Universidad San Sebastian

Facultad de Ingeniería y Tecnología

Earlgrey: Diseño y construcción de un framework JAVA para codificación de aplicaciones orientadas a servicios REST.

Proyecto de titulo presentado para optar al titulo de Ingeniero Civil Informático

Alumno: Angelo Alejandro Calvo Alfaro
Profesor Guía: Dagmar Pearce Poffan

2018

Dedicado a mis padres Haydee y Jose
por su incondicional apoyo a lo largo de mi vida,
y a mis amigos y todas aquellas personas que han
estado conmigo en los momentos mas difíciles.

Angelo

acalvoa@anima$ cat arigato.js

```
let fs = require('fs');
fs.readFile('gracias.txt', 'utf-8', (err, data) =>{
    if(error) console.log('No pude terminarla');
    console.log(data);
});
```

acalvoa@anima$ node arigato.js

Para comenzar con esto, no puedo evitar pensar en una frase que me ha dado energía para terminar este largo camino que comencé hace 11 años atrás y que cada cierto tiempo debo repetir para no olvidar que no todo en la vida a veces ocurre como a uno le gustaría.

"El rió no erosiona las rocas por la fuerza de su caudal, sino que por la constancia de su flujo"

Agradezco a mis padres Haydee Alfaro y Jose Calvo por todo el esfuerzo que han realizado a lo largo de mi vida para que ocurra este momento. Sus palabras e incondicional apoyo siempre han sido una factor clave en mi vida para no arrojar la toalla cuando siento que he perdido la ganas de seguir luchando. Agradezco también a toda mi familia, que nunca perdió la esperanza en mi.

A mi querida Franita, que siempre ha estado ahí cuando mas lo he necesitado, brindándome apoyo y cariño en los momentos cruciales. Sin tus exquisitas comidas, tus infinitas sesiones de cariño y tu ternura, este camino hubiera sido mas difícil de lo que fue.

Agradezco a mis compañeros de equipo y segunda familia; Alejandra Levill y mi hermano Sebastian Gonzalez, que siempre me han apoyado en mis decisiones y todo lo que se me ha ocurrido. Gracias por todo el apoyo y soporte que me han brindado en este ultimo tiempo, el cual fue clave para estar el día de hoy concluyendo este proceso.

A las personas de Controlaría General de la República, con los cuales he convivido día a día los últimos 4 años de mi vida brindándome risas, satisfacciones y nuevos desafíos.

A finalmente a mis amigos del alma Francisco Pereira, Fabian Constancio, Gustavo Plaza, Pablo Cartes, Gabriel Rivera y Santiago Vergara que siempre han estado ahí alentándome, tomando una cerveza a mi lado y dándose el tiempo de estar conmigo cada vez que lo he necesitado, así como todos aquellos que de alguna u otra forma han contribuido a que me halla transformado en el hombre que soy hoy en día.

acalvoa@anima$ sudo poweroff

Abstract

The following document aims to apply the project develop by Angelo Alejandro Calvo Alfaro to opt to Civil Informatic Engineer title on San Sebastian University Located in Santiago, Chile.

The project consist in a JAVA language framework develop for develop services oriented applications in a embed environment of easy implementation with a lot of features thinked in the easy develop of apps. The implementation have the main objective overcome the java language difficulties to make this type of apps, to minimize the gap with others populars languages used to affront this apps type. With this objective, the main characteristics present in frames of other languages will be characterized and the main JAVA language shortcommings will be analyzed to make a framework that will contain the main features identified and add a new features useful to develop or manage apps.

The develop of project will divided in two phases that will implemented in parallel to finalize the project. This fases are the develop of framework named Earl Grey and the build of a production system that implement the framework as a base of construction; This will be a project developed for Contraloria General de la Republica de Chile named "Observatorio Sector Municipal". Both fases will use the agile metodology Kanban to manage the software project and the metodology Gitflow the manage the code branchs.

Resumen

El presente documento pretende exponer el proyecto de Trabajo de Titulo efectuado por Angelo Alejandro Calvo Alfaro para optar al título de Ingeniero Civil en Informática de la Universidad San Sebastian, Sede Bellavista.

El proyecto consiste en la construcción de un framework para el lenguaje JAVA, de código abierto y libre para la comunidad de desarrollo[1], que permita abordar proyectos de desarrollo de software orientados a servicios en un entorno embebido de fácil implementación, con muchas características útiles para el desarrollo de aplicaciones. La implementación tiene como principal objetivo el superar las dificultades del Lenguaje JAVA para construir rápidamente aplicaciones orientadas a servicios[2], para así minimizar la brecha existente con otros lenguajes populares para abordar este fin. Para esto se caracterizaran los componentes útiles presentes en los frameworks de otros lenguajes y se analizaran las deficiencias propias del lenguaje JAVA con el objetivo de formular un framework que contenga las características identificadas y agregue nuevos componentes útiles para efectuar tareas de desarrollo y administración.

El desarrollo del proyecto se dividirá en 2 grandes fases que se implementaran en paralelo para alcanzar la finalización del proyecto. Estas fases son el desarrollo del

[1]El proyecto se encuentra disponible en el repositorio publico ubicado en la siguiente url: https://github.com/acalvoa/earlgrey

[2]Cuando hablamos de aplicaciones orientadas a servicios nos referimos al termino utilizado para referirnos a todas aquellas aplicaciones web que sus componentes han sido estructurados de tal forma, de ofrecer o consumir una biblioteca de operaciones llamadas servicios, los cuales están disponibles en un punto común llamado endpoint. Estos servicios desempeñan operaciones altamente definidas y pueden definirse bajo estandares como soap, jsonapi, etc.

framework que sera llamado Earlgrey y la construcción de un sistema productivo que implemente el framework como base de construcción; Este sera un proyecto desarrollado para la Controlaría General de la República de Chile llamado "Observatorio Sector Municipal". Para ambas fases sera utilizara la metodología ágil Kanban para la gestión del proyecto y la metodología de control de ramas Gitflow.

Índice general

Capítulo 1

Introducción

La amplia adopción de JAVA en la industria desencadeno toda la revolución en el mundo del desarrollo a lo largo de su historia, que ha visto nacer el uso y crecimiento acelerado de la web, así como de los conceptos de Cloud Computing[1]. JAVA es uno de los lenguajes mas usados a nivel mundial y se estima según Oracle, compañía de la cual forma parte este lenguaje, que el número de desarrolladores asciende a 9 millones a lo largo del mundo. JAVA es un lenguaje orientado a objetos, robusto, de fácil aprendizaje que ha permitido la concepción de toda clase de aplicaciones a lo largo de su historia tanto en el ámbito académico como corporativo.

En el ámbito de desarrollo web, JAVA provee su suite J2EE que permite el desarrollo de aplicaciones web a través del uso de Enterprise JAVA Beans[2] y componentes de software que permiten el desarrollo en N capas distribuidas, apoyándose de servidores de aplicaciones, los cuales son capaces de administrar y efectuar despliegues de aplicaciones web.

A pesar de la popularidad de JAVA, su evolución ha sido lenta en el ámbito de desarrollo web, perdiendo competitividad en contraposición de otros lenguaje que se

[1] Cuando hablamos de Cloud Computing, o computación en la nube hablamos de un paradigma de tecnologías de la información que permite ofrecer servicios computacionales a través del uso de una red, que en la mayoría de los casos es internet.

[2] Enterprise Java Beans o EJB por sus siglas, son una de las interfaces de programación que forman parte del estándar J2EE y define como los servidores de aplicaciones proveen objetos desde el lado del servidor.

ha especializados en abordar la construcción de aplicaciones web. Además java ha absorbido de una forma lenta los cambios en las tendencias de programación, siendo uno de los ejemplos mas tradicionales la lenta absorción de las expresiones lambda que fueron introducidas en la versión 8 de java, 5 años después de la adopción de estas por PHP, el cual es uno de los lenguajes mas populares para desarrollar web a lo largo del mundo.

En este contexto otros lenguajes como PHP, Ruby, Python, Node.js y su extensa comunidad han desarrollado toda una familia de frameworks y herramientas para abordar de una forma simple y elegante la construcción de aplicaciones web en distintas metodologías y arquitecturas. Esta diversidad, permite abordar con mucha flexibilidad todo tipo proyectos de software que requieran de arquitecturas de software especificas o híbridas, con una curva de aprendizaje relativamente fácil y una gran comunidad brindando soporte activamente a dichas plataformas.

JAVA en ese contexto se ha quedado levemente atrás, disponiendo de una escasa cantidad de frameworks robustos que permitan abordar proyectos de software de todos los tamaños. En ese contexto, la comunidad dispone de frameworks con una larga historia como Spring o Struts que abordan el desarrollo MVC[3] y han sido difundidos ampliamente como estándares de desarrollo web con JAVA. Herramientas como Jersey y Hibernate se han posicionado como populares componentes que se implementan en proyectos de software con el objetivo de cubrir las características de servicios REST y ORM[4], pero en caso como Hibernate este no es recomendado para abordar proyectos pequeños. A pesar de esto no existe una solución embebida que aborde de manera efectiva la construcción de aplicaciones orientadas a servicios como lo es el caso de Loopback de Node.js. Lo antes mencionado provoca que gran parte de la comunidad ágil de desarrollo no utilice JAVA en proyectos web, debido a a que no es sinónimo de agilidad y por lo complicado de implementar en comparación

[3]Hablamos de MVC para referirnos a una arquitectura de software llamada Modelo - Vista - Controlador, la cual plantea el desarrollo en 3 capas altamente definidas para interacción con el usuario (Vista), procesos de negocio (Controladores), así como datos y persistencias (Modelo).

[4]ORM es la sigla referida para referirnos a Object Relational Mapping, la cual es una técnica utilizada para convertir datos entre entidades orientadas a objetos y una base de datos que en la mayoria de los casos es de tipo relacional.

con otros lenguajes.

A pesar de esto, muchas empresas del rubro tecnológico que originalmente partieron usando JAVA han dado un vuelco volviendo a utilizar JAVA en sus proyectos, debido a múltiples beneficios que otorga el uso de este lenguaje de programación, lo que nos hace formular la pregunta ¿Es JAVA un mal lenguaje para abordar proyectos web o simplemente no dispone de las herramientas adecuadas para ser parte de las tecnologías que se caracterizan por implementar desarrollos de una forma simple y ágil?.

El presente trabajo tiene por objetivo el abordar la construcción de un nuevo framework JAVA que permita adoptar muchos de los componentes útiles presentes en frameworks de otros lenguajes y superar muchas de las deficiencias que tiene el lenguaje JAVA a la hora de abordar proyectos que opten por usa una arquitectura orientada a servicios RESTful[5].

[5]Los servicios RESTful, son un tipo de servicios web que implementan la arquitectura REST. Por sus siglas REST significa transferencia de estado representacional y es un tipo de mensaje que contiene toda la información necesaria para comprender la petición, a diferencia de sus antecesores como SOAP o RPC. REST se caracteriza por utilizar los verbos de petición HTTP (GET, POST, PUT, DELETE, PATCH) para definir operaciones altamente individualizadas.

Capítulo 2

Definición del problema

Cuando efectuamos una mirada al desarrollo tecnológico empresarial actual, notamos que existe una gran cantidad de necesidades en distintos sectores económicos, las cuales, a través de la implementación de tecnología, permitirían obtener un nivel de desarrollo mayor al actual. Un estudio efectuado por el ministerio de Economía [8] reveló que la totalidad de los sectores económicos se ven beneficiados por la implementación de las tecnologías de la información y comunicaciones, mas bien llamadas TIC's, en su cadena de valor, generando un impacto que va desde mediano a grande. Lo anterior, mediante la implementación de medidas a corto plazo que permitan incorporar TIC's en nuevos procesos de negocio y de tecnologías que apoyen las labores que tradicionalmente son efectuadas en ausencia parcial o absoluta de las TIC's.

Las necesidades de la industria tradicional han generado como efecto tres consecuencias que marcaron tendencias durante los últimos años.

Como primer elemento, podemos observar un acelerado crecimiento de empresas y entidades que han comenzado a hacer de las TIC's parte fundamental de sus actividades cotidianas, generando soluciones y nuevos productos que, en la mayoría de los casos, tienen algún componente tecnológico o son fundamentalmente de índole tecnológica.

Como segundo punto, observamos un crecimiento exponencial del área de desarro-

llo de software, que se ha vuelto una pieza fundamental a la hora de acompañar el desarrollo de nuevas tecnologías para la industria. Mediante la creación de nuevos lenguajes, metodologías, arquitecturas y tecnologías se ha incrementado la capacidad de la industria tecnológica para generar software y hardware acorde a las necesidades de la industria que están en constante evolución.

Como tercer punto, se observa un notable incremento en el numero de empresas del sector TIC's [5] motivado por diversos factores a nivel mundial que ha generado un creciente incremento de los puestos de trabajo requeridos por esta industria, así como una amplia y creciente lista de habilidades técnicas que los candidatos y empleados pueden ofrecer a estas empresas, dándose una amplia combinatoria de caminos mediante los cuales una empresa puede llegar a los resultados que requieran.

En ese contexto, uno de los factores fundamentales que una empresa considera a la hora de construir software[9] es la disponibilidad tecnológica de infraestructura, así como las habilidades técnicas de sus recursos humanos, que en muchos casos son especializados en ciertas tecnologías y metodologías. Es así, como los costos asociados a la adopción y adquisición de nuevas tecnologías, también llamados costos de reemplazo, son el principal factor racional a la hora de decidir respecto a la implementación de las mismas. Lo último provoca, como consecuencia, que estas empresas, en gran medida, sigan desarrollando soluciones informáticas en tecnologías que ya tienen implementadas. Esta decisión es respaldada fuertemente por factores psicológicos de la organización que en muchos casos sienten, con la tecnología que ya utilizan, una sensación de seguridad en diversos aspectos.

JAVA es un lenguaje de programación robusto, orientado a objetos, con mas de 20 años en el mercado de desarrollo, que ha lo largo de su historia ha sido ampliamente adoptado por la comunidad de desarrollo, evolucionando junto con la aparición de nuevas tecnologías, que han visto el explosivo crecimiento de internet, las comunicaciones, así como los conceptos que definen un software. Si bien JAVA en un comienzo no fue pensado en la web sino en el concepto de portabilidad -lo cual permitía que los desarrolladores escribieran software solo una vez sin importar el dispositivo- ha

ido mutando a lo largo del tiempo con el objeto de hacer la web y el uso empresarial uno de sus principales focos, para lo cual desarrolló herramientas y frameworks que permiten el desarrollo distribuido, apoyándose de servidores de aplicaciones que permiten el despliegue de aplicaciones web en componentes modulares. JAVA es un lenguaje de tipado estático de fácil aprendizaje, por lo que es muy utilizado en docencia para enseñar a programar debido a que su sintaxis es relativamente flexible y encapsulada, muy similar en ese contexto a C y C++, lo que permite adoptarlo con facilidad. Estas características, sumado a su larga historia en el mercado del desarrollo que ha generado toda una familia de aplicaciones y herramientas con un robusto soporte, han hecho de JAVA un verdadero estándar para la concepción de aplicaciones empresariales en el sector corporativo.

La amplia adopción de JAVA por las corporaciones ha generado la concepción de múltiples frameworks, herramientas y metodologías para la construcción de aplicaciones web, la cual ha tenido un auge en los últimos años en favor de las aplicaciones convencionales debido a la capacidad de utilizar los beneficios del concepto Cloud Computing otorgando entre otras características disponibilidad, compatibilidad, interoperabilidad y la capacidad de operar con múltiples usuarios concurrentes.

Tecnologías como JAVA EE y Spring entre otras, han surgido como respuesta a esta necesidad, implementando distintos paradigmas entre los cuales destaca el patrón modelo, vista, controlador. Así mismo, plataformas como los contenedores de aplicaciones Weblogic de Oracle, JBOSS de Redhat y Tomcat de Apache fueron construidos en consecuencia a la necesidad de manipular, monitorizar y albergar estas aplicaciones de una forma simple, así como efectuar operaciones altamente necesarias para entregar capacidades de disponibilidad, tolerancia a fallos y respuesta, mediante el uso de técnicas que incluyen el balanceo multinodo entre otros.

Sin embargo, a pesar su amplia popularidad en el sector corporativo, en los últimos años han surgido nuevas tecnologías que desde su concepción fueron pensadas en el desarrollo web y mobile, recopilando la experiencia de la comunidad e integrando características que las transforman en potentes herramientas a la hora de abordar

proyectos web que utilizan diversos paradigmas entre los cuales destaca la arquitectura orientada a servicios que permite, entre otros beneficios, transferir parte de la carga de la aplicación a los clientes, que actualmente poseen equipos con recursos no despreciables, debido a la reducción de costos que ha sufrido la fabricación de hardware.

La arquitectura orientada a servicios o SOA, es un modelo de desarrollo de software mediante el cual se elaboran librerías o repositorios de funciones en una maquina que actúa como servidor, las ejecutan una función en particular altamente definida. La figura 2.1 muestra la interacción entre los componentes de una arquitectura orientada a servicios, en la cual se puede apreciar un proveedor de servicios que actúa como entidad que disponibiliza métodos y los publica en un registro o catalogo de servicios. Esto métodos efectúan operaciones a nivel remoto para clientes que consumen dichos servicios, interactuando con el proveedor de servicios mediante normas definidas en contratos de servicios y protocolos dentro los cuales los mas populares son SOAP[1] y REST[2]. Estos métodos son registrados o expuestos en bibliotecas o registros de servicios que pueden ser consultados por los clientes o consumidores de servicios con el objetivo de conocer las posibilidades que disponen. SOA siempre ocurre en una interacción entre un cliente que consume y un servidor que disponibiliza servicios para lograr un fin, unidad o sistema en particular.

