure 4.4 shows a truncated log of perf-runner service.

```
# service that will execute the performance test suite and collect the logs.
systemd.services.perf-runner = {
  description = "execute the performance test suites";
  after = [ "load-perf-data.service" ];
  requires = [ "load-perf-data.service" ];
  wantedBy = [ "multi-user.target" ];
  script = "source /etc/profile && sh ${<perf_src>}/scripts/run-perf";
  serviceConfig = {
    Type = "oneshot";
    RemainAfterExit = true;
    User = "logicblox";
  };
};
```

Figure 4.3: perf-runner nix expression

Finally after the test are completed and the report are generated they will be automatically published to our application using the HTTP protocol.

4.4.4 Sprint 3

In the second sprint, our goal is to model the web application that the users will use to both start benchmarks and visualize the benchmarks runs. We started by choosing web framework. The web ecosystem has been evolving with consistent rate over the last few years. Having in mind that the application need to be modular, maintainable and stable. Also from the fact that the company made a strategic decision to move to React JS, Thus most of the frontend developers in the company are familiar with React. We choose it to build the frontend of our application.

React helps in defining the application frontend in a declarative approach. Component in react are like class functions in a pure functional language. The whole and single purpose of a react component is to render an object using the passed state. Most of the component that we wrote are stateless in the meaning that they don't depend of the application state to render. All the data that the component need to successfully render are passed to it in the instantiation time. This type of component are called presentational component.

We also used the Redux library that help to organize the overall state of the application. The later is saved in a one big data object that called Store, Redux then work using an action-dispatch paradigm. Component aren't responsible of getting the data from the backend and changing the store content. If a data need to be fetched, let's say user click on a button, then they only need to dispatch action. When an action finish the store will be updated and redux/react will only update the component that need to be updated. The

```
systemd[1]: Starting execute the performance test suites...
Starting Run 1
Done Run 1
compress the logs
[perf runner] - checking that all inputs are available ...
[perf runner] - file: 'lb-server.log' found.
[perf runner] - file: 'lb-web-server.log' found.
[perf runner] - file: 'results.csv' found.
[perf runner] - Generating the Benchmark website ...
splitting /tmp/test_results/lb-server.log in sections ...
splitting /tmp/test_results/lb-web-server.log in sections ...
creating section tests_attribute_mappings_SHOULD_assign_an_attributeId_for_each_Product ...
creating section tests_attribute_mappings_SHOULD_assign_an_attributeLabel_for_each_Product ...
creating section tests_attribute_mappings_SHOULD_assign_an_attributeLabel_for_each_SKU ...
...................
...................
...................
creating section tests_Start_New_Plan_Start_New_Plan_Setup_and_Service_call_SNP_Second_Transaction ...
[perf runner] - Benchmark website generated.
[perf runner] - Publishing the Benchmark website ...
[perf runner] - Benchmark website generated.
[perf runner] - Generating the benchmark report ...
[perf runner] - Checking the existence of the history of benchmark runs in AWS S3 ...
[perf runner] - Checking predictix/perf-history.csv ...
[perf runner] - Updating the full history of runs in S3 ...
[perf runner] - Full history is updated.
[perf runner] - Compressing the results tarball ...
[perf runner] - Uploading the results tarball to AWS S3 ...
[perf runner] - email is sent to 'chaker.benhamed@infor.com'
DONE.
systemd[1]: Started execute the performance test suites.
```

Figure 4.4: perf-runner log

figure 4.5 present the difference between a react application that use redux and another one that don't. Figure 4.6 shows the workflow of action dispatching in a react application.

