UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Arquitectura baseada en técnicas de procesamento masivo para a xestión de indicadores de rendemento nunha plataforma de fluxos de traballo.

Autor:
Jorge López Seijas

Directores:
Manuel Lama Penín
Juan Carlos Vidal Aguiar
Víctor José Gallego Fontenla

Grao en Enxeñaría Informática

Setembro de 2016

Traballo de Fin de Grao presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Grao en Enxeñaría Informática

D. Manuel Lama Penín, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, e **D. Juan Carlos Vidal Aguiar**, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela e **D. Víctor José Gallego Fontenla**, Investigador en formación no Centro Singular de Investigación en Tecnoloxías da Informacion da Universidade de Santiago de Compostela.

INFORMAN:

Que a presente memoria, titulada *Arquitectura baseada en técnicas de procesamento masivo para a xestión de indicadores de rendemento nunha plataforma de fluxos de traballo*, presentada por **D. Jorge López Seijas** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo nosa dirección no Departamento de Electrónica e Computación da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 7 de Setembro de 2016:

O director,

Manuel Lama Penín

O codirector,

Juan Carlos Vidal Aguiar

O codirector,

Víctor José Gallego Fontenla

O alumno,

Jorge López Seijas

Agradecementos

Á miña familia, por estar sempre aí apoiandome e confiando en min dandome toda a axuda da que dispoñían para que isto fose posible.

A D. Massé Guillaume, pola súa axuda desinteresada á hora de colaborar comigo e darme información para realizar a incorporación do seu desenvolvemento no presente proxecto.

Aos meus grandes amigos e compañeiros que fixeron que estes catro anos se pasaran voando e estiveron aí ante as dificultades atopadas durante todo este proceso de formación.

Aos meus titores, por confiar en min para a realización deste traballo e pola axuda proporcionada durante todo o proxecto.

Índice xeral

Índice de figuras

Índice de cadros

Capítulo 1

Introdución

Neste capítulo pretendese situar no contexto ao lector para que dispoña dunha pequena base para entender e seguir de forma sinxela e precisa o proxecto desenvolvido. Para conseguir iso presentase o problema a resolver e o estado actual da súa posible solución. De forma adicional mostraranse os obxectivos do presente TFG e a organización da memoria.

1.1. Motivación

Un fluxo de traballo describe o conxunto de actividades que é preciso realizar para acadar un obxectivo dado, incluíndo a coordinación temporal de ditas actividades así como os recursos que as levarán a cabo. Deste xeito, as vantaxes da aplicación dos fluxos de traballo son claros en canto á automatización das tarefas e á optimización de tempos de traballo. Por esta razón actualmente están sendo implantados nunha multitude de dominios, entre os que destacan a medicina, a industria, a educación e o marketing. Sen embargo, a pesares destas claras vantaxes, os beneficios de aplicar os fluxos de traballo non son evidentes, sobre todo se temos en conta que a súa implantación supón, polo xeral, elevados custes tanto dende o punto de vista económico como da configuración da organización na que se implantan. Para evidenciar esta mellora, é necesario medir e avaliar unha serie de indicadores de rendemento que indiquen en que medida a aplicación dos fluxos de traballo supón un beneficio para a organización.

Un **indicador clave de rendemento** (*key performance indicator*, en inglés, KPI) [1] é un parámetro que se define para cada dominio ou tarefa a levar a cabo e que sinala en que medida a implantación dun fluxo de traballo leva aparellados beneficios para a organización, ou o que é o mesmo, se o fluxo de traballo está a obter os resultados esperados. A través dun conxunto de KPIs pódese medir o nivel de calidade dun proceso, tanto en instantes particulares de tempo como o seu progreso ó longo do tempo. É importante mencionar que os KPIs dependen tanto das tarefas que se desexan resolver como dos obxectivos da propia orga-

nización na que se implantan. Por exemplo, nun fluxo de traballo dun dominio médico o indicador clave de rendemento pode ser o número de pacientes que foron tratados nun determinado período, mentres que noutro fluxo de traballo orientado ao marketing o indicador pode ser o número de usuarios activos con idade comprendida entre os 35 e os 50 anos, que vivan coa sua parella ou fillos e que teñan unha capacidade adquisitiva elevada.