El desarrollo de SOA ha promovido la concepción de tecnologías, basadas en la arquitectura de servicios, que actúan como cliente consumiendo información y operaciones desde un servidor de datos, el cual provee servicios HTTP mediante nomenclatura RESTful[12]. Así, tecnologías como Angular, ReactJS, Vue.js entre otras han cambiado de manera radical la forma mediante la cual se construyen aplicaciones, transformando las tradicionales aplicaciones monolíticas MVC, en sistemas distribuidos

[1]SOAP o Simple Object Access Protocol, es un popular protocolo utilizado en la arquitectura SOA para establecer un estándar de comunicación entre clientes y proveedores de servicios que utilizan mensajes en formato XML. Los contratos son establecidos en formato WSDL, los cuales descriptores de los servicios y su uso por parte de clientes que lo consumen.

[2]REST, hace referencia a un tipo de arquitectura orientada a servicios en la cual los mensajes tienen toda la información necesaria para efectuar una petición, sin necesidad de utilizar intermediaciones. Los contratos son inferidos en base a convenciones en vez de un elemento estructurado como el WSDL de SOAP.

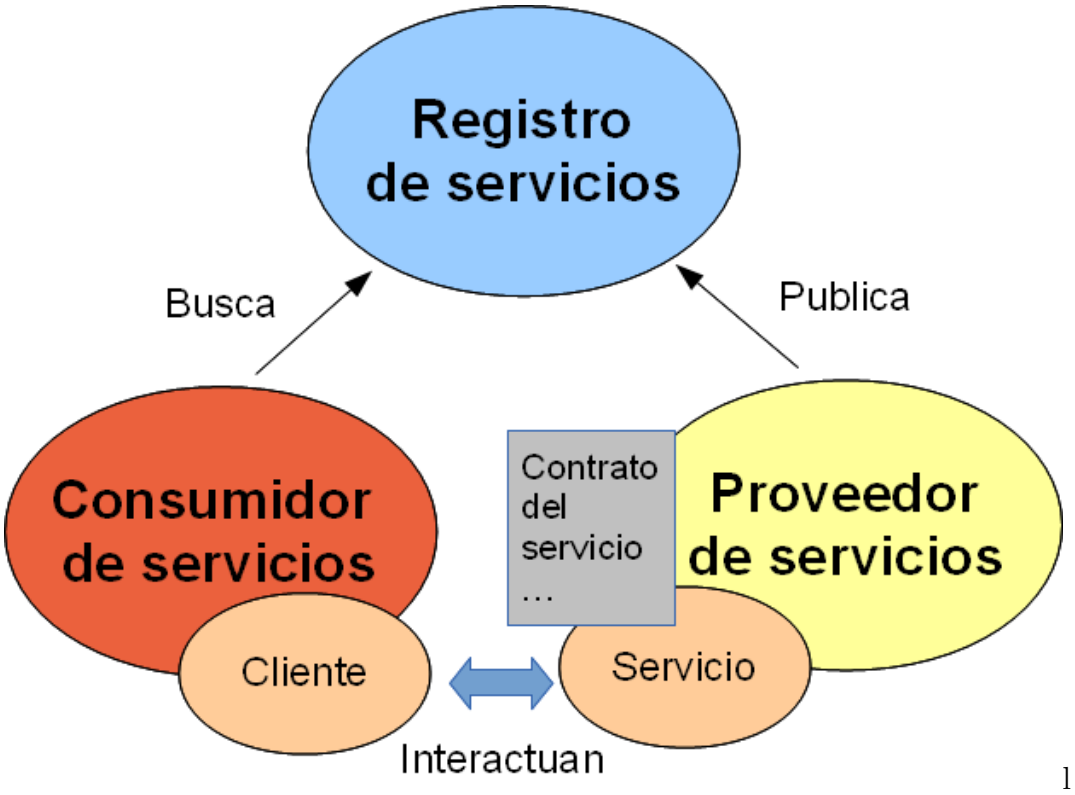


Figura 2.1: Figura muestra los componentes de una arquitectura orientada a servicios.

mediados por servicios. Así mismo, el desarrollo de nuevas metodologías y la necesidad de reducir las líneas de código fuente para obtener los mismos resultados, en beneficio de ganar tiempo en la ejecución y planificación de proyectos, ha desatado toda una revolución en el área de desarrollo que busca la optimización de los procesos de codificación.

Esta tendencia ha llevado a que la comunidad desarrolle activamente en otros lenguajes, como Nodejs, Python, Ruby, PHP, innovadores frameworks y herramientas que permiten desarrollar aplicaciones orientadas a servicios en una fracción del tiempo en comparación con métodos tradicionales, sin perder calidad en la construcción y ejecución de software, especializando dichos lenguajes en la construcción de aplicaciones web. Este trabajo, se traduce en una reducción de costos a la hora de construir software en las empresas que decidan adoptar estas tecnologías, las cuales general-

mente son aquellas tecnológicamente más vanguardistas, derivando en una ventaja competitiva a la hora de fabricar productos en contraposición a otras firmas tecnológicas mas conservadoras.

Si analizamos JAVA EE y su infraestructura, notamos que provee de una robusta pero compleja forma de escribir aplicaciones, que hace abuso de la redundancia de código y genera en la mayoría de los casos una baja escalabilidad y mantenibilidad de las aplicaciones. Esto, sumado a la gran flexibilidad para integrar dependencias y arquitecturas de aplicación, genera en la mayoría de los casos problemas de incompatibilidad a la hora de desplegar o migrar aplicaciones, elevada complejidad a la hora de efectuar correcciones o nuevas características si no han sido adoptadas normas estructuradas de codificación, y una dificultad para construir aplicaciones con fluidez desde cero debido a lo complejo que puede llegar a ser la construcción de un ambiente de desarrollo.

Si bien, se han implementado soluciones como Jersey, el cual es una herramienta para poder desarrollar servicios REST en JAVA y resuelve el problema de la construcción de servicios en cierta forma sobre una base JAVA EE, este lenguaje se sigue careciendo de un componente o framework que provea las estructuras y capacidades de modernos frameworks que posee una gran cantidad de características y operaciones integradas e implementadas en otros lenguajes, como Sailsjs, Django, Rails o Laravel, los cuales compiten por el mercado de la web y proveen entornos robustos de desarrollo con una rápida implementación. Si bien se puede obtener en JAVA un concepto similar mediante la integración de múltiples librerías y componentes, se requiere de conocimientos especializados y un correcto diseño de arquitectura para evitar problemas a la hora de construir un ambiente de desarrollo o efectuar despliegues.

En base a estos antecedentes, JAVA EE tiene desventajas en contraste con sus competidores, lo cual - sumado a que en la mayoría de los casos existe la necesidad del uso de contenedores de aplicaciones para poder desplegar aplicaciones web, ya que

estas no son auto ejecutables[3]- puede provocar que JAVA, a pesar de sus elementos positivos, no sea considerado o aceptado a la hora de implementar aplicaciones que utilizan tecnologías RESTful, donde se busca un entorno de rápido desarrollo y fácil implementación.

A pesar de esto último, un articulo publicado por la web especializada en tecnología GenBetaDev[7] reveló que grandes empresas tecnológicas que empezaron sus operaciones con lenguajes modernos han comenzado a utilizar JAVA debido a múltiples factores entre los cuales destacan el uso de máquina virtual, que permite entre otras cosas otorgar compatibilidad con otros lenguajes especializados que pueden compilar bytecodes[4] como Scala, Groovy, Clojure, entre otros; así como la alta oferta de desarrolladores JAVA en contraposición a los de otros lenguajes.

En base a lo anterior, existe la necesidad de disponer de un framework JAVA embebido que implemente las últimas tendencias de desarrollo y características ágiles, otorgando la capacidad de desarrollar aplicaciones en un tiempo menor al que usualmente se utilizaría en JAVA. Esto permitiría adoptar características de otros frameworks construidos por la comunidad, evitar problemas relacionados con el exceso e incompatibilidad de dependencias o el diseño incorrecto de una arquitectura de software y componentes que en la mayoría de las ocasiones hacen poco escalable un proyecto de software y su mantención evolutiva debido a la baja cohesión de sus elementos, establecer una forma clara de escribir software bajo normas estructuradas y disminuir los tiempos de desarrollo y preparación mediante el uso de características integradas sin perder la capacidad de despliegue en contenedores de aplicaciones tradicionales o tecnologías que actualmente usan las compañías; con el objetivo de utilizar la misma infraestructura ya existente.

[3]En este contexto se han desarrollado tecnologías como Spring boot, que es un framework que permite ser desplegado sin la necesidad de tener un contenedor de aplicaciones.

[4]Los bytecodes son un tipo de archivo compilado intermedio que puede ser interpretado por un interprete como la maquina virtual JAVA o JVM, para generar código binario en tiempo de ejecución, bajo un tipo de compilación llamado JIT o Just in Time Compilation

Capítulo 3

Objetivos del proyecto

3.1. Objetivos generales

El presente trabajo tiene por objetivo principal la construcción de un framework liviano basado en la arquitectura orientada a servicios REST para el lenguaje JAVA denominado Earlgrey, de código abierto y disponible para la comunidad, que integre características de otros frameworks existentes en otros lenguajes como Nodejs, Python, Ruby y PHP, permitiendo el desarrollo de aplicaciones de forma ágil y a una mayor velocidad a la que actualmente se desarrollan aplicaciones JAVA J2E tradicionales. Para este fin se plantearan los siguientes objetivos:

- Identificación y caracterización de los principales componentes presentes en otros frameworks que sean útiles y de común uso entre estos, con el fin de considerarlos en el diseño del framework. El propósito de esto es listar, evaluar y priorizar el desarrollo de características presentes en otros frameworks para dotar al lenguaje JAVA de posibilidades iguales o similares a la de otros lenguajes.

- Identificación de las dificultades propias del lenguaje JAVA con el objeto de abordarlas y proponer mejoras que permitan potenciar y facilitar su uso, para las limitaciones o dificultades características de un proyecto escrito en este lenguaje.

- Diseño y construcción, a partir de los componentes identificados, de la primera versión estable del framework Earlgrey, que permita el desarrollo de aplicaciones orientadas a servicios con independencia de la infraestructura tecnológica de despliegue que se ocupe; dotando, tanto a contenedores de aplicaciones como a implementaciones simples de JAVA, de la capacidad de utilizarlo.

- Construcción de 2 herramientas complementarias que permitan asistir al desarrollador en sus labores de codificación, con la finalidad de disminuir la cantidad de código que se escribe para lograr resultados con el framework.

- Desarrollo e Implementación un sistema productivo que utilice el framework Earlgrey y utilice las características proveídas por este para desarrollar aplicaciones que utilicen la arquitectura orientada a servicios REST.

3.2. Alcances del proyecto

Debido a la naturaleza tecnológica del proyecto y la evolución del lenguaje a lo largo de los años, el proyecto se abordará con los siguientes alcances:

- El framework Earlgrey no soportará todas las versiones de JAVA, utilizando como cota inferior la versión 7 en adelante. Esto es debido a que el framework hará uso de las características integradas y presentes sólo desde la versión 7 de JAVA en adelante, las cuales facilitarán su codificación.

- En framework no se podrá desplegar en contenedores de aplicaciones que usen como base una versión de JAVA inferior a la 7.

- El framework sólo soportará la arquitectura orientada a servicios. Si bien no se descarta a futuro la posibilidad de dar soporte a otros tipos de arquitecturas, este proyecto solo abordará la construcción de aplicaciones orientas a servicios.

Capítulo 4

Planificación

Para desarrollar las actividades y los objetivos planteados previamente, el proyecto se dividió en 13 etapas altamente definidas en 10 meses de ejecución. La primera fase de actividades consistió en entender y definir el problema, así como analizar y diseñar una solución a la problemática propuesta. Durante esta fase se abordaron 5 actividades en cascada. Posterior a la fase de estudio se abarcaron las actividades referentes a la fase de desarrollo y preparación previa a este, abordando en paralelo el desarrollo del framework en cada una de sus versiones, la aplicación portal "Observatorio municipal", así como la interfaz de comandos (CLI) y el arquetipo Seed.

Desde la versión 0.4 del framework, se incluyo la participación de 3 colaboradores efectuando trabajos de QA[1] y colaboración directa efectuando correcciones o fragmentos de código. Así mismo, la codificación del componente CLI fue efectuado por Sebastian Gonzalez Villena.

La figura 4.1 muestra la carta gantt desarrollada por el proyecto, describiendo el proceso que se llevo a cabo por el presente trabajo de titulo y sus etapas.

[1] QA o Quality Assurance, es un proceso por el cual se efectúa control de calidad de los productos entregados, asegurando que estos efectúen los procesos que describen sin errores o incidencias.

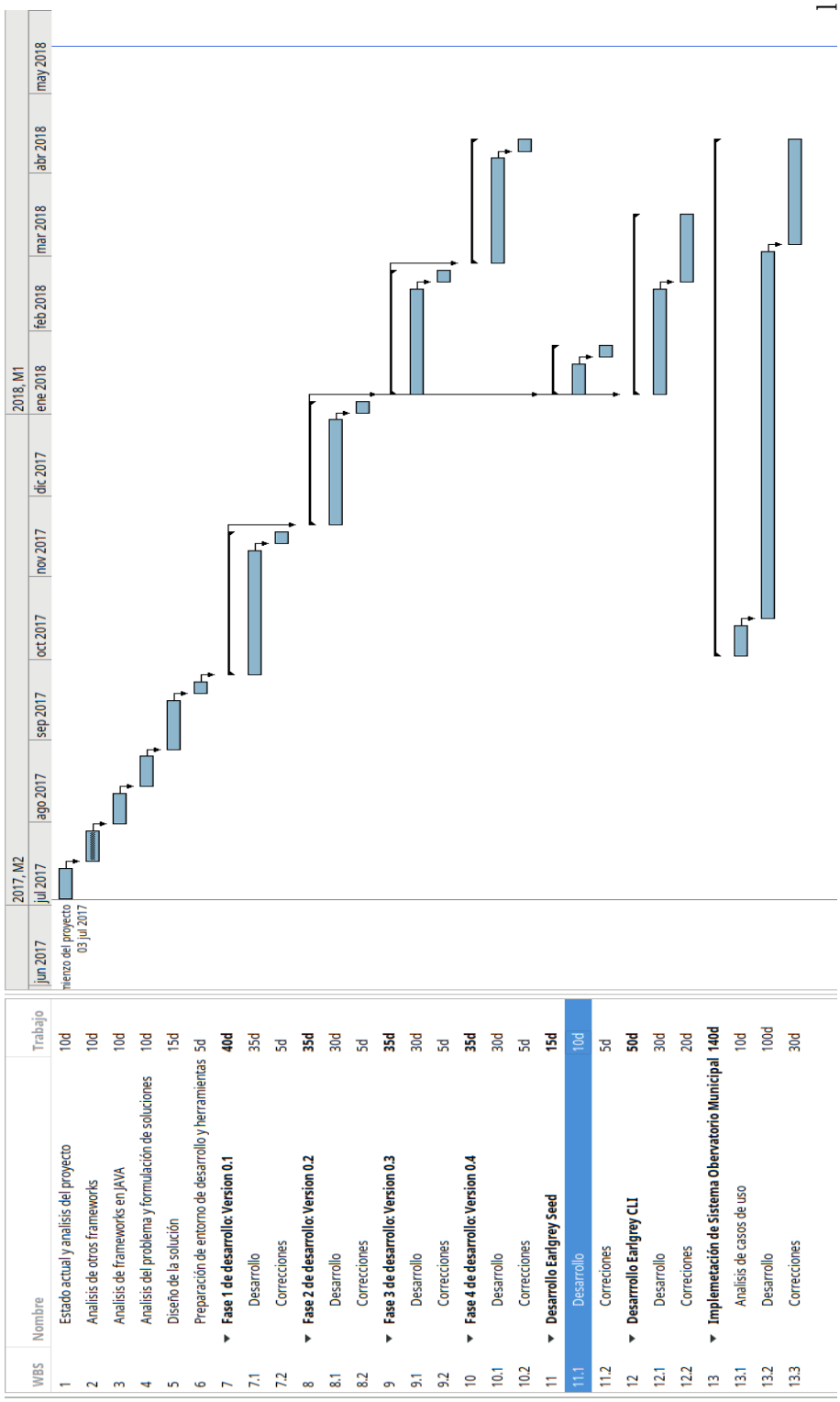


Figura 4.1: Figura que muestra la carta gantt con la planificación del proyecto.

Capítulo 5

Estado del arte

Actualmente, en el mundo del desarrollo existen muchas alternativas para afrontar un desarrollo TI orientado a servicios. Para dar una completa mirada al problema es imperativo observar como este es abordado por otros lenguajes . Si bien los otros lenguajes otorgan una gran flexibilidad a la hora de elegir que solución implementar para construir una aplicación en particular, existen ciertas soluciones que se destacan del resto a la hora de construir las aplicaciones desde cero o "From scratch".

5.1. El estado del arte en otros lenguajes

En este contexto destacan -y con una enorme comunidad respaldándolos- frameworks como Ruby on Rails de Ruby, Phalcon y Laravel de PHP, SailsJs, Loopback y Express de Node.js y Django de Python entre otros, los cuales han logrado posicionarse rápidamente en la comunidad de desarrollo por su capacidad de emprender proyectos con una agilidad sorprendente.