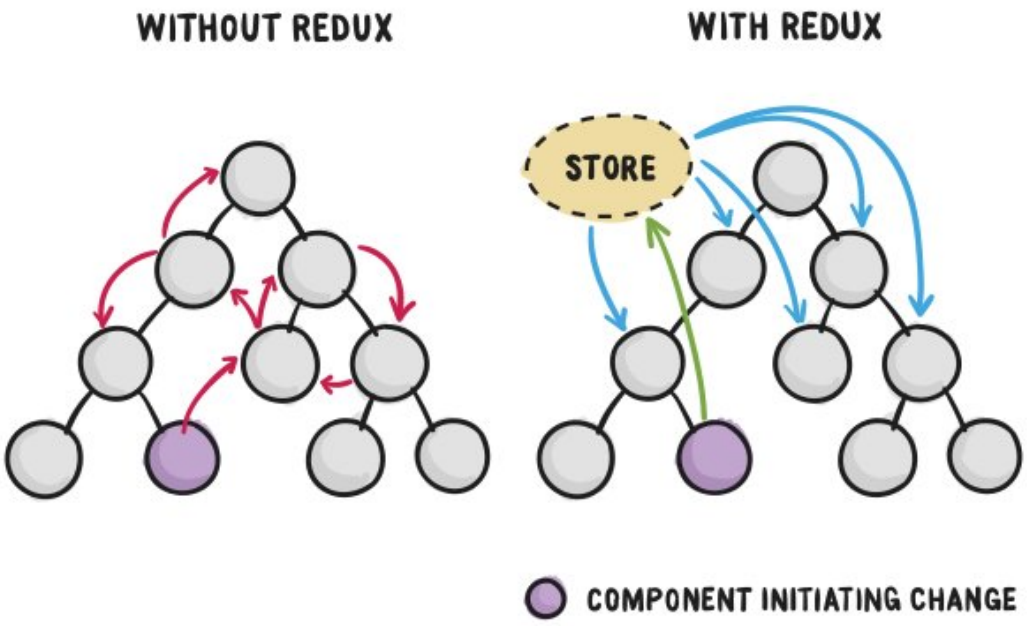

Figure 4.5: React application with and without Redux

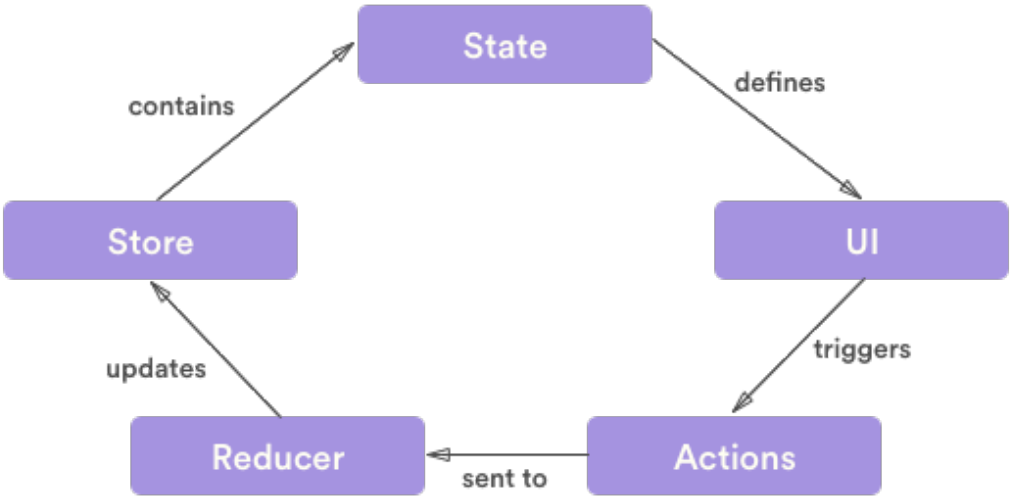

Figure 4.6: Workflow of react application

4.4.5 Sprint 4

In this sprint we mostly focused on the authentication and the authorization part of the application.

4.4.5.1 Authentication

To authenticate users we used Google+ API. Since all developers in Predictix have @predictix.com Google account they can use it to authenticate to the application check 4.4. If it's first login, a new user will be stored in the database. Then all future requests will be associated to that user using JWT token. The figure 4.4 shows the list of the account that the user should select from after he click Login with Google button.