O presente Traballo Fin de Grao (TFG) pretende abordar a variabilidade das KPIs asociadas aos fluxos de traballo, desenvolvendo un editor que permita definir de forma sinxela as diferentes KPIs que poidan ser de interese para calquera tipo de fluxo de traballo, tales como o número de veces que os usuarios executaron un fluxo de traballo, o seu tempo medio de execución, o número de usuarios cun determinado perfil que levaron a cabo unha tarefa, etc. Ademais de facilitar a definición das fórmulas asociadas a unha KPI, o presente TFG tamén automatiza a súa evaluación e o seu posterior almacenamento en base de datos, o editor permite elixir a compoñente gráfica que mellor se adapte para visualizar o valor puntual ou a súa evolución ao longo do tempo, dependendo do tipo de indicador definido, de xeito que o usuario poderá acceder a toda a información das KPI definidas no editor dende un cadro de mandos personalizado.

1.2. Contexto técnico

Neste apartado pretendese poñer en contexto ao lector comentando de xeito moi breve os aspectos técnicos relacionados co desenvolvemento do TFG así como os pasos para solucionar o problema citado anteriormente.

En primeiro lugar debemos dispoñer dos rexistros de eventos dos que se extraerá a información necesaria para o cálculo dos valores dos KPI. Sen embargo, o presente TFG centrarase en como se van a procesar estes rexistros sen ter en conta como foron xerados. Isto implica que dentro do sistema poderá haber información que nunca se empregue ou que simplemente non sexa relevante para o procesamento das KPIs que existan dentro da plataforma.

Para o procesamento dos rexistros de eventos comentados anteriormente, farase uso de Scalding [2], unha libraría de Scala que permite o procesamento de traballos *Map-Reduce* e que se integra de xeito sinxelo coa infraestrutura Apache Hadoop [3]. A elección desta plataforma para o procesado dos datos ven motivada pola potencial elevada cantidade de rexistros xerados polo xestor de fluxos de traballo durante a súa execución.

A través de Scalding definirase a expresión da KPI que se computará mediante Apache Hadoop, indicando que datos se teñen que filtrar dos rexistros de eventos e as operacións que se deben executar sobre eses datos. Esta expresión, xunto con outra información da KPI (como poden ser os seus datos básicos, ou as posibles representación que pode tomar: gráficas e non gráficas), almacenaranse dentro dunha base de datos noSQL.

Este proxecto esta formado por dúas partes ben diferenciadas. Unha primeira parte, que consistirá nun editor que permita a **xestión completa das KPI**, o cal inclúe as seguintes funcionalidades:

- Creación, que permite elaborar unha diversa variade de KPIs, enviando a súa formulación asociada, escrita en Scala mediante o editor Scalakata [4], creado por Massé Guillaume, para ser executada en Apache Hadoop que se encarga de realizar o procesamento *Map-Reduce* e almacenamento dos datos obtidos de dita operación en base de datos.
- Modificación, que permite tanto a edición dos datos básicos dunha KPI ou a súa formulación coma se se desexa engadir novas representacións ou editar as xa existentes, enviando de novo a formulación asociada a Apache Hadoop para que realice as operacións mencionadas durante o proceso de creación.
- Eliminación, que elimina tanto os datos introducidos polo usuario para deseñar as KPIs como os datos procesados por Apache Hadoop para dita KPI.
- Filtrar as KPIs mediante un patrón, deste xeito conseguese amosar na lista de xestión das KPIs únicamente as KPIs que coinciden con dito patrón.

A segunda parte da interface web consistirá nun **cadro de mando** ou *dashboard* que permite:

- Amosar os valores das KPIs, que o usuario considere oportunas para mostrar unha ou varias representacións a partir dos valores obtidos e procesados por Apache Hadoop que almacenou anteriormente en base datos.
- A súa configuración e ordenación das diferentes representacións ao usuario, mediante o seu movemento e redimensionamento.
- O gardado, se o usuario o desexa, do estado actual do dashboard, para que no seguinte inicio, o dashboard manteña as mesmas representacións exactamente onde o usuario as posicionou.