5.1.1. Node.js

Node.js fue construido con un solo propósito, el cual es utilizar el lenguaje ECMAScript[1] desde el lado del servidor. Para este fin, su núcleo está basado en el motor

[1] El ECMAScript es un lenguaje de programación asíncrono orientado a eventos comúnmente llamado Javascript

V8 de Google. En este contexto y analizando las posibilidades disponibles para este lenguaje, tenemos a Sailsjs[1], el cual es uno de los frameworks mas populares para Node.js. Este framework se ha caracterizado por automatizar e integrar en su núcleo muchas de las tareas que normalmente necesita un desarrollador, con un patrón MVC, bastante familiar a otros frameworks como Ruby on Rails. Una de sus características más relevantes es la capacidad de construir API's dirigidas por modelos fuertemente escalables y simples, que permiten construir aplicaciones orientadas a servicios. Además, Sailsjs provee herramientas complementarias como su Command Line Interface, o CLI, que permite entre otras cosas, generar código a partir de la consola de texto. Esto transforma la creación de nuevos componentes o la creación de un proyecto en una tarea que solo lleva unos minutos de espera mientras el CLI hace su trabajo, permitiendo fabricar software de calidad y con un alto grado de confiabilidad de forma rápida, ágil y simple.

La premisa de este framework es que las aplicaciones siempre estén listas para ser desplegadas en entornos productivos, o lo que en la práctica se denomina "Production-Ready", característica que lo hace potente a la hora de efectuar maquetas o prototipos en base a iteraciones. Adicionalmente, pone a disposición un potente componente ORM (Object Relational Mapping)[2] multiparadigma denominado Waterline que permite operar con múltiples bases de datos y provee una gran cantidad de operaciones para efectuar persistencia y búsqueda de información.

Existe, como alternativa a Sailsjs, una solución que resuelve efectivamente la necesidad de generar una API de servicios escribiendo una cantidad de código reducida. Nos referimos a Loopback, un framework actualmente en propiedad de IBM, que tiene como uno de sus principales beneficios el ser pensado y especializado para la construcción de API's, utilizando como principal característica la construcción de API's dirigidas por modelos o lo que en la práctica se denomina "Model Driven Api's", que permite escribir puntos de conexión e interacción con modelos de datos a

[2]Un componente ORM es un componente de software que permite efectuar operaciones sobre un componente de persistencia de información como una base de datos, a través de interfaces estandarizadas, de forma que la construcción de software no este ligada al componente de persistencia que se utilice.

través del uso del concepto RESTful y sus operaciones HTTP. Al escribir un modelo de datos Loopback automáticamente este construye un set de características que permiten efectuar operaciones CRUD sobre un componente de datos que puede ser generalmente una base de datos. Además de lo último mencionado, automáticamente provee una herramienta gráfica que permite manejar y configurar dinámicamente las operaciones y modelos disponibles en la API de servicios, dotándolo de un potente mecanismo para efectuar operaciones en caliente. También provee un entorno CLI que permite escribir modelos con una gran rapidez, bajos estándares de codificación y buenas prácticas, a través del uso de la consola.

Otra de las grandes propiedades del lenguaje Node.js es su gestor de paquetes NPM, que permite integrar y almacenar de forma centralizada todo un repositorio de piezas de código efectuados por la comunidad, transformando a Node.js en un potente lenguaje a la hora de integrar componentes que disminuyan las tareas de codificación.

5.1.2. PHP

PHP es uno de los lenguajes con mayor aceptación para el uso en desarrollo web, con una estimación de alrededor de 20 millones de sitios web y plataformas funcionando en el mundo escritas en este lenguaje. Es un lenguaje de programación multiparadigma que fue diseñado para la creación de contenido web dinámico, siendo uno de los primeros lenguajes de programación que podía ser embebido en un documento HTML.

Actualmente, la popularidad de PHP sigue siendo elevada debido a su fácil aprendizaje y una gran comunidad de desarrolladores a lo largo del mundo, permitiendo la concepción de múltiples herramientas de desarrollo y frameworks que han promovido aún más su uso. Entre las alternativas que mas destacan, dentro de PHP, notamos Laravel y Phalcon.

Si analizamos a Laravel, es un framework que fue creado con el objetivo de desarrollar aplicaciones y servicios web de una manera elegante y simple. Para este fin, dispone del uso de herramientas como el ORM Eloquent, el Motor de plantillas Blade -que

permite una clara y fácil implementación del patrón MVC-, así como múltiples herramientas dentro de las cuales destaca el uso de Composer y la posibilidad de utilizar la Artisan Console que es el CLI desarrollado para Laravel que pretende estructurar y generar código en un proyecto. El objetivo principal de Laravel es evitar el código espagueti[3] para lo cual promueve una sintaxis de construcción simple, elegante y con baja redundancia.

Por otro lado, Phalcon PHP fue pensado en el rendimiento y se presenta como un framework PHP escrito en C que tiene como principal misión ser mas rápido y procesar una mayor cantidad de peticiones en comparación con otras soluciones en el mismo lenguaje como, por ejemplo, CodeIgniter. Proveyendo una sintaxis muy fluida, elegante, así como la posibilidad de escribir API's REST con gran facilidad. Phalcon otorga una mayor flexibilidad por lo que no impone el uso de ORM pero si hace uso de modelos de datos que posteriormente deben ser gestionados bajo consultas de bases de datos. El resultado es un framework ágil y flexible, con una gran escalabilidad.

5.1.3. Phyton

Phyton es un lenguaje de programación multiparadigma y multipropósito, que fue pensado para escribir un código altamente legible y natural en su comprensión. Para ello es un lenguaje altamente estructurado y propone una serie de postulados bajo los cuales un código se considera phytónico o no phytónico [4]. Phyton es muy popular en la comunidad de desarrollo siendo uno de principales lenguajes utilizados a lo largo del mundo.

Ahora, cuando pensamos en un framework para el desarrollo web en Phyton, de forma inherente hacemos referencia a Django el cual se convirtió en el framework

[3]El código espagueti, hace referencia a un conjunto de malas prácticas de codificación que dan como resultado código poco legible, carente de estructura y difícil de mantener. Estas malas prácticas que están en su mayoría basadas en una nula aplicación de un estándar de buenas prácticas[11] generan código que, debido a su naturaleza enredada, hace referencia a los espaguetis.

[4]Establecemos como Phytónico o no Phytónico a aquellos códigos que siguen los principios de legibilidad y transparencia de phyton

por excelencia utilizado en este lenguaje para la construcción de aplicaciones web y servicios. El propósito principal de este framework es la reutilización de código y la escritura de la menor cantidad posible de este para la obtención de resultados altamente escalables. Para esto proporciona múltiples herramientas como lo es un ORM, un motor de plantillas MVC, una robusta API de base de datos y un middleware que permite la extensión de este. La comunidad detrás de Django es enorme y se ha encargado de mejorar día a día este framework y sus posibilidades, proporcionando entre otras cosas diversos CLI y componentes de código reutilizables a modo de plugins.

5.1.4. Ruby

Ruby es un lenguaje de programación orientado a objetos, que fue pensado en la belleza del código proporcionando una sintaxis simple y elegante al escribirla y natural al ser leída por parte del usuario. En consecuencia, ruby fue pensado en la productividad y la simpleza a la hora de desarrollar nuevas aplicaciones.

El framework web por excelencia utilizado en Ruby es Rails, denominado Ruby on Rails, el cual es un framework que sigue el patrón MVC y postula que incluye todo lo que necesitas para crear un aplicación. Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones productivas escribiendo una menor cantidad de código que en otros frameworks y con configuraciones mínimas. Para esto dispone de una serie de componentes que han sido creados por la comunidad distribuidos en formato de RubyGems, que permiten elaborar complejas aplicaciones uniendo y haciendo uso de componentes en formato Plug and Play. La comunidad detrás de Ruby on Rails ha ido creciendo a lo largo de los años, lo que ha popularizado el uso de este framework.

5.2. El estado del arte en JAVA

Cuando pensamos en JAVA, rápidamente se nos viene a la mente un lenguaje que ha tenido una larga historia y que originalmente fue pensado en la portabilidad de plataformas. JAVA pretendía resolver un problema común en un tiempo en donde

existían una gran cantidad de tipos plataformas y arquitecturas de hardware distintas. Este problema generaba que un mismo componente de software tuviera que se escrito y compilado tantas veces como numero de sistemas de arquitectura se deseara abordar. En este contexto, JAVA, postuló el uso de solo un código y el uso de un intérprete intermedio que se denominó máquina virtual o JVM (Java Virtual Machine), la cual era capaz de leer código escrito y compilado en JAVA, llamado Bytecode y transformarlo en código máquina a través del uso de un compilador JIT [5]. Este concepto resultó revolucionario para la industria de fabricación de software que comenzó a adoptarlo en sus desarrollos para escribir aplicaciones.

JAVA es un lenguaje orientado a objetos de fácil aprendizaje y con una sintaxis muy similar a C y C++, lo que resulta en un lenguaje muy natural al programador que no tarda demasiado tiempo en aplicar su sintaxis. A pesar de todo esto, JAVA ha tenido que evolucionar a lo largo de su historia junto con el desarrollo de internet, lo que le ha hecho experimentar grandes cambios en su propio lenguaje y en la forma de entender el mismo. JAVA ha ido evolucionando lentamente en pro de la fabricación de aplicaciones web, un área que ha sido muy potenciada dentro de este lenguaje a través de la disposición de una gran cantidad de herramientas que responden a la necesidad de usuarios y corporaciones que necesitan de una plataforma segura y confiable para desarrollar aplicaciones web. No obstante, esta evolución ha sido lenta, y la adopción de nuevas características a una tasa relativamente lenta, como por ejemplo las expresiones lambda que fueron implementadas en JAVA 8 -a pesar de estar hace años disponibles-, causa que la comunidad de nuevos desarrolladores se decida a favor del uso de lenguajes más dinámicos y con una actualización más frecuente en pos de la adopción de nuevas características y tendencias. Aún así, JAVA es la opción favorita a la hora de abordar desarrollo corporativo debido a su robustez, sumado a su JVM que en la actualidad es capaz de ejecutar y trabajar con distintos lenguajes como Groovy, Scala o Clojure.

[5] El compilador JIT o Just in Time es un tipo de compilador que ejecuta labores de compilación a código máquina en tiempo de ejecución, normalmente es el tipo de compilación utilizada por los intérpretes de código para generar código maquina.

5.2.1. Frameworks en JAVA

En la actualidad, según un articulo publicado en la web OpenWebinars[2] sobre cuales son los 7 frameworks JAVA con mayor popularidad en la industria durante el año 2017, el framework con mayor popularidad es Spring MVC que representa un robusto componente para el desarrollo de aplicaciones en tres capas o a través de sus módulos de diversos aspectos de construcción que un usuario desee abordar. Con Spring es posible abordar la construcción de API's REST mediante la implementación de uno de sus módulos que permiten el despliegue de servicios REST.

Por otro lado, tenemos a Jersey, un toolkit[6] que ha ido ganando popularidad a lo largo de los años orientado al desarrollo de servicios REST. Una de sus principales características es la simpleza con la cual permite desarrollar API's con bastante facilidad y rapidez. Jersey sólo provee las interfaces para desplegar servicios webs, por lo que no entrega ninguna otra herramienta extra para efectuar desarrollo.

Finalmente, tenemos el framework proporcionado por JAVA, este es J2E o JAVA Enterprise Edition, que se basa en el desarrollo web modular mediante el uso de JAVA Beans. Es un framework de N capas distribuidas que se apoya en otros componentes para efectuar labores como la publicación de servicios web. Estos desarrollos requieren de un servidor de aplicaciones para ser desplegadas.

5.2.2. Componentes mas utilizados

Sumado al uso de frameworks, la comunidad ocupa componentes auxiliares para efectuar desarrollos en JAVA como el uso de ORM's y otras herramientas utilizadas para el despliegue de servicios REST. Con todo, el componente que más destaca es Hibernate, que es un componente ORM y es utilizado ampliamente para la interacción con persistencia de datos y bases de datos. Hibernate entrega una gran cantidad de herramientas y opciones para efectuar estas labores, aunque Hibernate no esta recomendado para proyectos de pequeña escala debido a su integración. Es por ello,

[6] Un toolkit es un juego de herramientas utilizadas para desarrollar bajo un marco estructurado, que provee funciones y características que facilitan las tareas de desarrollo.

que en esos casos se opta por ocupar conectores y consultas directas en la base datos, careciendo de un componente adecuado y de fácil implementación para estos fines. Por último, existen una infinidad de componentes para desarrollar una multitud de funciones y utilidades como la renderización de interfaces de usuario, motores de templates y otros elementos que pueden resultar útiles a la hora de desarrollar servicios web.

5.2.3. Desafíos en el lenguaje JAVA

A pesar de las bondades de JAVA y la extensa comunidad que lo soporta, existe una persistente relación entre JAVA y la asociación con un lenguaje de desarrollo no ágil, debido a la fuerte carencia de frameworks que permitan un desarrollo "From Scratch" en cuestión de minutos como lo proveen otros frameworks como Sailsjs, Laravel y Ruby on Rails. Hay que agregar que es sabido que JAVA no es la mejor opción para abordar proyectos pequeños ya que, por ejemplo, Hibernate -el ORM por excelencia en JAVA- no se recomienda para proyectos de pequeña escala. Si además analizamos la sintaxis y forma de escribir una funcionalidad en JAVA vs. otros lenguajes, la mayoría de los desarrolladores concuerda en que en JAVA se deben escribir más lineas de código. Como punto final, en muchos casos JAVA necesita de una cantidad desmedida de dependencias para lograr un objetivo pequeño, algo que es muy criticado en la construcción de sistemas de escala reducida que no necesitan de una gran cantidad de componentes basura para lograr un objetivo.

En conclusión, es necesario implementar herramientas que permitan el desarrollo ágil en JAVA para así dotar a este lenguaje de características que lo vuelvan una opción viable cuando sea necesario abordar rápidamente problemas y sistemas pequeños o de cualquier otro tamaño que necesiten de una fácil implementación y una sintaxis simple pero a la vez poderosa a la hora de desarrollar soluciones orientadas a servicios REST.

Capítulo 6

Estudio del problema

Cuando reflexionamos acerca de como mejorar la experiencia de los desarrolladores a la hora de escribir aplicaciones orientadas a servicios en el lenguaje JAVA, es inevitable pensar en otros frameworks ocupados por la industria, en los cuales el fabricar servicios web es una tarea casi elemental debido a la naturaleza de las aplicaciones actuales. Si bien no se duda que JAVA sea un lenguaje robusto y completo para desarrollar aplicaciones, muchas veces carece de flexibilidad para efectuar cierto tipo de elementos de software que en otros lenguajes son triviales, retrasando el proceso de fabricación componentes de software. Además, en virtud de la complejidad del proyecto de software, la implementación de estos puede verse afectada dependiendo del contenedor de aplicaciones utilizado. Para estudiar la problemática es necesario identificar las necesidades que requiere JAVA para superar las limitaciones que posee a la hora de ser utilizado en proyectos de cualquier tamaño y escala.

6.1. Caracterización de cualidades en otros frameworks

Con el fin de abordar el primer objetivo de este trabajo de titulo, analizaremos las necesidades para transformar a JAVA en un lenguaje con mejores características a la hora de abordar proyectos de construcción de software a partir de patrones comunes presentes en otros frameworks y que son prácticas comunes entre ellos. Los patrones identificados se listan a continuación:

i **Existencia de un ORM ligero:** Notamos que tanto SailsJs, Ruby on Rails, Loopback y Laravel disponen de un ORM integrado dentro de su núcleo, el cual es liviano y simple a la hora de ser utilizado. Lo que mas destacan debido a su semejanza de lenguaje con JAVA son Sailsjs y Laravel los cuales solo implementan clases con atributos, que posteriormente son mapeados para obtener objetos y datos desde la base de datos.

ii **Existencia de un CLI:** La mayoría de los frameworks presentes en otros lenguajes poseen command line interfaces que permiten una reducción en el tiempo de codificación e implementación inicial, asistiendo labores en la creación de código estructurado e inicio de nuevos proyectos. Los CLI representan la piedra angular en la agilidad con la que se efectúa de rápidamente desarrollos "From Scratch" en tan solo minutos.

iii **Servicios REST embebidos:** Todos los frameworks en los cuales se basa este trabajo de titulo, proveen de forma nativa servicios web RESTful los cuales son escritos como controladores enrutables[1]. Sobresale la simpleza de SailsJs o Laravel a la hora de declarar un controlador enrutable.

iv **Model Driven API's:** Una de las características de mayor utilidad del frameworks Loopback y SailsJS es la capacidad de generar API's por defecto, a partir de la escritura de modelos de datos. Para esto implementa por defecto las operaciones CRUD[3], así como operaciones de búsquedas especificas, disponiendo de una completa API RESTful basada en las buenas practicas descritas en el RESTful Cookbook [12].

v **Sintaxis clara y reducida:** La frameworks en los que esta basado este trabajo están construidos sobre el deseo de querer escribir menos código, proporcionando una sintaxis simple y elegante bajo la cual escribir aplicaciones, evitando de esta forma la fabricación de código espagueti, que puedan ser escalables y mantenidas fácilmente en el tiempo. Este deseo hace profundiza en evitar la

[1] Cuando nos referimos a controladores enrutables hacemos referencia a un conjunto de métodos y funciones que pueden ser expuestos como un servicio REST en un endpoint[2] con una ruta URL definida.