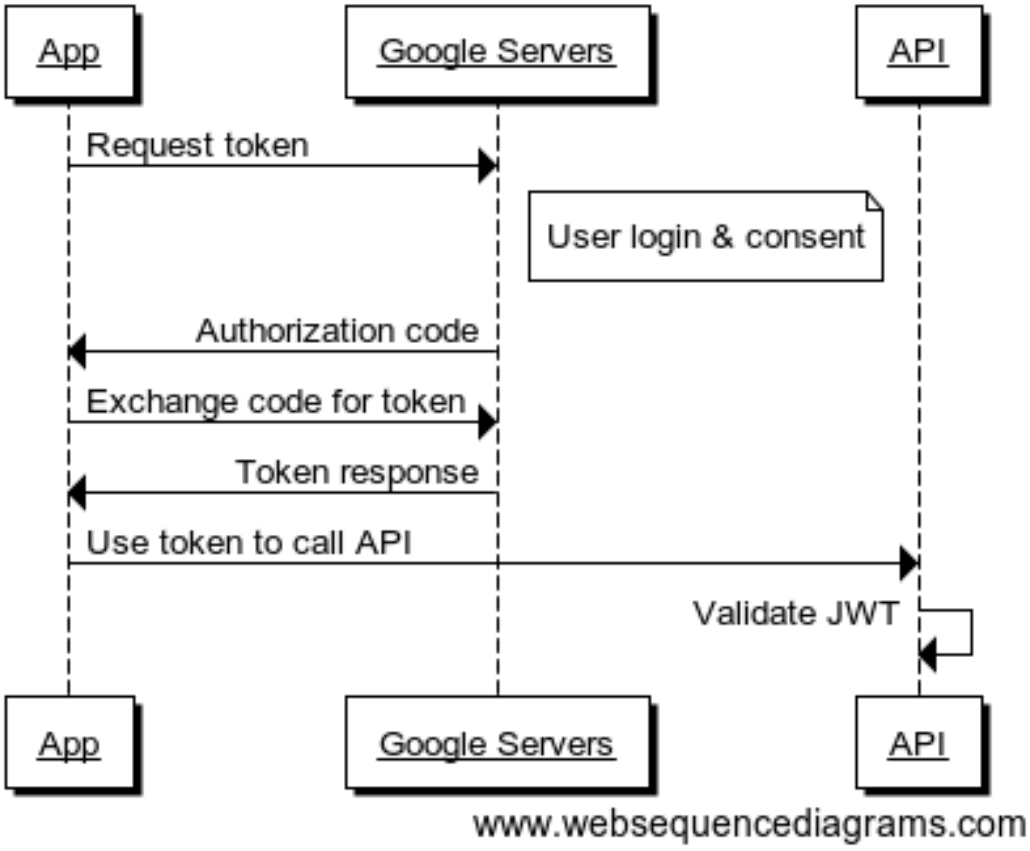

Figure 4.7: Google authentication workflow

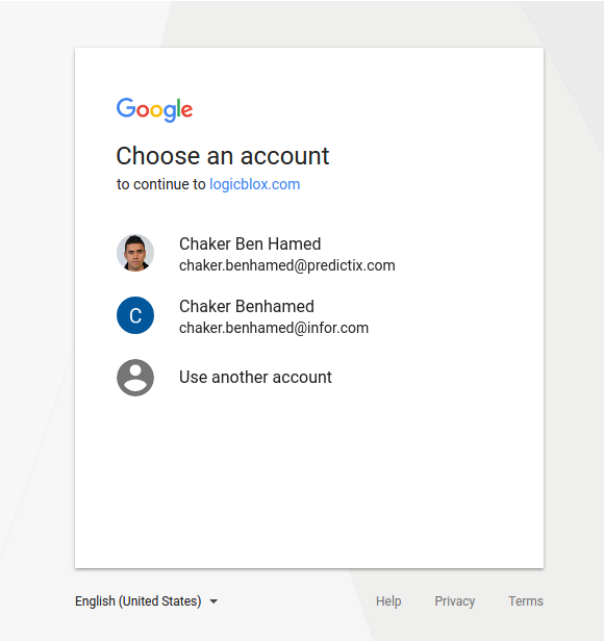

Figure 4.8: user's Google accounts

4.4.5.2 Authorization

Since the application is used to check the performance of Predictix application. Project owner choose to allow all application user the right to check all accounts benchmarks and history. However, for running new benchmarks the user must have the appropriate rights. Thus, we associated an account role to every user. So either the user have a create benchmark right, admin right or it have only read right. Check the entity relationship diagram 4.9 that explain how this is done in the database We use the entity in every user request to check if the user has the appropriate right, If not we use Pyramid exceptions to return the 403 HTTP response to the user.