1.3. Obxectivos

O obxectivo principal do proxecto é o **desenvolvemento dunha arquitectura orientada a servizos que permita a creación, execución e visualización de indicadores clave de rendemento**, usando para elo técnicas de procesamento masivo de datos. De xeito máis concreto, os obxectivos do TFG son os seguintes:

- **Desenvolvemento dunha infraestrutura que permita o almacenamento e a execución dos KPIs** creadas polos usuarios. Para elo, seguirase o paradigma das arquitecturas orientadas a servizos que permitirá tanto a execución nunha plataforma de procesamento masivo das KPIs definidas como o seu acceso e almacenamento nunha base de datos, facilitando así a súa manipulación e posterior reutilización.

- **Desenvolvemento dunha interface gráfica para a xestión dos KPIs**, permitindo a súa creación, modificación, eliminación, consulta e representación a través de diferentes compoñentes visuais. Prestarase especial atención á especificación das expresións e/ou fórmulas das KPI nunha linguaxe orientada ao procesamento masivo de datos.

- **Integración da infraestrutura de KPI dentro dunha plataforma que fai uso de fluxos de traballo**, e na que as KPI son diferentes para cada tipo de fluxo de traballo.

1.4. Organización da memoria

Nesta sección documéntanse as diferentes etapas, fases e desenvolvementos que se realizaron para poder resolver os obxectivos definidos neste TFG. Para maior claridade explicase cada unha das etapas e tarefas que se levaron a cabo:

- No ***capítulo 2*** mostrase a identificación e o análise de requisitos realizado no TFG.

- No ***capítulo 3*** descríbese a xestión levada a cabo ao longo da realización do TFG. Neste capítulo encontrase a definición do alcance, o análise de riscos, a xestión da configuración utilizada, a planificación do proxecto, o cálculo dos custos e a xestión das comunicacións.

- No ***capítulo 4*** descríbense as ferramentas empregadas para o desenvolvemento do TFG, o análise das tecnoloxías existentes no momento, patróns de deseño e arquitectura a empregar.

- No ***capítulo 5*** descríbese o deseño e a implementación do software a desenvolver. Este capítulo está dividido nas seccións correspondentes ao deseño da arquitectura de alto nivel e ao deseño do software, incluído os diagramas de clases e de secuencia, que describen a interacción entre os compoñentes do sistema.

- No ***capítulo 6*** descríbense as probas tanto unitarias como de integración para levar a cabo a validación do sistema.

- O ***capítulo 7*** está formado polas conclusión obtidas da realización do proxecto, os problemas atopados nel así como as posibles melloras que se poderían realizar no futuro.

  Ademais, tamén se inclúen dous apéndices:

- O ***apéndice I*** contén o manual de instalación do sistema.
- O ***apéndice II*** contén o manual de usuario indicando algunhas pautas e operacións básicas para a utilización do sistema.

Capítulo 2

Análise dos Requisitos

Este capítulo tratará unha das fases máis importantes dentro do desenvolvemento dun proxecto software: a análise dos requisitos. O obxectivo fundamental desta fase pasa por coñecer o que quere o cliente, como o quere e as posibles restricións que se deberán ter en conta en todas as fases do desenvolvemento.

Como se pode supoñer, unha vez comentado o parágrafo anterior, esta tarefa debe ser levada a cabo con sumo coidado, debido a que unha mala análise dos requisitos pode dar lugar a que o proxecto fracase.

Para a súa realización procederase en primeiro lugar a definir os obxectivos, (¿porqué queremos construír o software?), e o alcance, (¿Ata onde queremos chegar?), do proxecto, para ter unha base para facer un estudo dos casos de uso do sistema que se van a realizar e dos cales se realizará posteriormente un análise dos requisitos. Estes requisitos deben ser moi ben definidos debido a que serán a base de todo o proxecto e representan o contrato entre cliente e desenvolvedor. Os casos de uso extraídos serán obtidos e discutidos en sucesivas reunións co cliente, xunto cos requisitos peor definidos, deste xeito procurase construír un sistema que se adapte o mellor posible ao cliente.