[3] Termino en ingles, que hace referencia a las operaciones Create (Crear), Read (Leer), Update (Actualizar) y Delete (Borrar

escritura de codigo redundante y dejar gran parte de las operaciones al núcleo del framework, el cual tiene preescritas rutinas para efectuar una gran cantidad de operaciones.

vi **Consola de administración:** Una de las características mas innovadoras de Loopback es la disponibilización de una consola de administración para manejar diversos aspectos de framework y su operación. Esta consola es un componente embebido dentro del mismo núcleo por lo que el usuario desde el momento cero, dispone de una potente herramienta que lo ayuda en el desarrollo de nuevos componentes, así como la disponibilización y configuración de nuevos servicios.

vii **Estructura de archivos clara:** La mayoría de los frameworks disponen de un arquetipo claro, mediante el cual el desarrollo de nuevas aplicaciones es algo natural y fácil de entender.

viii **JSON como estándar de información:** Siguiendo la tendencia de comunicación de información todos los frameworks implementan por defecto el manejo del formato JSON dentro de sus rutinas. El implementar JSON como estándar de comunicación permite una gran compatibilidad entre sistemas escritos en lenguajes distintos mediante el uso de servicios.

ix **Politicas como control de acceso:** Para implementar un control de acceso a los recursos de una forma escalable, los frameworks hacen uso de políticas que el framework utiliza para validar la pertinencia de acceso a un recursos. El nucleo de estos framework procesa estas políticas que son asignadas tanto a las APi's como a los recursos a los que se desea acceder, obteniendo así un mecanismo altamente fiable para efectuar control de acceso y credenciales.

x **Implementación simple:** Todos los frameworks antes descritos tienen como cualidad, su rápida implementación tanto en entornos de desarrollo como productivos, en los que destacan Ruby on Rails y SailsJS que pueden desplegar aplicaciones en minutos con solamente el uso de la linea de comandos.

6.2. Deficiencias y cualidades detectadas en JAVA

Para efectuar la construcción de un framework y abordar el segundo objetivo propuesto, es necesario hacer un análisis de las cualidades ya presentes en JAVA que seria bueno mantener, así como deficiencias que seria posible solucionar a través de la construcción del framework. Los elementos detectados se nombran a continuación:

6.2.1. Elementos positivos a mantener

i **Sintaxis de enrutamiento de Jersey:** La sintaxis de enrutamiento y declaración de endpoints en Jersey es natural y de muy fácil uso. Para esto hace uso de anotaciones, las cuales son una potente herramienta para escribir metacodigo. La notación de Jersey para definir y enrutar servicios web debe ser una piezas fundamental de un framework JAVA orientado a servicios.

6.2.2. Elementos negativos a mejorar

i **Control de archivos Properties:** Los archivos properties utilizados en la construcción de software muchas veces generan problemas en los despliegues, sobre todo cuando estos archivos aumentan en numero al incorporar nuevos componentes. Al ser archivos de texto planos, son susceptibles a rutas establecidas dentro del mismo código. Como segunda dificultad al no estar centralizadas, cada vez que se haga una modificación en las properties que involucre un despliegue sera necesario reemplazar los archivos por unos nuevos y la dificultad aumenta aun mas si se trata de un sistema multi nodo en donde habría que reemplazar el archivo en cada uno de los nodos. Seria ideal que existiera un mecanismo que permita centralizar el uso de properties y permita configurarlas directamente en una interfaz gráfica.

ii **Acceso a archivos Logs en tiempo real:** Los logs son otra de las dificultades existentes en lenguaje JAVA a la hora de abordar aplicaciones WEB y el problema se acentúa aun mas cuando se trata de sistemas multi nodo, en donde encontrar un error consistirá en revisar los logs de cada uno de los nodos. Además el acceso a los logs no se genera en tiempo real, sino que se deben

revisar los archivos de texto que los contenedores de aplicaciones generan. Seria un gran avance el poder tener dentro de una interfaz gráfica de administración la posibilidad de observar los logs que se van desarrollando en tiempo real.

iii **Recarga de configuraciones del sistema:** Cada vez que se desea cargar una nueva configuración del sistema es necesario en la mayoría de los casos hacer un reinicio del sistema. En este contexto los servidores de aplicaciones carecen de configuraciones caliente, sobre todo si el sistema usa properties para almacenar sus configuraciones. En este contexto seria útil que las configuraciones no necesitaran reiniciar la aplicación o el contenedor de aplicaciones y así tener la posibilidad de desplegar configuraciones en caliente.

iv **Ausencia de todos los método HTTP:** Los contenedores de aplicaciones no implementar métodos que son una parte fundamental de REST como lo es el caso del método PATCH. Si bien JAVA y sus Servlets soportan la petición PATCH los contenedores de aplicaciones no le dan soporte. Esta característica debe ser mejorada para poder efectuar operaciones RESTful en su totalidad.

v **Incompatibilidad entre servidores de aplicaciones:** En JAVA es muy frecuente que al programar una aplicación para un servidor de aplicaciones en particular, este no se compatible con otros servidores de aplicaciones debido al uso de rutinas propias de dicho servidor de aplicaciones. Esto genera un problema de escalabilidad y mantención en los software que son construidos para un contenedor en particular. Lo ideal en este caso es tener un framework neutro, que sin importar en que servidor de aplicaciones sea desplegado funcione de igual manera.

6.3. Solución propuesta al problema

En base a los antecedentes antes caracterizados y propuestos, se plantea la construcción de un framework tomando como referencia todas las características antes individualizadas y agregando funcionalidades que entreguen como resultado un framework de fácil uso e implementación. En este contexto se establecen dos postulados bajo los cuales estaría formada la filosofía del framework.

- El primer postulado es **"One Framework ":** Este establece que sin importar el medio que se utilice para desplegar una aplicación construida con este framework, el mismo código servirá en cada una de ellas, evitando los clásicos problemas de compatibilidad existentes entre contenedores o el uso de distintas bases de datos. Si decides utilizar un servidor de aplicaciones como JBOSS, Weblogic, Tomcat o Glassfish, o ninguno de estos utilizando las funciones embebidas del framework para levantar un servidor, no necesitas de configuraciones ni dependencias adicionales, el mismo código que fabricas sirve para todos estos. Este postulado aborda el problema descrito en el punto 5.3.5 que generaba problemas en la portabilidad de los software desarrollados.

- El segundo postulado es **"Let the machine's work to machines":** Este se basa en la obsesión por escribir menos código para obtener excelentes resultados. En esto solo es posible a través de transferir parte de la responsabilidad sobre algunos elementos a la maquina, al momento de crear nuevas aplicaciones y desplegar su funcionamiento, evitando de esta forma la redundancia de código. Para este fin el framework debe ser capaz de asistir y entregar herramientas al desarrollador desde el inicio del ciclo de vida, para de esta forma mejorar su experiencia a la hora de codificar aplicaciones. Este postulado pretende centrarse en los elementos descritos en los punto 5.1.1, 5.1.2, 5.1.3, 5.1.4 y 5.1.9 con el fin de automatizar tareas.

Basados en los postulados antes descritos, el framework debe considerar como mínimo las siguientes características para cumplir con las necesidades antes individualizadas:

i **Solo un paquete** Uno de los elementos claves que el framework debe abordar es su simpleza a la hora de implementarlo, cualidad descrita en el punto 5.1.10. Este debe proveer mecanismos para una rápida implementación en cualquier tipo de proyectos con una escasa cantidad de lineas de código. Para este fin el framework resultante debe esta distribuido en tan solo 1 paquete JAR el cual deberá tener embebido todos los elementos que necesita, sin necesidad de paquetes externos, entregando al usuario una gran cantidad disponible de funciones para efectuar las labores que necesita.

ii **Consola de administración** Una de las características mas innovadoras del

framework es el despliegue de una consola embebida desde la cual poder administrar y configurar los software construidos, cualidad descrita en el punto 5.1.6, 5.3.1, 5.3.2 y 5.3.3. Se utilizara la reflexión de clases para tener un conocimiento acerca de todos los componentes presentes en la aplicación, de forma que en todo momento el framework tenga conocimiento sobre si mismo y permita ayudar al desarrollador a través de una interfaz gráfica la configuración de los componentes. Dentro de esta consola se manejaran las properties de una forma centralizada, evitando el uso de archivos, para así poder obtener la capacidad de administrar en tiempo real las configuraciones sin necesidad de reiniciar el aplicativo. También se podrán desplegar Logs en tiempo real, obteniendo de esta forma la capacidad de monitorear en todo momento el comportamiento de una aplicación desarrollada.

iii **ORM** El tener un ORM simple y facil uso es fundamental para el desarrollo de este framework, lo cual fue descrito en el punto 5.1.1. Es por esto que se escribirá un pequeño pero potente ORM que permita efectuar operaciones con base de datos y permita utilizar los registros presentes en ella como objetos relacionales. Este ORM tiene como objetivo el estandarizar y facilitar la forma en que los usuarios interactúan con elementos de persistencia de datos. Para esto se propone que este ORM tenga soporte para múltiples bases de datos y que tenga como mínimo integradas las funciones CRUD. Además debe disponer de todo un set de herramientas para hacer manejos de tipos de datos especiales y la flexibilidad para utilizar consultas personalizadas a base de datos son perder la opción de obtener obejtos relacionales a partir de estas.

iv **API RESTful nativa:** El framework debe por defecto proveer de una API nativa con todos los métodos HTTP RESTful implementados, en la cual se puedan generar endpoints de una forma simple y efectiva usando JSON como estándar de comunicación. Las necesidades de este elemento fueron descritas en los puntos 5.1.3, 5.1.8, 5.2.1 y 5.3.4. Para esto se hará uso de la sintaxis declaratoria de Jersey y se utilizara la estructura basada en controladores usada por SailsJs para generar nuevos endpoints a través del uso de reflexión.

v **CLI:** Es importante para obtener los resultados requeridos y la consideración

del punto 5.1.2, de la fabricación de un CLI que minimice el esfuerzo a la hora de implementar un nuevo proyecto o se utilice para generar nuevas piezas de código que sigan un patrón de buenas practicas.

vi **Arquetipo:** Otro elemento clave, descrito en el punto 5.1.7, es la estructuración de un arquetipo que permita establecer una clara y natural estructura de proyectos. Este punto sera fundamental para establecer orden en la construcción de software con el framework. La idea es que el CLI, al iniciar un nuevo proyecto haga uso de este arquetipo o "Seed" para establecer la estructura de directorios.

vii **Model Driven API's:** Una de las características principales de Loopback, abordadas en el punto 5.1.4, debe estar presente en este framework para permitir la fabricar agil de aplicaciones basadas en modelos de datos y operaciones CRUD. La implementación de este patrón, permitirá obtener servicios web CRUD únicamente definiendo modelos de datos.

viii **Sintaxis simple:** A partir del uso de anotaciones y automatizando al máximo las operaciones del framework, pretendemos generar una sintaxis JAVA clara y consisa, llevando la fabricación de nuevas características a controladores altamente definidos y tipificados, así como la definición de modelos de datos simples y auto explicativos. Es objetivo principal que el usuario, solo escriba código pertinente a la característica que desea crear. A través de este punto se aborda la característica descrita en el punto 5.1.5.

ix **Multinodo:** El Framework debe tener la capacidad de funcionar en formato multinodo, tal como se describe el punto 5.3.1, de forma que cada vez que se ejecute una nueva configuración se pueda efectuar una propagación automática de los cambios.

Capítulo 7

Metodología, herramientas y ambiente de desarrollo

7.1. Metodología

Para abordar la construcción del framework debido a su naturaleza evolutiva y en base a características incrementales se decidió el uso de la metodología ágil KANBAN para la construcción del framework. Además para el desarrollo de las características se utilizo la metodología de ramas Git Flow [3].

7.1.1. Metodología Kanban

La metodología Kanban es una metodología ágil enfocada en la gestión de procesos mediante el uso de tarjetas, en un tablero que muestra de una forma visual los estados en los que se encuentran cada una de las tareas de un proyecto en ejecución. Tuvo origen en Japon, y su nombre hace referencia a u símbolo visual que se utiliza para desencadenar una acción.
El tablero que se muestra en la figura 7.1 tiene origen el flujo de las tareas dispuestas en forma de tarjeta y puede ser dividido en las distintas fases del desarrollo de características de un software. Para este fin se disponen de distintas fases donde las principales son Backlog, To-Do, In Progress, In revisión y Done. Kanban es de fácil uso y permite una visualización natural del estado del proyecto. Además muchas

plataformas de repositorio de código como Github, Gitlab y Bitbucket la tienen implementada por defecto, lo que ayuda mucho en la gestión de las caracteristicas que se estan desarrollando.

7.1.2. Metodología Gitflow

La metodología Gitflow[4] es una metodología creada con el fin de efectuar un trabajo estructurado y planificado con repositorios de código, con el fin de mantener una norma y un flujo estandarizado en los procesos de fabricación de nuevas características. Para este fin define ramas que permanecen a lo largo del ciclo de vida de un software y otras que son creadas de forma dinámica. En este contexto existen 3 ramas que nunca varían, las cuales son master y develop. La rama master es aquella que esta lista para ser puesta en producción, mientras que la rama develop es la que va incorporando las ultimas características y será pronto integrada dentro de la rama master. Además de las dos ramas principales, se implementan una serie de ramas para ayudar al desarrollo en paralelo entre los miembros del equipo. A diferencia de las ramas principales estas tienen un ciclo de vida muy corto y son creadas y destruidas a medida que se necesitan durante el ciclo de vida. Estos no son ramas como tales, sino tres tipos de ramas utilizadas para diferentes usos los cuales son.

- **Rama Feature:** La cual se utiliza para el desarrollo de nuevas funcionalidades. Estas ramas son creadas y se utilizan solo durante el ciclo de vida de la funcionalidad que luego es unida a otra rama, dejando de existir. Son unidas dentro de la rama develop, una vez que la característica ha sido desarrollada.

- **Rama Release:** Es una rama que tiene como único propósito unir la rama develop con la rama master, y estabilizar el código antes de formar el merge final.

- **Rama Hotfix:** Es una rama que se utiliza para efectuar correcciones a características desplegadas en la rama master, de forma de actuar inmediatamente frente a cualquier error detectado. Son eliminadas una vez que la revisión fue resuelta y son unidas directamente con la rama master.

Figura 7.1: Figura que muestra un tablero Kanban utilizado durante la ejecuión del proyecto.

7.2. Herramientas

Se utilizaron diversas herramientas para la fabricación del framework las cuales serán mencionadas a continuación:

- **Maven:** Se utilizó maven como herramienta para gestionar los paquetes requeridos por el framework. Tambien se utilizo para distribuir y generar el paquete JAR del framework.

- **GIT:** Se utilizo GIT como herramienta de control de versiones y código, con el objetivo de tener un adecuado y confiable control de ramas de desarrollo.

- **Gitub:** Se utilizo Github como repositorio base para el framework a desarrollar, pero además fue utilizado para gestionar a través de sus tableros de proyectos la metodología KANBAN y el incremento de nuevas características. La ubicación del repositorio oficial del framework es https://github.com/acalvoa/earlgrey.

- **Eclipse Neon:** Se utilizo eclipse como IDE base para la construcción del framework debido a la facilidad que posee para integrarse con servidores de aplicaciones.

- **Visual Studio Code:** Se utilizo para desarrollar la mayoría de las interfaces de usuario y todo los elementos de la consola escritos en Angular. Visual Studio Code además fue utilizado para trabajar con las ramas del repositorio y el commit de todos los cambios del proyecto.

- **Angular Seed 2:**[6] Se utilizo para desarrollar la consola de administración bajo un marco altamente estructurado y extensible, con capacidades de empaquetamiento y reducción de código.

- **Angular 2 Seed CLI:**[10] Se utilizo para acelerar las tareas de fabricación de código en la consola de administración, tanto para crear el proyecto, como para escribir nuevas características de una forma simple y altamente definida en estructuras de directorios.

7.3. Ambiente de desarrollo

Para desarrollar la solución propuesta se utilizo el ambiente de desarrollo detallado a continuación:

- **JAVA JDK 7:** Como se menciono previamente la solución fue construida en JAVA 7, por lo tanto el soporte proveído es desde esta versión, debido a que JAVA 7 incorporo muchas características útiles para fabricar el Framework, siendo una de estas la no declaración de filtros.

- **JBOSS 6.4+ EAP:** Se utilizo como contenedor de aplicaciones la versión enterprise de JBOSS, en gran parte debido a que la aplicación de prueba con la que se desea probar el framework utiliza JBOSS 6.4+.

- **Angular 4:** Se utilizo Angular como framework frontend para construir las interfaces de usuario y la consola de administración de Earlgrey. Para esto se hizo uso de Angular Seed 2, como esqueleto para construir la aplicación.

- **Node.js 7:** Se utilizo Node.js para fabricar los CLI necesarios para el framework Earlgrey.

- **Docker:** Se utilizo docker para efectuar las pruebas en multiples bases de datos y las tareas multi nodo, levantando contenedores con las instancias requeridas.