4.4.5.3 Security

Application performance is confidential internal information that shouldn't be shared out of the company . Thus the application that serve the result should only be accessible to company employees only. For that we used EC2 security groups. EC2 security group [9] are network rules that can be set on Amazon cloud instances. So the application will only be accessibly from certain IPs. We only allowed access to people who are connected to the company internal VPN.

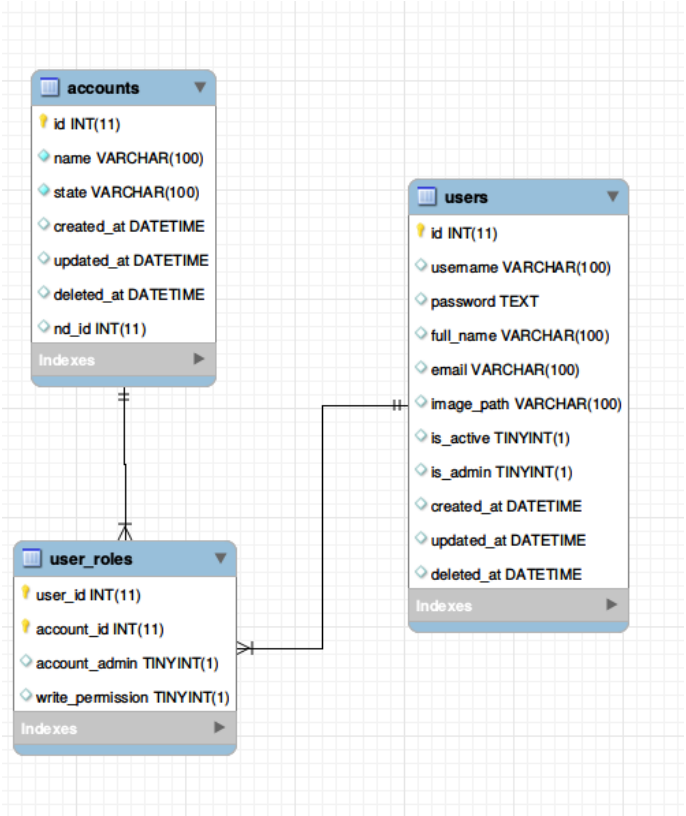

Figure 4.9: User Role entity

On top of that we setup an HTTPS certificate and we redirect all non-secured request to use the HTTPS protocol on port 443. The figure4.10 show How Google Chrome is confirming that the site is using a valid HTTPS certificate.

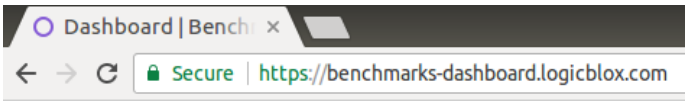

Figure 4.10: The HTTPS certificate view in Chrome

4.5 Final overview

In this section we will list the final state of the development of the application. Figure 4.11 present the entity-relationship diagram we established by the end of sprint 4. The diagram show the different entities of the application. Figure 4.12 on the other hand shows the endpoints of the application API and the documentation of each one, this include the

parameters, URL, payload and the expected result.