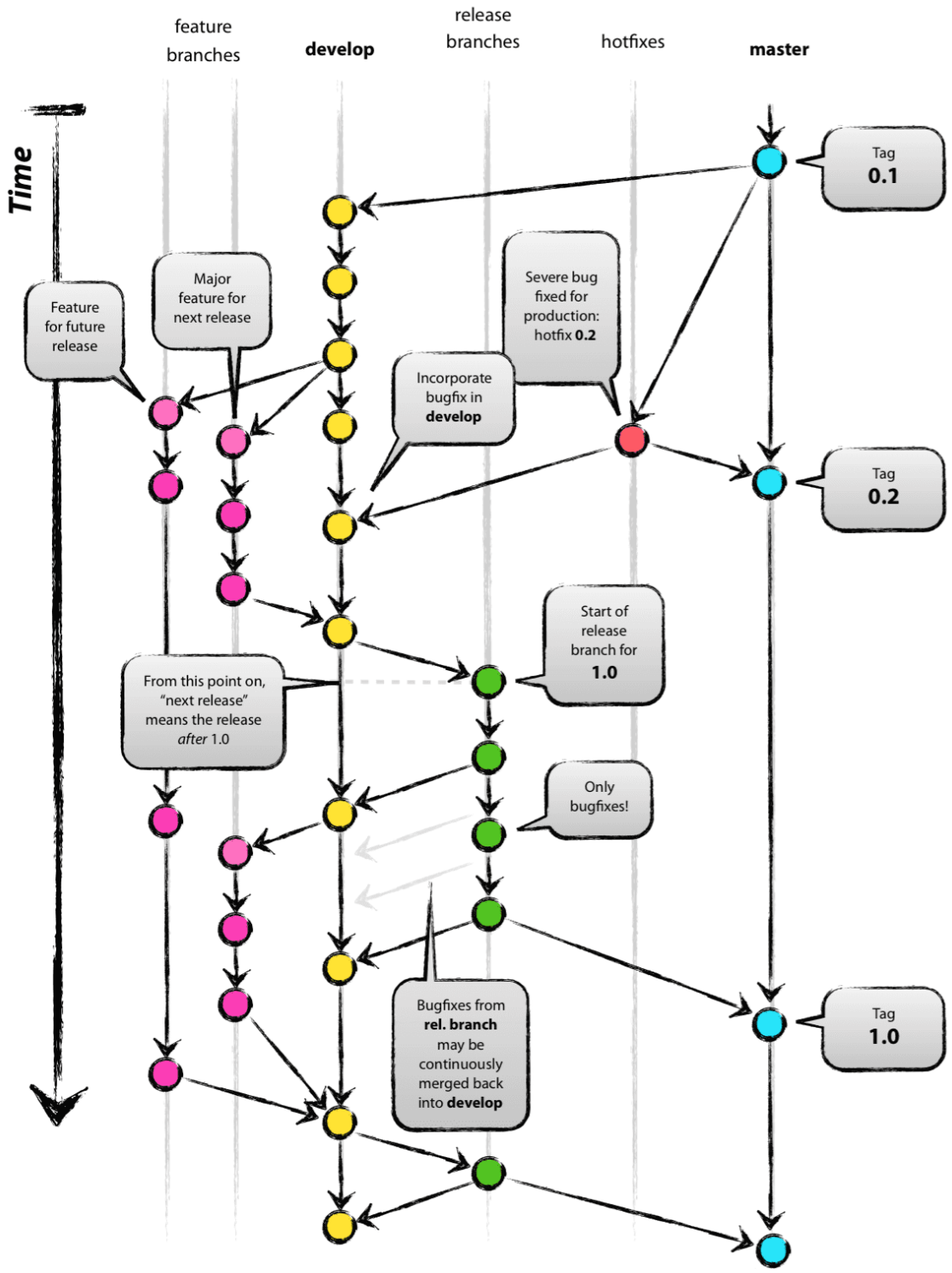


Figura 7.2: Imagen que muestra el flujo de trabajo con gitflow.

Capítulo 8

Desarrollo

8.1. Análisis de la Solución

Para abordar el desarrollo de la solución, primero fue necesario establecer la arquitectura de software bajo la cual opera la solución. Si bien el framework conserva patrones de la arquitectura MVC utilizando un esquema de modelos y controladores, su arquitectura principal es la orientada a servicios, ya que debe disponibilizar de servicios a todo tipo de aplicaciones de una manera rápida y simple. Aún así, el framework fue construido haciendo un híbrido entre las arquitecturas mencionadas ya que, a pesar de todo, se decidió utilizar un patrón de tres capas para escribir nuevas funcionalidades y aplicaciones.

8.2. Diseño de la Solución

La tarea de diseñar una solución que cumpliera con los objetivos propuestos tuvo como principal desafío el definir como comenzar el desarrollo de la misma. En ese sentido, para abordar el problema fue necesario, en un inicio, plantear los componentes involucrados dentro de la arquitectura del framework. Para este fin, se ha planteado que el mejor camino es que el framework funcione como una capa de abstracción de la aplicación desarrollada que toma control del código y las clases que esta genera. Además, en el caso de usar un servidor de aplicaciones, el framework debiera comportarse como un Middleware que intercepte las peticiones que provienen desde el

servidor de aplicaciones.

La base de funcionamiento del framework será el uso de reflexión como elemento y herramienta clave para obtener conocimiento acerca de la aplicación y sus clases, y hacer de estas parte de él en cada instancia que se ejecute. Al utilizar reflexión podremos utilizar clases escritas fuera del framework como si fueran parte de él, transformando a la aplicación en una sola unidad que interactúa de forma coordinada. También, gracias a esta propiedad, podemos mapear toda la aplicación para así ejecutar labores administrativas e, incluso a futuro, la posibilidad de incorporar o modificar clases desde dentro de la consola de administración. Junto con la identificación de los componentes de la aplicación, fue necesario definir las herramientas que serían un complemento y ayuda en el trabajo de codificación con el framework Earlgrey.

8.3. Arquitectura y Componentes

Como parte del diseño de la solución fue necesario definir la arquitectura y los componentes planteados en la figura 8.1, de la cual se explicara cada uno de sus componentes y su planteamiento. En este contexto el framework Earlgrey se separa en dos universos de componentes claramente definidos y diferenciados.

8.3.1. Núcleo de Earlgrey

- **Kernel**: Es el núcleo de funcionamiento de Earlgrey, mediante el cual se instancia y permite ejecutar las operaciones de este. El kernel es el encargado de comunicar los objetos y mantenerlos en memoria, así como se recibir configuraciones de alto nivel que provienen directamente desde el código de la aplicación. El kernel para funcionar necesita el contexto de la aplicación que se esta desarrollado, debido a que de otra forma se generan ClassPath distintos por el funcionamiento interno de java y esto genera que el framework no pueda cargar la aplicación dentro de su base de conocimiento, ya que es necesario el conocimiento de la existencia de las clases en la aplicación para poder ejecutar reflexión sobre ellas.

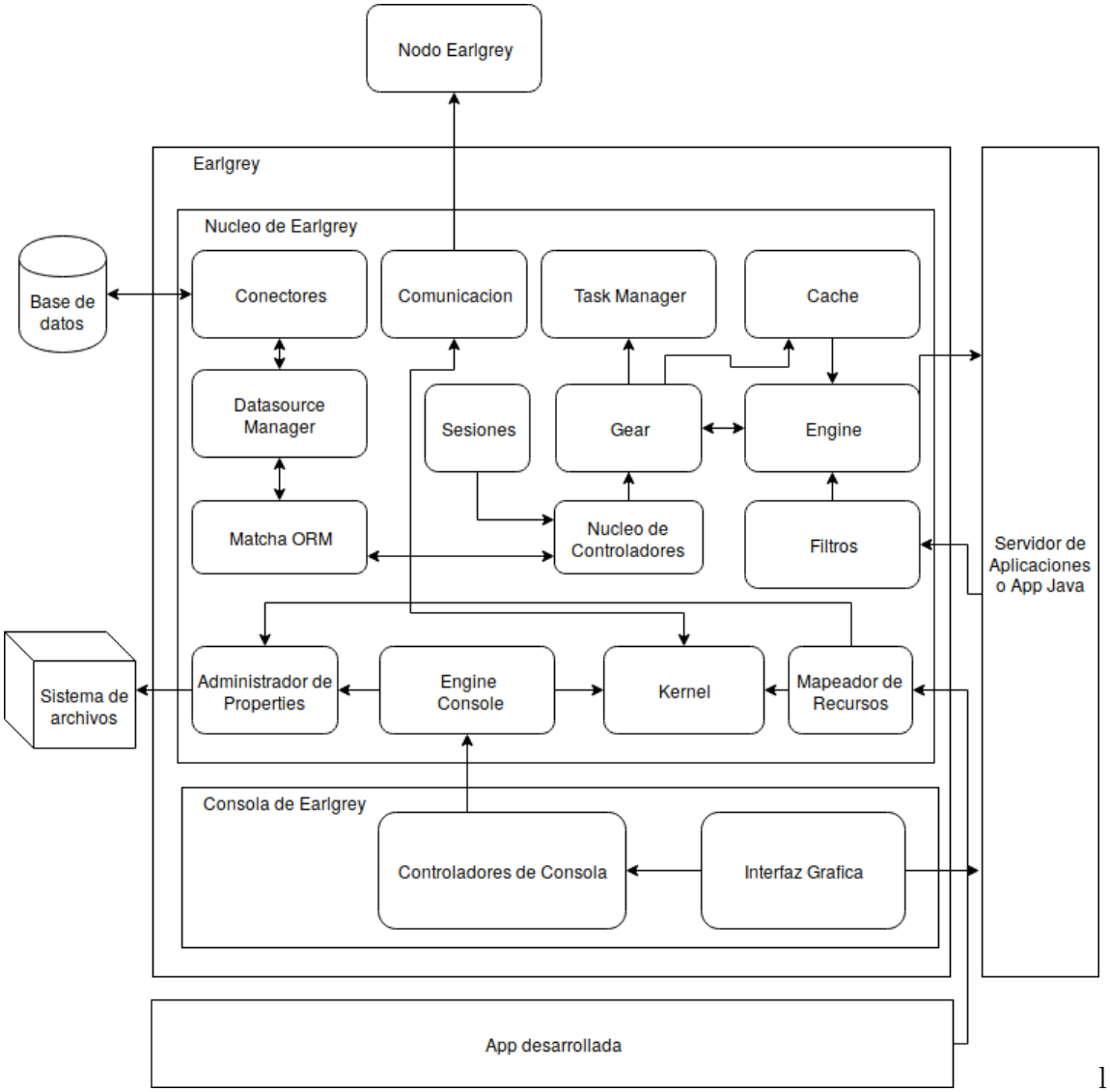


Figura 8.1: Figura que muestra la arquitectura de los componentes y su interacción

- **Mapeador de Recursos:** Es el componente encargado de efectuar reflexión sobre los archivos de la aplicación desarrollada y sobre todo el framework. Es el encargado además de leer la sintaxis de anotaciones proveídas por el framework para ejecutar diversas operaciones para desplegar y configurar servicios, generar nuevas properties y configuraciones, así como para generar modelos de datos.

- **Filtros:** Los filtros son el recurso utilizado para interceptar los flujos HTTP del servidor de aplicaciones y redirigirlos al núcleo de Earlgrey. Desde el momento que Earlgrey inicializa este toma el control de la aplicación dejándole al servidor HTTP o contenedor de aplicaciones solamente la gestión de respuesta generadas por el framework o las tareas de servir archivos. La figura 8.2 explica la forma en funciona este mecanismo.

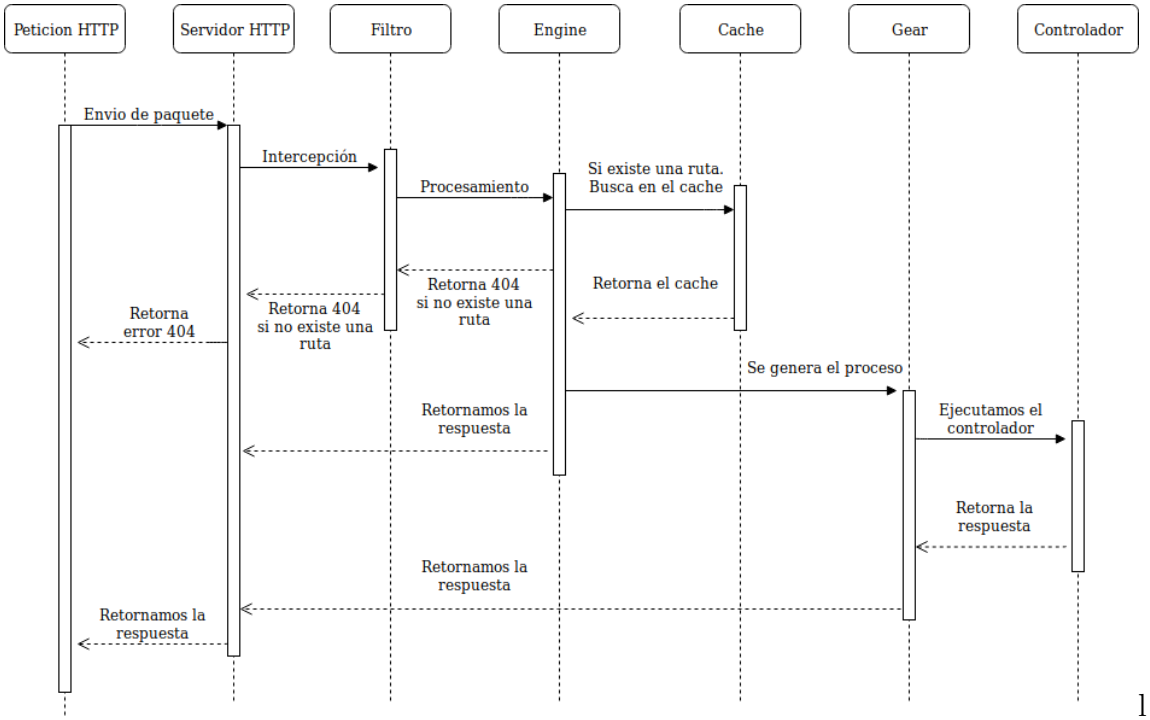


Figura 8.2: Establece el diagrama se secuencia para una llamada HTTP en Earlgrey

- **Engine:** Es el componente que inicia el ciclo de enrutamiento y generación de procesos dentro de Earlgrey. Engine es quien procesa las peticiones HTTP y las

asigna a los controladores que se han definido para responder a dichas rutas. Además es el encargado de procesar respuestas en que han sido previamente guardadas en cache, para evitar su procesamiento.

- **Cache:** Es el núcleo que administra el Cache de la aplicación. En este contexto Earlgrey dispone de dos tipos de cache integrados dentro de su sintaxis los cuales son el cache general, que es utilizado en todas las peticiones, y el cache de usuario que solo es usado por cada sesión de usuario recurrente.

- **Sesiones:** El manejador de sesiones se encarga de gestionar, almacenar y establecer los tiempos de vida de las sesiones de los usuarios que utilizan los endpoints de earlgrey. Cada vez que un usuario hace uso de la plataforma se le asigna un identificador y se guarda su objeto de sesión, las cual es posible rescatar a futuro. El manejador de sesiones soporta JWT de forma nativa por lo que es posible usar esta tecnología sin grandes complicaciones. Las sesiones por si solas tiene establecido un tiempo de vida máximo, que una vez finalizado acuáticamente ejecutan un proceso que las quita de memoria.

- **Gear:** Los Gears son los elementos encargado de ejecutar cada llamada designada por el Engine. Los Gear se encargan de obtener parámetros, atributos y ejecutar procesos de reflexión sobre un controlador enrutado para ejecutarlo. Los Gear contienen toda la información acerca de la petición y están contenidos dentro de un proceso que al ser sacado de memoria se encarga de eliminar todos los elementos existentes sobre si mismo.

- **Task Manager:** Es el administrador de procesos de Earlgrey. Se encarga de encapsular cada uno de los procesos en unidades de objeto, para de esta forma tener el control de estos y gestionarlo. El Task Manager se encarga de limpiar procesos de memoria una vez que estos han finalizado, asegurándose que todos ejecuten y limpien las variables y conectores que han utilizado. En caso de ser necesario, el task manager es capaz de quitar un proceso de memoria y eliminarlo.

- **Núcleo de Controladores:** El Núcleo de controladores es el encargado de efectuar todas las operaciones para disponibilizar un controlador para su uso.

El núcleo de controladores se encarga de asignar los controladores cada vez que estos son solicitados por un Gear y asignarle todos los elementos que este necesite. Además este se encarga de gestionar las Politicas (Policies), las cuales determinan los controles de acceso a los distintos controladores existentes.

- **Cronjobs:** El administrador de cronjobs es el núcleo central que administra las tareas programadas que son posibles de escribir con el framework.

- **Comunicación:** El objeto de comunicación es el encargado de interactuar con otros nodos que utilizan earlgrey con el objeto de sincronizar información entre estos dando por ejemplo las properties y otros elementos volátiles. Por defecto, los nodos de comunicación abren un socket en el puerto 11750 para efectuar labores de comunicación con otros nodos, el cual puede ser modificado al momento de instanciar Earlgrey. Todos los nodos se pueden configurar desde la consola de administración.

- **Datasource Manager:** El Datasource Manager es el encargado de gestionar todas las conexiones con orígenes de datos. Se encarga de inicializar todos los conectores y mantenerlos funcionando de dentro de los marcos de configuración. Es el encargado de suministrar conectores al ORM cuando este los requiera y cerrarlos cada vez que dejen de ser ocupados o por orden del Task Manager. Cada vez que Earlgrey reconoce la necesidad de otro framework declarado en la sintaxis de sus modelos, automáticamente lo crea para ser definido en la consola de configuración dentro de los ambientes de properties que maneja internamente. Earlgrey soporta Datasources definidos de forma manual, como datasources definidos por un contenedor de aplicaciones.

- **Conectores:** Los conectores son clases escritas para interactuar con las bases de datos que soporta el framework Earlgrey. Existe una por cada origen de datos soportado por Earlgrey. Para este trabajo Earlgrey soporta 4 tipos de bases de datos. Oracle, Postgres, Mysql y Sails Disk que es una estructura fabricada con el objetivo de manejar información en disco, concepto que lo hace altamente eficaz para prototipos.

- **Matcha ORM:** Matcha es el ORM creado para Earlgrey, diseñado con una

sintaxis simple y elegante que le permiten manejar operaciones CRUD en base de datos. Las operaciones soportadas por Matcha son find(), findOne(), update(), delete(), create(), match(), count() y specialQuery(). Además Matcha es capaz de gestionar tipos de datos personalizados que pueden efectuar operaciones distintas a su comportamiento tradicional, mapeando los elementos de forma que todo es transparente al usuario. Además Match tiene flexibilidad para soporta el uso de sentencias SQL directas en caso de ser necesario la cuales son automáticamente mapeadas, aunque dicha característica no se recomienda utilizar para mantener el concepto principal de Ealrgrey, One Framework. Matcha también soporta Multitenant para gestionar aplicaciones que requieren de múltiples clientes.