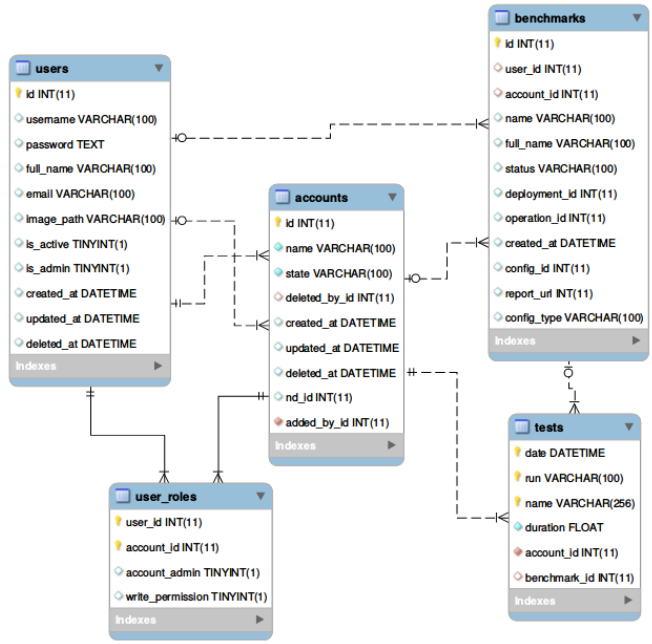

Figure 4.11: Entity relationship diagram

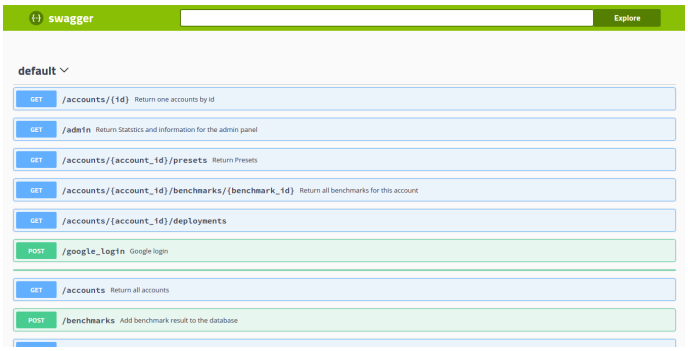

Figure 4.12: API documentation

4.6 Validation tests and Deployment

Hydra is a Nix-based continuous build system that constantly checks out code sources of software projects from version management systems such as Mercurial, to build, test and release them. The build tasks are described using Nix expressions. This allows a Hydra build task to specify all the dependencies needed to build or test a project.

In fact, the code of our application is, currently and constantly pushed in a repository in BitBucket, which is a web-based hosting service for projects that use either the Mercurial or Git revision control systems, such as GitHub. In order to have our application ready to be deployed, we added a file called default.nix to our project, that actually defines the project's nix-expressions. Afterwards, we created a project under Hydra, that we called *Benchmarks Dashboard*. Thus, the *Benchmarks Dashboard* project on Hydra, will be pulling our code from our BitBucket repository along with the default.nix file in order to perform the automated build and unit-tests.

The following are some screen shots about our Hydra project build. The figure 4.13 represents the configuration for our project. We can notice that the links to the different project's dependencies are defined, such as our source code and the nixpkgs repository which contains the definitions for all packages available through the nix package manager.