- **Administrador de Properties:** A través de procesos de reflexión, todas las properties, datasources y configuraciones definidas con la sintaxis de Earlgrey son reconocidas por el Administrador de properties, que se encargar de disponibilizarlas de forma automática en la consola de configuración de Earlgrey, para ser configuradas y persistidas en disco duro en formato JSON. El administrador de properties tiene soporte para manejar múltiples ambientes de despliegue, cada uno con su set de propiedades, los cuales pueden ser cambiados en tiempo de ejecución sin necesidad de reiniciar la aplicación. Si Earlgrey se encuentra configurado en esquema de enjambre con otros nodos conectados, automáticamente se encarga de propagar los cambios efectuados en las properties, datasources, configuraciones o ambientes de despliegue.

- **Engine Console:** El Engine Console es similar en funciones al Engine tradicional, pero se encarga de gestionar únicamente los controladores utilizados por la consola y que son parte del paquete Ealrgrey. Al igual que Engine, hace uso de los Gears para ejecutar las labores, aunque su forma de funcionamiento utiliza por defecto políticas suministradas solo para el perfil de administrador.

8.3.2. Consola de Administración

- **Controladores de Consola**: Los controladores de consola son una serie endpoints diseñados para responder a las configuraciones y peticiones efectuados

por la consola de configuración y su interfaz de usuario. Efectúan una serie de operaciones que permiten administrar todos los aspectos de del framework.

- **Interfaz Gráfica**: La interfaz gráfica representa la interfaz de usuario desarrollada para operar con la consola de Earlgrey. Esta interfaz fue desarrollada en Angular 4. Tanto la interfaz de usuario como la consola, están embebidas en el paquete Earlgrey, por lo que esta disponible sin efectuar ningún tipo de configuración extra.

8.4. Herramientas complementarias

8.4.1. CLI

Como parte de los objetivos principales fue necesario definir conceptualmente una herramienta que asistiera a los desarrolladores en la creación de nuevos proyectos y permitiera a través del uso de generadores, de la fabricación de plantillas de componentes del framework Earlgrey. Para este fin, se contemplo la posibilidad de dar soporte al IDE eclipse, que es el IDE JAVA por excelencia. Además se contemplo inicialmente darle soporte al Framework Angular para integrar proyectos dentro de su plataforma, mediante la fabricación de solo un paquete final.

8.4.2. Framework Seed

Fue necesario establecer un proyecto básico, con una estructura de directorios definida, que pudiera ser extensible y escalable a la construcción de cualquier proyecto que desee usar como base Earlgrey. Este proyecto se baso en la idea de un arquetipo que define la estructura básica de un proyecto Earlgrey ideal, que tuviera una lectura natural para los desarrolladores, evitando el exceso de empaquetamiento. El Seed además muestra toda una serie de buenas practicas de codificación que hacen posible el desarrollo de un proyecto claro y libre de redundancias. Earlgrey Seed da una muestra básica de sus funcionalidades que pueden ser utilizadas por el usuario para elaborar sus proyecto.

8.5. Implementación de Earlgrey

Para implementar la solución se separo el proyecto en 4 fases incrementales que fueron abordadas con la metodología Kanban en la plataforma github. Cada una de estas fases efectúo un release de la solución con su versión correspondiente. Es importante destacar que al momento de terminar la fase 1, el framework dispuso de una versión mínima funcional que era capaz de implementar un numero considerable de funcionalidades. Es importante señalar que durante la construcción de Earlgrey se comenzó simultáneamente la construcción de un sistema que serviría para probar y analizar continuamente requerimientos evolutivos que se dieran durante su uso. Este sistema sirvió para validar los componentes que eran construidos en el framework, dando un feedback inmediato al proceso de construcción. Cada una de las fases se encuentran presentes en el repositorio libre del framework[1]

8.5.1. Fase 1. Versión 0.1:

La fase 1 del proyecto tuvo como meta alcanzar la construcción de un arquetipo que implementara un gran numero de funcionalidades del framework con una implementación parcial de la sintaxis que se baso en el uso de anotaciones y el uso de reflexión sobre sus componentes. Para este fin fue desarrollado como un proyecto web JAVA dinámico y baso su funcionamiento en el uso de Servlets que eran instanciados directamente desde la aplicación a construir. La aplicación que se debía codificar no estaba separada de todos los componentes de los que disponía Earlgrey y era tratado como 1 solo proyecto que solo podía ser desplegado en un contenedor de aplicaciones. Se diseño e implemento una versión primitiva y reducida de Matcha ORM que permitía efectuar operaciones básicas sobre la base de datos como efectuar operaciones select() o create(). A la vez, se implemento una primera versión de la consola que permitía ver logs en tiempo real y administrar properties con persistencia.

[1] El repositorio del proyecto se encuentra albergado en github en la siguiente url; https://github.com/acalvoa/earlgrey

8.5.2. Fase 2. Versión 0.2:

La fase 2 del proyecto tuvo como principal propósito el lograr el empaquetamiento del framework en un solo archivo JAR. Para este fin, el proyecto debió convertirse en un proyecto maven que tuviera la capacidad de generar un solo archivo al finalizar su proceso de compilación, integrando librerías, componentes JAVA y la consola de administración. Durante esta fase, se produjeron las mayores dificultades del proyecto debido a las propiedades del lenguaje JAVA, lo que durante un largo tiempo impidieron cumplir el hito, lo que provoco la investigación de multiples caminos para aproximarse a la solución. Para lograr el objetivo se tuvo que abandonar muchas de las ideas que se consolidaron en la primera versión, como el uso de servlets auto declarados, que no eran desplegados al encontrarse dentro del paquete JAR, ya que se generaban Classpath y contextos distintos entre el paquete JAR y la aplicación desarrollada. El elemento clave para encontrar la solución fue el uso de filtros que a pesar de encontrarse en el paquete podían ser usados por el framework y el servidor HTTP. Desde estos filtros todas las petición HTTP fueron desviadas al núcleo del framework que se encargo de procesar las peticiones, delegando al servidor http o contenedor de aplicaciones solo la tarea de servir archivos o desplegar mensajes de error. Además a partir de uso de filtros se logro superar una de las principales deficiencias del los contenedores de aplicaciones, el uso del verbo PATCH en las llamadas HTTP. Al superar, el mayor de los problemas, se pudo compilar un solo paquete que era llamado desde la aplicación a desarrollar a través del uso de un ServletListener en los servidores de aplicaciones. Junto con esto se agregaron mejoras incrementales al ORM como el método match() y el desarrollo de las capacidades de cache y gestión de sesiones. Ademas se actualizo el núcleo de properties agregando 3 tipos distintos de properties, los properties simples, las option properties para efectuar decisiones de una lista de opciones y los setproperties para guardar objetos completos de configuración.

8.5.3. Fase 3. Versión 0.3:

La fase 3 del proyecto tuvo como objetivo el abordar una de las principales focos del framework, crear una sintaxis simple de usar y disminuir la cantidad de lineas

de código para obtener los mismo resultados. En ese contexto se rediseño el núcleo de properties y se agregaron los datasources como parte de las propiedades de un entorno. Las configuraciones del sistema también pasaron a ser elementos independientes. El núcleo de enrutamiento también fue rediseñado y se opto adoptar parte de la notación del framework Jersey para definir rutas y servicios, así como la adopción total de las buenas practicas del RESTful Cookbook[12]. La manera de escribir controladores y acciones de controlador también cambio permitiendo permitiendo el uso de controladores sin ruta. El componente de cache fue reescrito añadiendo la posibilidad de generar cache global y por usuario. Se implemento la posibilidad de usar los métodos HTTP OPTIONS y HEAD, y se agrego el uso de CORS para gestionar peticiones Cross-Domain. La consola de configuración cambio muchas de sus estructuras, a nivel de interfaz de usuario y agrego el login de administrador para evitar el acceso indeseado a las configuraciones del framework. Se comenzó a escribir la documentación del framework y las herramientas complementarias, además se añadieron 3 contribuidores al proyecto que comenzaron a utilizar el framework en 2 proyectos distintos al caso de pruebas que se utilizo inicialmente para la construcción del framework.

8.5.4. Fase 4. Versión 0.4:

La fase 4 de construcción del proyecto fue afinar el framework para obtener una versión release LTS[2]. Se actualizo la consola a su versión final, la cual permite todos los tipos de configuración y visualización. Además se agregó la comunicación entre nodo que permite compartir configuraciones y elementos volátiles entre estos. A nivel de ORM se implementaron todas las operaciones faltantes, y las propiedades de Multi-Tenancy[3]. Además se implemento, una propiedad que permite el despliegue de la aplicación solo en formato de API, lo que permite evitar el context path de la api.

[2]Sigla que hace referencia a Long Term Support, o versión de largo soporte. Este tipo de versiones reciben actualizaciones para corregir bugs o elementos detectados pero no reciben cambios que alteren el funcionamiento actual de la versión

[3]Hace referencia al concepto de Muti cliente y es una propiedad utilizada cuando los clientes de una aplicación hacen uso de distintas bases de datos.

Se implementaron los Cronjobs[4] y Model Blueprints[5]. Otra de las propiedades que los contribuidores pidieron fue la posibilidad de tener la característica llamada Earlgrey Boot, que permite desplegar la aplicación solo a través del uso de un proyecto JAVA convencional. Además se solicito como característica extra la posibilidad de escribir documentación desplegable en un manual de API, característica similar a la efectuada por API DOC, pero siendo una caracteristica embeida del framework.

8.6. Implementación de herramientas complementarias

La implementación de las herramientas complementarias comenzó su ejecución en una fase tardía del proyecto general, efectuando las primeras aproximaciones conceptuales a comienzos de las fase 3 del proyecto. La codificación de estas comenzó a finales de la fase 3 del proyecto cuando las estructuras y sintaxis del framework ya adquirieron una estabilidad definitiva. La implementación de las herramientas se detalla a continuación.

8.6.1. Implementación de Command Line Interface

La implementación del Command Line Interface fue efectuada desde la fase 3 del proyecto, quedando su codificación a cargo de Sebastian Gonzalez Villena bajo el diseño y aprobación del equipo. El Command Line Interfaz denominado "Infusion" se baso en el Earlgrey Seed para establecer los comandos y estructuras de codificación capaces de generar. El CLI fue desarrollado en Nodejs y baso su implementación en el proyecto Angular Seed 2 CLI desarrollado previamente por nosotros. Tuvo como principal objetivo la capacidad de generar estructuras a través de la linea de comandos, que siguieran las nomas de buenas practicas de codificación en Earlgrey y por sobre todo ahorrar tiempo en la inicialización y construcción de una solución de software usando el framework earlgrey. Añade múltiples funcionalidades como

[4]Los Cronjobs son tareas programadas que permiten al framework ejecutar operaciones durante su ejecución en horas determinadas

[5]Es una propiedad que permite desplegar API's RESTful con operaciones CRUD solamente a partir de la declaración de un modelo de datos

soporte para eclipse, proyectos Angular y mavenización nativa y toda una gama de generadores para ahorrar tiempo en la generación de código fuente. Actualmente se distribuye bajo la plataforma NPM de Nodejs.

8.7. Implementación de Earlgrey Seed

Earlgrey Seed tuvo como base de implementación un pequeño proyecto de pruebas que se utilizo para el desarrollo de pequeñas características en el framework a nivel de núcleo y consola. Este proyecto fue transformado y adaptado para fabricar el arquetipo ideal de Earlgrey. Una de sus mayores dificultades fue que su implementación cambio muchas veces a causa de los cambios que se introducían frecuentemente en el framework, causando muchas veces una incompatibilidad de su estructuras. Finalmente con el finalización de la versión 3 el arquetipo alcanzo un grado de madurez que le permitían abordar proyectos maven sin ninguna dificultad, lo que permitió su documentación.

8.8. Pruebas funcionales efectuadas

Las pruebas efectuadas para validar el cumplimiento de los objetivos del framework, fueron las siguientes:

- **Pruebas de funcionalidad en distintos contenedores de aplicaciones:** Para probar el principio de **One Framework** se probo él despliegue del framework en los contenedores de aplicaciones JBOSS, Weblogic, Glassfish y Tomcat.
- **Pruebas de funcionalidad en aplicaciones JAVA:** Se probo la instanciación del framework en un proyecto java convencional para probar que es posible desplegar Earlgrey sin necesidad de un contenedor de aplicaciones.
- **Pruebas de operaciones de ORM Matcha:** Se efectuaron pruebas sobre todas las operaciones disponibles por el ORM, generando endpoints que ejecutaban las operaciones en conexión con una base de datos de prueba.

- **Pruebas de funcionalidad en distintas bases de datos:** Se probo la capacidad para operar con múltiples bases de datos a traves del uso del ORM. Para este fin se probo la base de datos Oracle, Mysql y Postgres.
- **Pruebas de funcionalidad Multi Nodo:** Se dispuso Earlgrey en una configuración de enjambre y se hizo operar en este formato, verificando la sincronización de sus elementos.
- **Pruebas de operaciones RESTful:** Se efectuaron todas las operaciones REST sobre los endpoints para verificar su correcta operación.

8.9. Caso de implementación real

Para verificar el funcionamiento del framework se planteo la construcción de un proyecto real con el framework. Para este fin, la Controlaría General de la República se dispuso la construcción de un sistema de complejidad intermedia, que fuera capaz de desplegar la información generada por los sistemas de personal, auditoría y contabilidad de la nación en una plataforma georeferenciada, actuando de esta forma con el concepto de un observatorio de información publica de distintos aspecto sobre los que Contraloria tiene jurisdicción. Para este fin, se decidió el uso de una arquitectura orientada a servicios como plataforma de backend, un desarrollo en el framework Angular a nivel de frontend, desplegado sobre una base de datos Oracle y un servidor de aplicaciones JBOSS EAP 6.4. Para este fin, se considero perfecto el uso de Earlgrey para implementar el proyecto, ya que permitía finalmente la obtención de un paquete WAR único desplegable con gran facilidad en el servidor de aplicaciones en una configuración de dos nodos. Los requerimientos funcionales del sistema denominado "Observatorio Sector Municipal" fueron definidos en el documento de requerimientos del Observatorio Municipal, vease anexos.

Capítulo 9

Resultados y Discusión

Al finalizar el desarrollo del framework en su versión estable y utilizarlo para implementar en un proyecto real puesto en producción se pudieron observar los siguientes resultados.

- Respecto a la fabricación de un framework que lograra codificar de una forma ágil aplicaciones orientadas a servicios, se logro implementar un framework robusto que permitió cumplir con los objetivos propuestos, implementando y disponibilizando una API embebida que permite proveer a clientes, de endpoints de servicios a partir de controladores basados en rutas definidas y políticas de acceso; Estas características permiten aumentar la productividad a la hora de desarrollar aplicaciones orientadas a servicios. En ese contexto, respecto al hito de tener una sintaxis que disminuyera la cantidad de código escrito, se optimizo y automatizaron muchas operaciones con el objetivo de asistir el desarrollo de software, sin embargo la sintaxis de un controlador simple se asemeja mucho a la que se logra trabajando con frameworks como Spring boot y Jersey. Sin embargo tiene sintaxis mucho mas reducida y simple a la hora de trabajar con operaciones de bases de datos, uso de propiedades de sistema, properties y politicas.
  En ese caso disponemos de algunos ejemplo que se muestran en la figura 8.1 y 8.2. En estas figuras observamos que la sintaxis es similar en cantidad de lineas de código, pero la gran diferencia radica en que en earlgrey ese controlador ha sido mapeado por la la consola de administración y puede ser manejado

```
//CONTROLADOR DE PRUEBA PARA EFECTUAR DESARROLLO EN EARLGREY.

@ControllerAction(
description = "Acción del controlador utilizado para efectuar el test",
name = "test_data",
version = 1)
@Route("/")
@GET
public static void test(HttpRequest req, HttpResponse res){
     Date date = new Date();
     DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG);
     String formattedDate = dateFormat.format(date);
     res.ok(formattedDate);
     return;
}
```

Figura 9.1: Código de pruebas para analizar la sintaxis de Earlgrey en un servicio web de pruebas

desde su interior. Además de disponer de properties configurables en el panel de administración y otras propiedades como políticas y opciones configurables. Otra versatilidad que fue posible obtener con Earlgrey es su simpleza de implementación, que permite estar trabajando con el framework desde cero en tan solo unos minutos.

- Otro elemento desarrollado es la consola de administración, lo cual es una de las mayores innovaciones del framework. La consola de administración de Earlgrey representa una de las herramientas mas poderosas del framework y que permite disminuir las tareas de desarrollo y configuración de una aplicación desarrolladas con el framework. Con el resultado obtenido logramos el despliegue de logs en tiempo real así como la configuración en caliente de properties, datasources y configuraciones generales de la aplicación. Además la configuración multi nodo implementada permite una fácil sincronización entre enjambres de servidores lo que lo hace perfecto para el despliegue empresarial de alta disponibilidad.

```
//CONTROLADOR DE PRUEBA PARA EFECTUAR DESARROLLO EN JERSEY.

@Path("/hello")
public class test {
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String test() {
        Date date = new Date();
         DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG);
         String formattedDate = dateFormat.format(date);
         return formattedDate;
    }
}
```

Figura 9.2: Código de pruebas para analizar la sintaxis de Jersey en un servicio web de pruebas

Otro punto a considerar es que la consola fue de gran ayuda durante el desarrollo y la configuración del proyecto "Observatorio Sector Municipal", debido a que solo fue necesario crear un paquete para todos los entornos de despliegue, en donde se configuraban y era posible elegir los entornos de despliegue a utilizar en las properties. Además la característica fue útil durante el desarrollo debido a que permitía observar los logs en tiempo real que ocurrían en los servidores. La figura 8.3 muestra la consola de administración que dispone Earlgrey de forma nativa en su implementación.