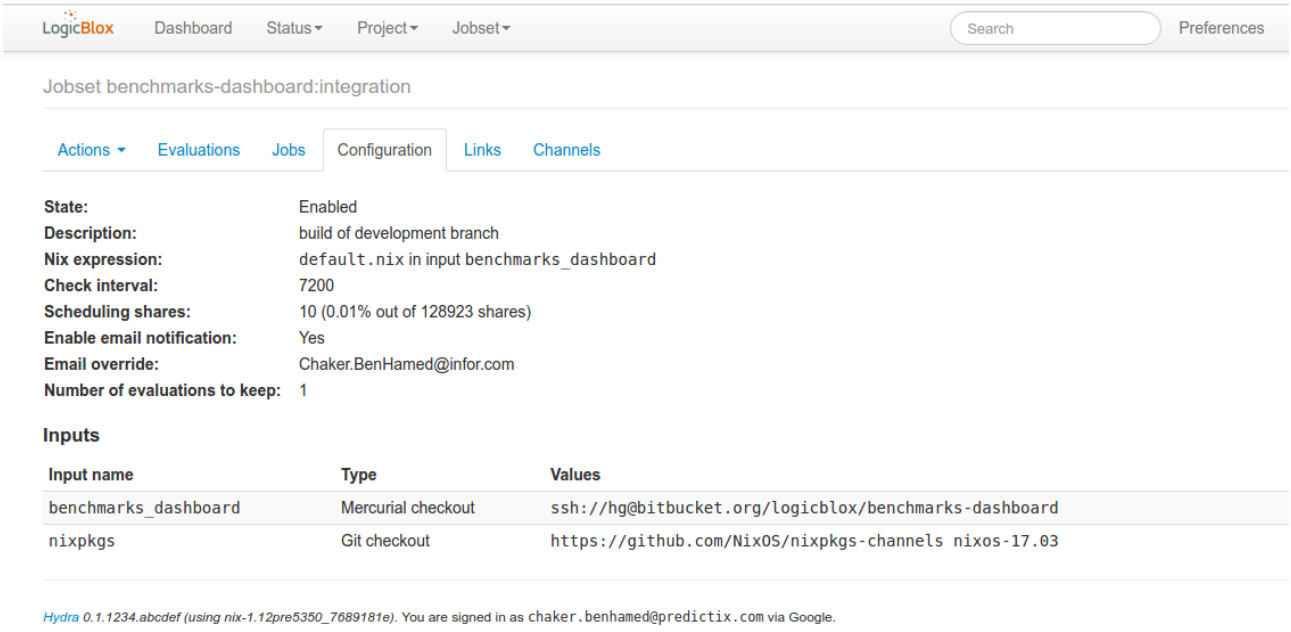

Figure 4.13: Benchmarks Dashboard Hydra configuration screen

The figure 4.14 represents the Hydra evaluation screen, where we can see the history if the build and whether there are errors in the build or not.

In Hydra we can also run unit test and report various indicator about the quality of the application. The figure 4.15 shows the coverage report of the test of our application.

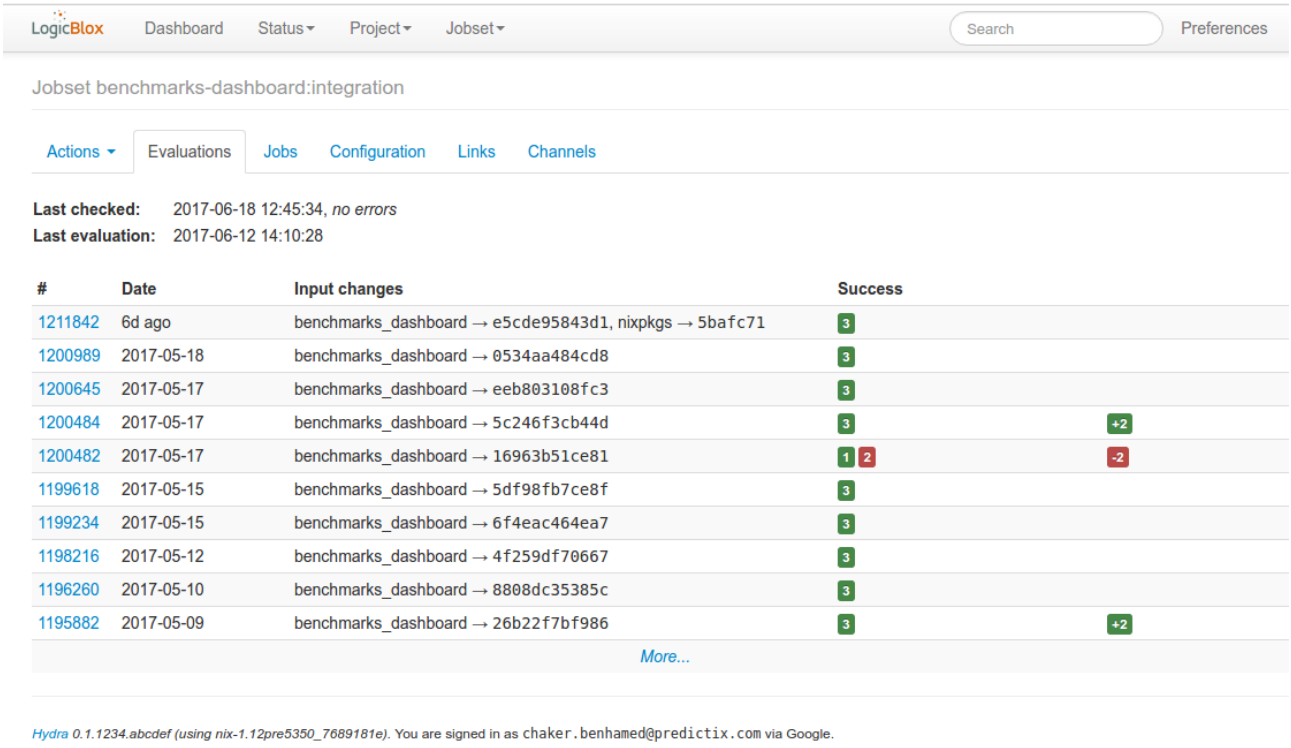

Figure 4.14: Benchmarks Dashboard Hydra evaluations

Coverage report: 88%

Module	*statements*	*missing*	*excluded*	*coverage*
BD_backend/__init__.py	46	3	2	93%
BD_backend/api/__init__.py	0	0	0	100%
BD_backend/api/account.py	88	7	0	92%
BD_backend/api/admin.py	9	0	0	100%
BD_backend/api/benchmarks.py	26	0	0	100%
BD_backend/api/home.py	21	0	0	100%
BD_backend/api/tests.py	55	36	0	35%
BD_backend/api_doc.py	5	0	4	100%
BD_backend/models/__init__.py	28	0	0	100%
BD_backend/models/account.py	21	0	0	100%
BD_backend/models/benchmark.py	33	8	0	76%
BD_backend/models/meta.py	23	0	0	100%
BD_backend/models/user.py	41	0	0	100%
BD_backend/models/user_role.py	14	0	0	100%
BD_backend/mongodb/__init__.py	12	7	0	42%
BD_backend/nixops_dashboard.py	97	7	1	93%
BD_backend/routes.py	4	0	0	100%
BD_backend/security.py	18	2	0	89%
BD_backend/serializers.py	29	0	0	100%
BD_backend/tasks/__init__.py	12	2	0	83%
BD_backend/vars.py	2	0	0	100%
Total	**584**	**72**	**7**	**88%**

coverage.py v4.0.1, created at 2017-06-12 14:17

Figure 4.15: Coverage report

Conclusion

For developing the Benchmarks Dashboard, we opted for the scrum methodology based on incremental development. This section was devoted to the definition of the sprints we fixed as well as the hardware and software technologies that we used. We, also, presented the results of the execution of the application through its interfaces, and our application's deployment process, while giving the performance evaluation.

General conclusion

Big data and Cloud computing is the latest buzzwords in information technology. Having tremendous amounts of complex data that need to be proceed on near real-time performance, can be tricky. So having a clear performance audit is critical to monitor the performance of the application. Also providing the way to the developer to start new benchmark in a click of a button is a major productivity boost.

In the present report, we began by presenting the general context of our project while introducing some important concepts, such as deploying, Benchamrks, workflows and Nix that is used across all of Predictix projects and that we used, as a consequence. We also stated our projects problematic and an overview over its goals. Then, we defined our projects requirements and we specified them, following the scrum methodology. Afterwards, we proceeded with the project design through mockups, while defining our projects static and behavioural aspect. Finally, we described in the last chapter the implementation, performance evaluation and deployment processes and presented an overview of the achieved work.

My internship at Predictix was, undoubtedly, one of the most important and enriching experiences in my life. This experience was rewarding for me in terms of academic achievements and professional experience. Indeed, my internship assignment allowed me to put my academic knowledge into practice and to gain experience in software development. This was accomplished through successfully completing development tasks. But added to these technical competencies, I have also acquired soft skills through the daily experience of working within the team of developers at Predictix as we had the opportunity to work in an encouraging, collaborating and teamwork driven environment of highly skilled engineers.

Bibliography

[1] Nix: A Safe and Policy-Free System for Software Deployment https://nixos.org/ eelco/pubs/nspfssd-lisa2004-final.pdf

[2] Python https://docs.python.org/2/faq/general.html

[3] Mozilla developer network https://developer.mozilla.org/en-us/docs/web/javascript

[4] Atlassian, jira software provider www.atlassian.com

[5] SOA http://aisel.aisnet.org/cgi/viewcontent.cgi?article=3331&context=cais

[6] JSON http://www.json.org

[7] NixOps https://nixos.org/nixops/

[8] EC2 instance types https://aws.amazon.com/ec2/instance-types/

[9] EC2 securiy groups http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html

[10] Grommet http://grommet.io

[11] Grommet Components http://grommet.io/docs/components