- Para finalizar, la implementación del sistema "Observatorio Sector Municipal" que se muestra en la figura 8.4, fue clave para entender la utilidad las funcionalidades construidas con el framework. La construcción de este sistema revelo muchas deficiencias del framework que rápidamente eran corregidas para brindar el soporte que necesitaba. El construir un proyecto real con el framework desarrollado doto de robustez a la solución y validación de cada una de sus componentes. La incorporación de colaboradores al proyecto y el uso por parte de estos para desarrollar 3 proyectos adicionales con el framework, le dio valida-

ción, correcciones y soporte por parte de los usuarios que lo harán evolucionar hacia un mejor framework en versiones futuras.
Además se inicio la construcción de un nuevo proyecto en Controlaría General de la República de Chile que hace uso del framework Earlgrey, que consiste en la construcción de la primera API RESTful de propósito general acerca de información y datos comunes en los sistemas administrados por esta.

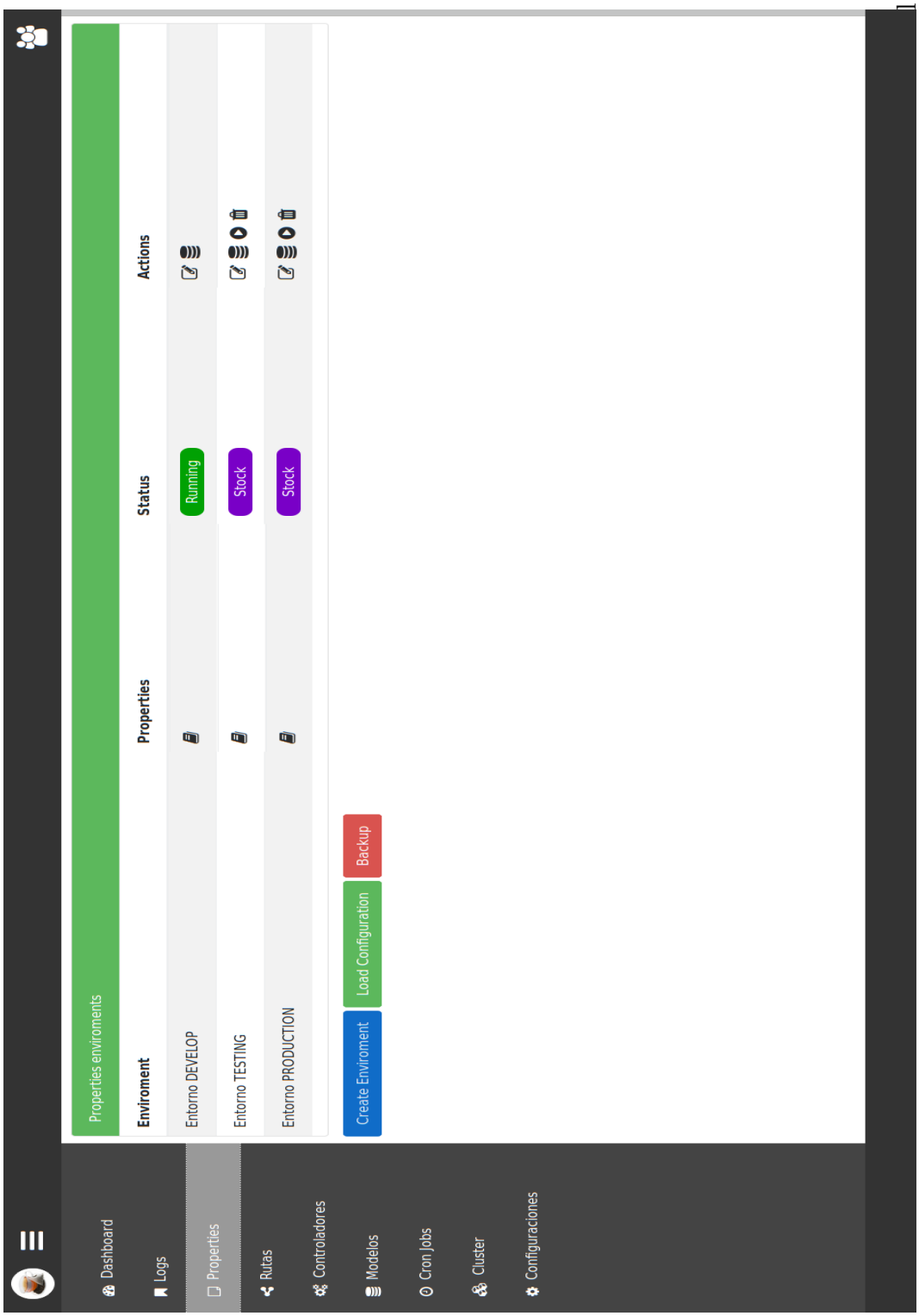


Figura 9.3: En esta figura se muestra la consola de administración de Earlgrey

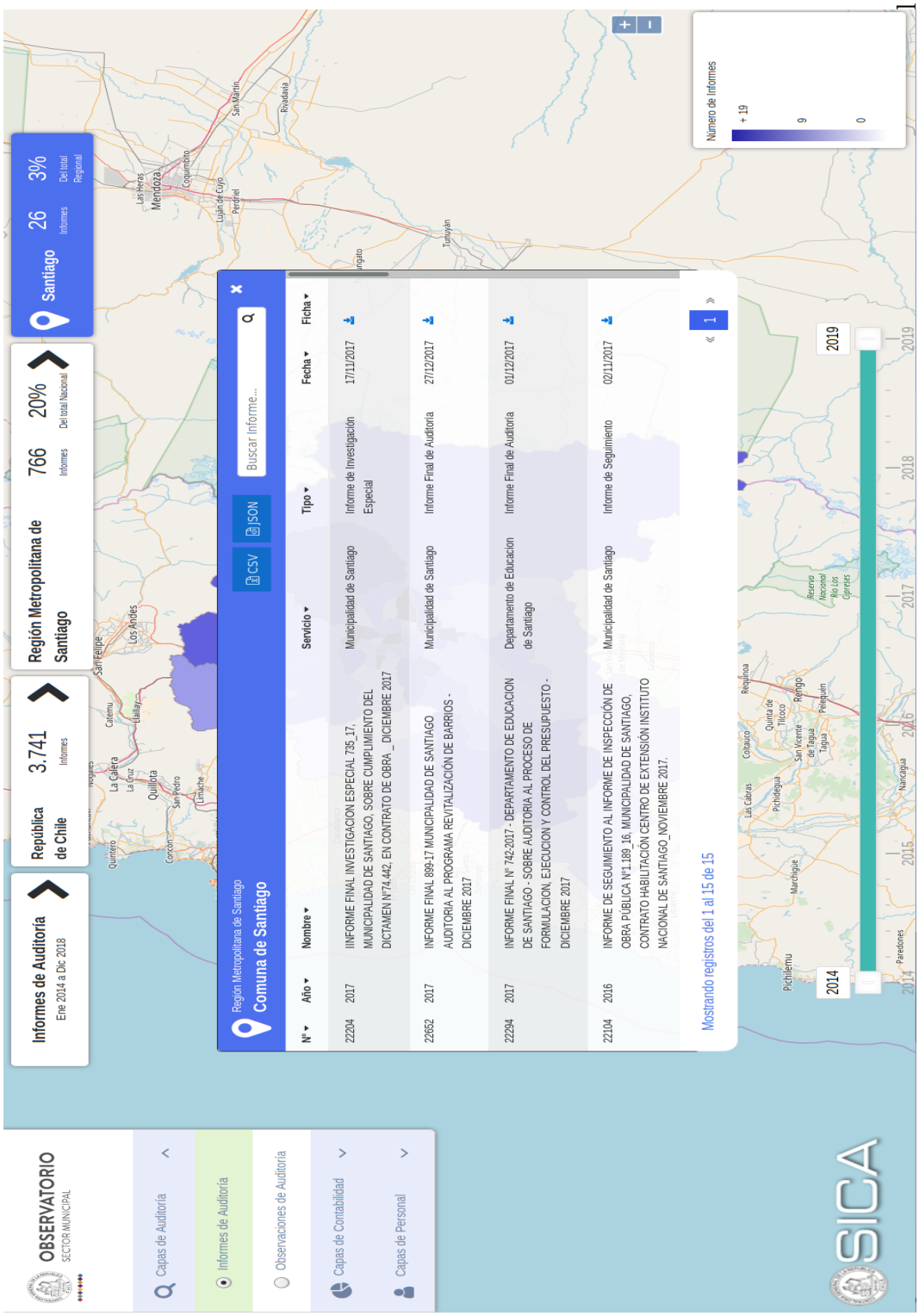


Figura 9.4: En esta figura se observa la versión productiva del Observatorio Sector Municipal

Capítulo 10

Conclusiones

Al finalizar el presente trabajo de construcción del framework Earlgrey, se pudo identificar las siguientes conclusiones:

- JAVA es un importante lenguaje de programación que ha estado vigente en la industria por mas de 20 años siendo uno de los favoritos por el mercado corporativo. Grandes empresas del sector tecnológico actualmente han optado por el uso de JAVA a pesar de partir inicialmente con lenguajes de nueva generación, debido a muchas características presentes en JAVA como la JVM. Sin embargo existe una gran brecha en comparación con otros lenguajes cuando nos referimos a la construcción de aplicaciones de una forma ágil, debido a la inexistencia de frameworks de fácil implementación que otorguen las cualidades de sus competidores.

- Se identificaron los beneficios en la construcción de un framework JAVA integrado que permita fabricar aplicaciones orientadas a servicios, con una implementación simplificada, que otorgue a los desarrolladores de herramientas y características que les permitan fabricar código y componentes con un menor esfuerzo.

- Se logro identificar de forma eficaz las características positivas presentes en frameworks de otros lenguajes, tales como el uso de ORM's, consolas de administración, API's RESTful embeidas y sintaxis simples. Cabe destacar que la mayoría de los framework de lenguajes como PHP, Ruby, Nodejs tiene pro-

piedades en común que han sido ampliamente adoptados y aceptados por la comunidad de desarrollo y vienen embebidos en estas distribuciones; Para obtener estas características en JAVA se deben hacer uso de multiples paquetes por separado, cada uno con su configuración independiente, lo que aumenta de forma significativa los tiempos de implementación o despliegue de una aplicación.

- Se identificaron las debilidades del lenguaje JAVA a la hora de construir aplicaciones en contraposición con lenguajes modernos y sus frameworks, caracterizando de esta forma los elementos que pueden ser mejorados a través de la implementación de un framework. Es importante destacar que una de las mayores falencias detectadas es el uso de properties en sistemas multi nodo y los logs en tiempo real.

- Se logro implementar con éxito un framework estable y funcional que lograra resolver las problemáticas identificadas y permitiera transferir los beneficios de otros framework a un solo entorno de desarrollo. El desarrollo permitió a través de solo un paquete java y la codificación de tan solo 1 linea, del despliegue de todo un sistema integrado listo para operar a peticiones.

- Se desarrollo un sistema productivo con éxito que permitió utilizar las características desarrolladas en el framework. El sistema desarrollado llamado "Observatorio Sector Municipal", producto desarrollado para la Controlaría General de la República. Este sistema permitió durante su desarrollo la detección temprana de múltiples fallas y características necesarias a incluir en el framework. El utilizar este proyecto como prueba de implementación durante todo el ciclo de construcción del framework, permitió un desarrollo incremental de las características y una aproximación a una implementación en servidores reales.

- La adhesión de contribuidores al proyecto y el uso del framework en otros proyectos, permitió desarrollar una pequeña comunidad de desarrollo que dio validación, soporte, revisión y apoyo a las tareas de codificación del earlgrey y sus características.

Capítulo 11

Trabajos Futuros

Haciendo un análisis profundo del estado del framework en su versión 0.4, se observa que aun queda mucho trabajo futuro por hacer. Los requerimientos de un framework cambian de la mano con la evolución del lenguaje y sus propiedades. Así mismo las nuevas tendencias de desarrollo conforman todo un set de posibilidades para abordar a futuro por el framework. Sin embargo a pesar esto se observan elementos que serian posibles de implementar en el mediano plazo para mejorar la usabilidad, así como su eficiencia. Los principales trabajos futuros detectados en el framework son.

- Debido a que JAVA utiliza la JVM, existe la posibilidad de que se superen muchas de las limitaciones de sintaxis del lenguaje a través del uso de lenguajes como Scala y Groovy. Estas limitaciones, que en muchos casos provocan redundancias de código y una baja eficiencia en ciertos algoritmos podrían ser superadas fácilmente a través del uso de fragmentos de código desarrollados en lenguajes con otros enfoques.

- Matcha ORM actualmente es parte del núcleo de Earlgrey, pero seria óptimo que siguiera una linea de desarrollo independiente, de manera que pudiera ser implementado en cualquier proyecto de software sin necesidad de usar Earlgrey. Matcha es un ORM liviano y de fácil implementación, ideal para proyectos de pequeña a mediana escala. Además se aumentaría el soporte para un mayor numero de bases de datos y características. Sin embargo a pesar de que sigan caminos distintos, Earlgrey implementaría de forma nativa Matcha por lo que su actual funcionamiento no se vería alterado.

- Dar soporte para comunicaciones en tiempo real, a través del protocolo Websocket. Esto permitiría entre otras cosas permitir el desarrollo de aplicaciones con posibilidades mensajes "Push & Pull" por parte de los clientes y servidores.

- Dar soporte para la fabricación de microservicios. Lo cual permitiría al framework ofrecer mayores posibilidades para desarrollar aplicaciones concurrente que requieren un balanceo de carga diferenciado.

- Maven, es una excelente herramienta para integrar dependencias y efectuar operaciones de compilación, pero la descentralización de sus repositorios es un problema a la hora de integrar dependencias en comparación con administradores de paquetes como NPM de Node.js. Si bien no es parte de la construcción del framework, la construcción de un repositorio central con las cualidades de NPM para el lenguaje JAVA seria una excelente herramienta para efectuar labores de desarrollo.

Bibliografía

[1] The Sails Company. Sails framework. url: https://sailsjs.com/.

[2] Daniel Ortego Delgado. Los 7 mejores frameworks de java de 2017.

[3] Vincent Driessen. A successful git branching model. url: http://nvie.com/posts/a-successful-git-branching-model/.

[4] Ganda Software Factory. ¿cómo utilizar git flow para gestionar las ramas en git? url: http://howto.gandasoftwarefactory.com/desarrollo-software/2016/como-gestionar-ramas-con-git-flow-20160718/.

[5] Antoni Fuentes. El número de empresas tecnológicas se disparó un 43 % durante la crisis. url: https://www.elperiodico.com/es/economia/20170408/empresas-tecnologicas-crecen-crisis-tic-5961994, 2017.

[6] Minko Gechev. Angular seed. url: https://github.com/mgechev/angular-seed.

[7] Antonio Leiva. ¿por qué empresas que empiezan con lenguajes modernos se vuelven a java? url: https://www.genbetadev.com/java-j2ee/por-que-empresas-que-empiezan-con-lenguajes-modernos-se-vuelven-a-java.

[8] Fomento y Turismo Ministerio de Economía. Estudio de levantamiento de necesidades tecnológicas en subsectores económicos. 2010.

[9] Prof. Josep Valor Prof. Sandra Sieber. Criterios de adopción de las tecnologías de información y comunicación. *Ebcenter*, 2008.

[10] Angelo Calvo Alfaro Sebastian Gonzalez Villena. Angular 2 seed cli. url: https://github.com/acalvoa/angular2seedcli.

[11] Pablo Sánchez. Consejos y buenas prácticas en programación. url: http://personales.unican.es/sanchezbp/teaching/faqs/programming.html.

[12] Joshua Thijssen. How to do stuff restful. url: http://restcookbook.com/.

Apéndice A

Plan director Geoportal "Observatorio Sector Municipal"

INFORMACIÓN CGR CON VISTA COMUNAL
PLAN DIRECTOR

CARACTERÍSTICAS DEL DOCUMENTO

Historial Versiones			
Fecha	**Versión**	**Creado por**	**Descripción**
29/06/2016	1.0	Alejandra Levill	Creación de Plan Director

Aprobación del documento			
Acción	**Preparó**	**Revisó**	**Aprobó**
Nombre	**Alejandra Levill**		**Esteban Navarro**
Cargo	**Jefe de Proyecto**		**Jefe de Área**
Fecha	**29/06/2016**		**30/06/2016**

Registro de cambios				
Nº	**Fecha**	**Tipo**	**Descripción del cambio**	**Solicitante**
1				
2				
3				

TABLA DE CONTENIDOS

1 INTRODUCCIÓN

Nombre del proyecto: Información CGR con vista comunal

Sigla del proyecto: GEOPORTAL Comunal

El propósito del documento es identificar y definir las necesidades y características del proyecto: Información CGR con vista comunal, en adelante **GEOPORTAL Comunal.** El detalle de cómo el GEOPORTAL Comunal cubrirá cada necesidad se detallara en los documentos de requerimiento, análisis y diseño respectivo que se construirán con cada jefe de proyecto y sus respectivas contrapartes de los Sistemas donde se capturará la información.

2 SITUACIÓN ACTUAL

En la actualidad la Contraloría no dispone de un sistema o portal donde se despliegue información que genera la esta entidad fiscalizadora de forma georreferencia, luego por estrategia se ha decidido desarrollar un Portal con información generada por Contraloría como primera versión y en una segunda versión añadir información que viene a la Contraloría y disponer capas de información según nivel de navegación, es decir, información a Regional, Comunal y cruzar con otras capas de información. El Portal no se realizaran análisis espacial solo se entregará información mediante una ficha definida.

3 OBJETIVOS DEL PROYECTO

El objetivo del Portal es entregar un canal donde se muestre la información con base a los antecedentes que llegan a la Contraloría y en los cuales existe algún pronunciamiento por parte de CGR facilitando la accesibilidad y fácil compresión de los antecedentes a los usuarios que deseen consultar.

En un corto plazo se dejará disponible en el Portal la información que se guarda en la Contraloría y disponibilizar esa información a los usuarios que deseen acceder a esos antecedentes.

4 RESULTADO ESPERADO

El resultado es disponer de un portal donde reúna información de Contraloría en materias donde se ha pronunciado e información que llega a la Contraloría para estudio, separadas por capas información que sean de fácil lectura y entregue fichas de información según nivel de navegación

5 ALCANCES DEL PROYECTO

En una primera instancia se entregará información a nivel comunal con respecto a las auditorias que se han realizados en las localidades con una ficha resumen con accesos específicos a los informes.

En una segunda instancia se incorporará los antecedentes sobre los pronunciamientos que ha realizado la CGR en relación a temas de Personal, Contabilidad u otras áreas que se estimen convenientes.

En una tercera etapa se incorporará información que llega y es guardada en Contraloría y así dejarla disponible en capas según nivel de información en las áreas que se encuentre accesible dichos antecedentes.

El portal no realizará análisis de geometrías, tampoco pretende ser un repositorio donde se descargaran reportes, solo será un Portal donde se pueda visualizar información clara y concisa donde el mínimo para consultar será el comunal.

6 SUPUESTOS

Para que el éxito del Portal es necesario que cada jefe de proyecto de los Sistemas que facilitaran la información debe estar en comunicación con sus respectivas contrapartes de negocios y entreguen las directrices hasta que nivel de información es posible obtener en sus respectivos sistemas y que información no se puede desplegar en el GEOPORTAL Comunal por ser información confidencial o por su casuística en el trato de dichos antecedentes.

7 RESTRICCIONES

El desarrollo del portal será abarcado en 3 grandes hitos y así cumplir con el plan estratégico de disponer del GEOPORTAL Comunal en conjunto con el lanzamiento del nuevo Portal institucional.

Hito 1: Disponer de los Informes de Auditoría por comuna, esa información se dejará disponible según definición entregada por el jefe de proyecto de SICA y su contraparte de negocio

Hito 2: Al hito 1 se sumará la entrega de las capas de información de los pronunciamientos que ha realizado en otras áreas como son en el ámbito personal, contabilidad.

Hito 3: Se incorporará las capas de información de los antecedentes que ingresan y se guardan en la Contraloría en las áreas de auditoría, personal, contabilidad.

8 DEPENDENCIAS

El portal como se mostrará información de varios sistemas tiene las siguientes dependencias,

1.- Sistema de Auditorias, SICA

2.- Sistema de Personal del Estado, SIAPER

3.- Sistema de Contabilidad de la Nacional, SICOGEN

9 ANÁLISIS DE COSTO BENEFICIO

Será el primer portal que pueda articular y unificar información donde contraloría se ha pronunciado en sus diversas áreas, permitiendo al usuario tener en un solo lugar antecedente con información concisa.

Con esto será los primeros pasos para logra eficiencia, llegar a la excelencia y entregar antecedentes de una manera estrategia para todo aquel usuario que desee acceder a la información de la Contraloría.

10 ANÁLISIS DE RIESGO PRELIMINARES

Como GEOPORTAL Comunal tendrá conexión directamente con las bases de datos de cada Sistema que esté involucrado, se debe tener en consideración que si algún sistema cambia su estructura se puede ver mermada la información que se muestra de esa área.

El desconocimiento de la calidad de información de cada Sistema, puede generar que algunas fichas no muestren la información en su plenitud como se ha definido.

Se deberá tomar las medidas pertinentes para que el GEOPORTAL Comunal sea capaz de mostrar información aunque algún Sistema se encuentre en contingencia, es decir, despliegue los antecedentes de las otras áreas involucradas.

11 HITOS DEL PROYECTO

Hito	Descripción	Fecha de Entrega
HITO 1	Disponer de los Informes de Auditoría por comuna	Septiembre 2016
HITO 2	Incorporar capas de información de los pronunciamientos que ha realizado en otras áreas	enero 2017
HITO 3	Incorporará las capas de información de los antecedentes que ingresan y se guardan en la Contraloría	Junio 2017

Apéndice B

Casos de uso Geoportal "Observatorio Sector Municipal"

GEOPORTAL MUNICIPAL
CASOS DE USO

CARACTERÍSTICAS DEL DOCUMENTO

Historial Versiones			
Fecha	**Versión**	**Creado por**	**Descripción**
05/01/2017	1.0	Alejandra Levill	Descripción de Geoportal Municipal

Aprobación del documento			
Acción	**Preparó**	**Revisó**	**Aprobó**
Nombre	**Alejandra Levill**		**Esteban Navarro**
Cargo	**Jefe proyecto**		**Jefe Área**
Fecha	**05/01/2017**		**09/01/2017**

Registro de cambios				
N°	**Fecha**	**Tipo**	**Descripción del cambio**	**Solicitante**
1				
2				
3				

TABLA DE CONTENIDOS

1 NOMBRE DE CASO DE USO

Nombre	Geoportal Municipal		
Código	GEOMUN-01		
Importancia	Alta		
Creado por	Alejandra Levill	**Fecha**	05/01/2017
Modificado por			
Nombre		Fecha	
Actores			
Funcionario CGR No funcionario CGR			
Componentes de la Pantalla			

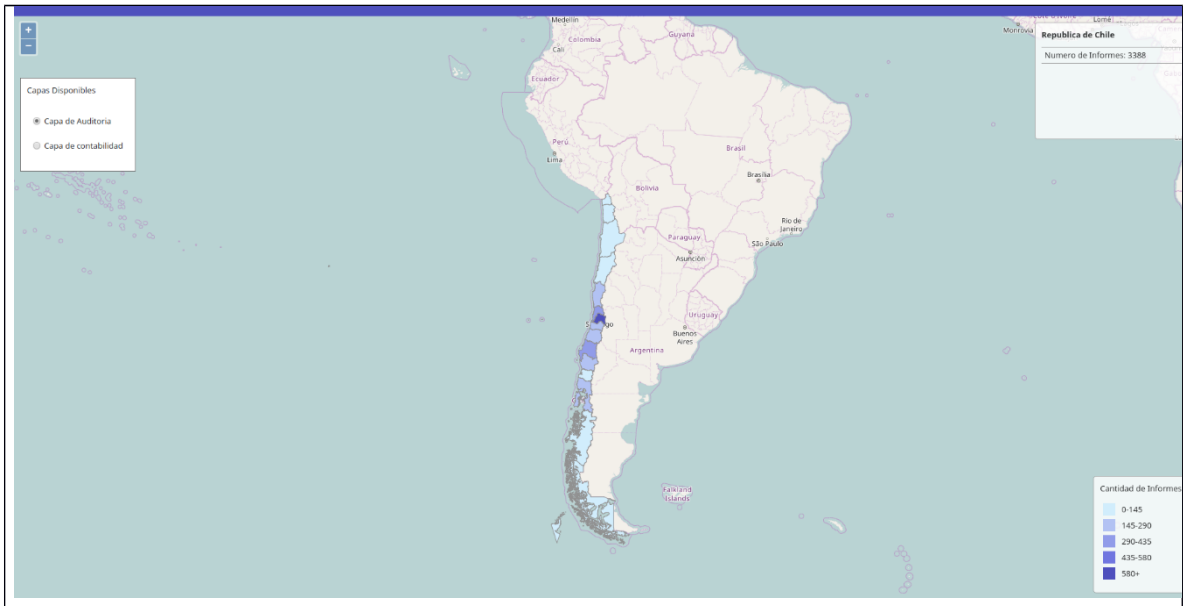
El Geoportal Municipal, estará embebido en el actual portal de Contraloría, en el se podrá acceder a información preparada o realizada por la Contraloría.

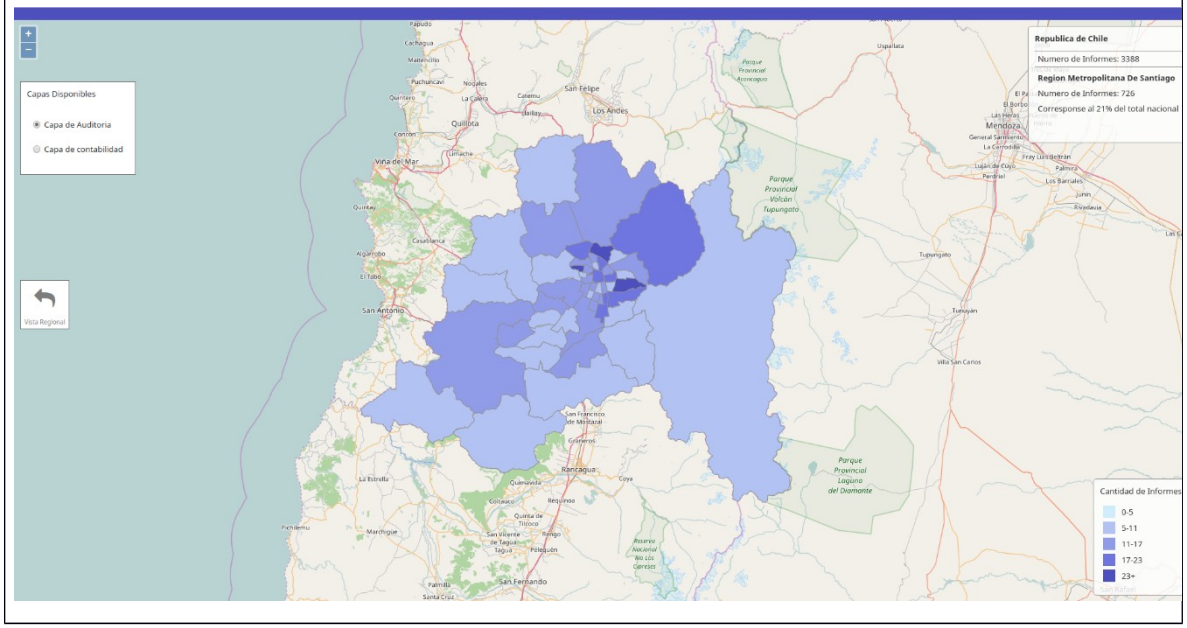
Se podrá acceder a la información, la cual estará disponible en un árbol dividido por capas de información, accediendo por medio de Region-Comuna o direccionado directamente a la Comuna que desea consultar

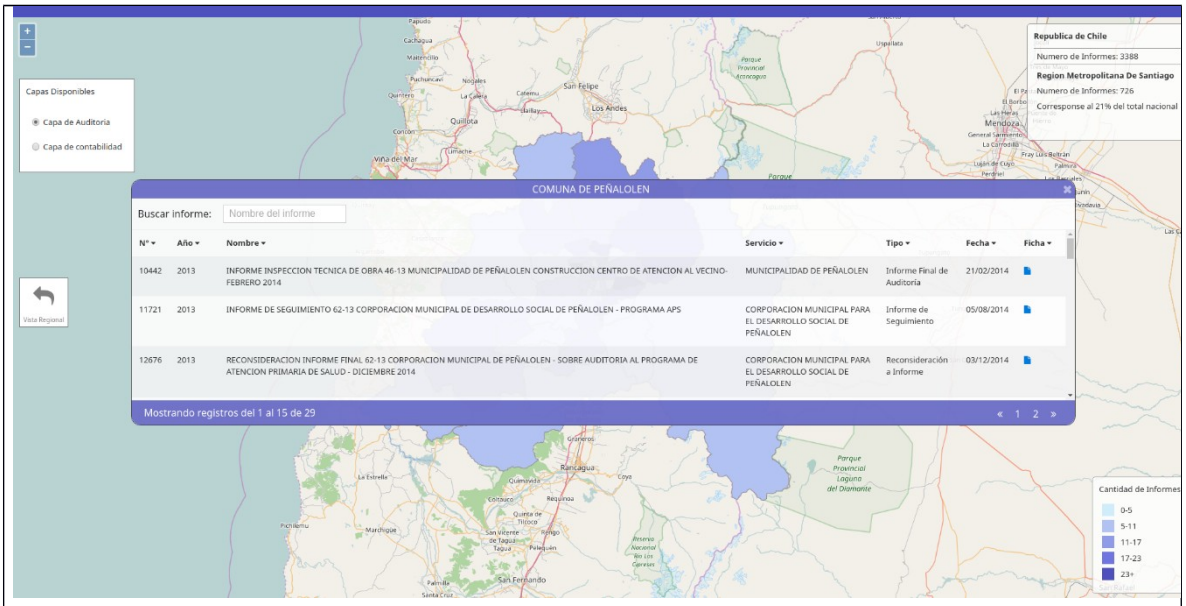
La información a mostrar será

- Auditoria
- Contabilidad
- Personal

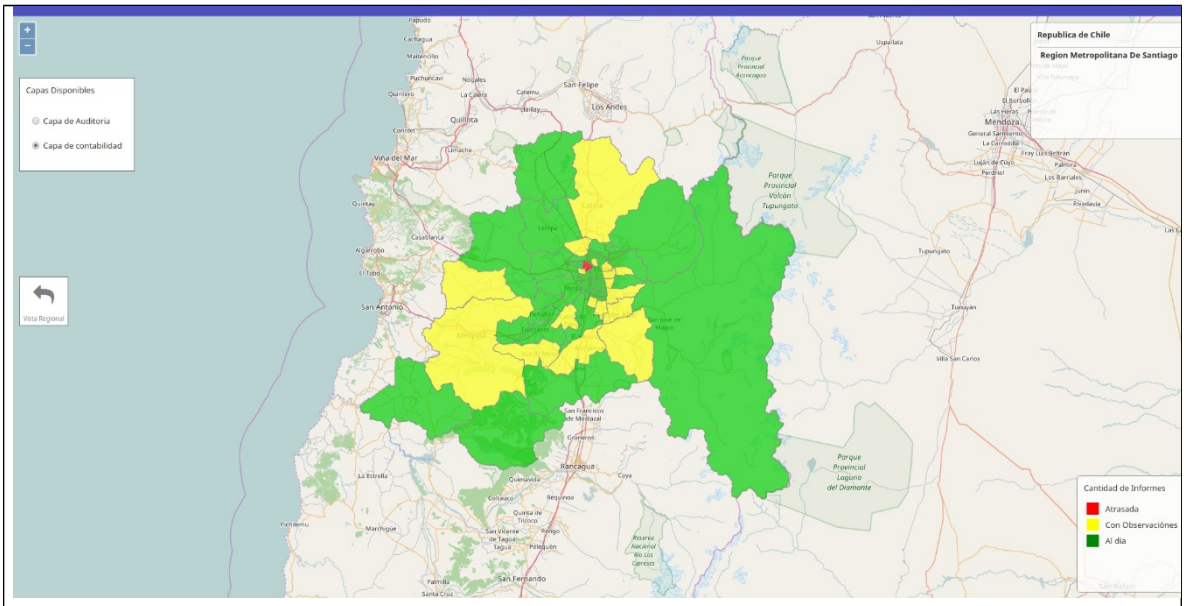
Las cuales en primera versión se dejará disponible 5 capas de información que abarquen las áreas indicadas.


Capas Disponibles
Capa de Auditoria
Capa de contabilidad
Republica de Chile
Numero de Informes: 3388
Cantidad de Informes
0-145
145-290
290-435
435-580
580+


Capas Disponibles
Capa de Auditoria
Capa de contabilidad
Vista Regional
Republica de Chile
Numero de Informes: 3388
Region Metropolitana De Santiago
Numero de Informes: 726
Corresponse al 21% del total nacional
Cantidad de Informes
0-5
5-11
11-17
17-23
23+


Capas Disponibles
Capa de Auditoria
Capa de contabilidad
Vista Regional
Republica de Chile
Numero de Informes: 3388
Region Metropolitana De Santiago
Numero de Informes: 726
Corresponse al 21% del total nacional
COMUNA DE PEÑALOLEN
Buscar informe:
Nombre del informe
N° Año Nombre Servicio Tipo Fecha Ficha
10442 2013 INFORME INSPECCION TECNICA DE OBRA 46-13 MUNICIPALIDAD DE PEÑALOLEN CONSTRUCCION CENTRO DE ATENCION AL VECINO- FEBRERO 2014 MUNICIPALIDAD DE PEÑALOLEN Informe Final de Auditoria 21/02/2014
11721 2013 INFORME DE SEGUIMIENTO 62-13 CORPORACION MUNICIPAL DE DESARROLLO SOCIAL DE PEÑALOLEN - PROGRAMA APS CORPORACION MUNICIPAL PARA EL DESARROLLO SOCIAL DE PEÑALOLEN Informe de Seguimiento 05/08/2014
12676 2013 RECONSIDERACION INFORME FINAL 62-13 CORPORACION MUNICIPAL DE PEÑALOLEN - SOBRE AUDITORIA AL PROGRAMA DE ATENCION PRIMARIA DE SALUD - DICIEMBRE 2014 CORPORACION MUNICIPAL PARA EL DESARROLLO SOCIAL DE PEÑALOLEN Reconsideración a Informe 03/12/2014
Mostrando registros del 1 al 15 de 29
« 1 2 »
Cantidad de Informes
0-5
5-11
11-17
17-23
23+


Capas Disponibles
Capa de Auditoria
Capa de contabilidad
Republica de Chile
Porcentaje de Cumplimiento
0% - 20%
20% - 40%
40% - 60%
60% - 80%
80% - 100%



Flujo Normal

Paso	Acción del usuario	Acción/Respuesta del sistema

Flujo Alternativo

Paso	Acción del usuario	Acción/Respuesta del sistema

Precondiciones

Cada Sistema debe brindar la metadata que se mostrará.

Post condiciones

Caso de Éxito: Se puede acceder a la información de cada

Caso de Fracaso: No muestra información de las capas

Excepciones