2.1. Obxectivos e alcance do proxecto

O **obxectivo** principal do desenvolvemento é o de proporcionar unha ferramenta que permita a edición e visualización das KPI (Key Performance Indicator) que permita medir o nivel de rendemento dun proceso ou fluxo de traballo de xeito sinxelo. Os obxectivos individuais a acadar son os seguintes:

- Realizar unha interface sinxela, intuitiva e usable.
- Facilitar o desenvolvemento de todo tipo de KPIs, simplificando a creación das que sigan un patrón moi habitual.
- Mostrar mediante diversas formas de representación os resultados das KPIs.

Con respecto ao **alcance** do proxecto, este define o que se vai realizar e o que vai quedar excluído do proxecto actual, en varias reunión chegouse a seguinte conclusión do que se vai á realizar:

- Para realizar o desenvolvemento empregarase o framework para o desenvolvemento web denominado React. A interface gráfica deberá ser usable dende o punto de vista estético e intuitivo, empregando a regra KISS.
- Para poder implementar a personalización do dashboard, este deberá permitir asociar a cada usuario un dashboard, xunto coas KPIs que o usuario decidiu representar no seu dashboard.
- Para a representación de gráficas empregaranse librerías/bibliotecas externas, para facer máis sinxelo o desenvolvemento introducindo parámetros a representar. Ademais debe permitir a súa integración co framework web citado anteriormente.
- Para a parte de administración, debese permitir que calquera usuario autorizado con un nivel de coñecemento medio, poida crear unha KPI, nas diferentes etapas durante o proceso de creación (datos, formulación e representación), podendo editala ou engadir un ou máis xeitos de representación sobre unha mesmas KPI.

2.2. Análise dos casos de uso

É importante facer un análise de casos de uso xa que estes pretenden definir situacións reais de accións que os usuarios poden levar a cabo no sistema desenvolto e poder deste xeito ter a funcionalidade desexada nun nivel de abstracción superior de en que consiste o sistema a desenvolver. Para realizar unha correcta identificación dos casos de uso debe analizarse en primeiro lugar que actores interactuarán co sistema.

2.2.1. Actores

Defínese actor a calquera entidade externa ao sistema que garda relación con este e demanda unha necesidade ao sistema. Estes poden ser usuarios reais (persoas) ou outros sistemas externos que se comuniquen co sistema.

No caso deste proxecto, temos dous actores, que poden realizar diferentes funcións. Por unha banda, aínda que dentro do sistema poidan coexistir múltiples usuarios, con diferentes parámetros a amosar, onde cada un deles pode editar o nivel de detalle, as accións que poden levar a cabo no sistema son sempre as mesmas. Por outra banda temos o actor que realiza as operacións necesarias para xerar os rexistros dos fluxos de traballo que se empregaran neste proxecto para

obter os datos en función da KPI deseñada. A especificación formal detallada destes actores atópase no cadro 2.1 e 2.2:

Cadro 2.1: Actor 1. Representa os usuarios do sistema.

ID	*AC-001*
Nome	*Usuario*
Descrición	*Este actor representa aos usuarios do sistema, que interactuarán con este para o tratamento das KPIs, parte de administración, e a súa representación nun dashboard, parte de visualización de resultados.*

Cadro 2.2: Actor 2. Representa o xestor de fluxos de traballo.

ID	*AC-002*
Nome	*Xestor de fluxos de traballo*
Descrición	*Este actor representa a parte do sistema encargada de realizar os rexistros dos fluxos de traballo para o posterior uso no sistema a desenvolver, sen entrar en detalle de como os xera.*

2.2.2. Casos de Uso

Segundo os expertos de IBM os casos de uso [6] son:

> *Los casos de uso se crean para refinar un conjunto de requisitos de acuerdo con una función o tarea. En lugar de la tradicional lista de requisitos que quizá no trate de forma directa el uso de la solución, los casos de uso reúnen requisitos comunes basados en el tipo de función u objetivo. Los casos de uso definen qué harán los usuarios o funciones en la solución y un proceso empresarial define cómo realizarán esas funciones.*

Dito iso unha correcta identificación dos casos de uso que forman parte do sistema pode ser moi importante para ter unha boa identificación dos requisitos que definen o proxecto software a desenvolver, xa que proporcionan dun xeito bastante abstracto unha representación das actividades e as posibles iteracións que o usuario real pode levar a cabo dentro del.

Por outra banda, os casos de uso, como se comentou anteriormente, aportan unha representación o suficientemente sinxela de comprender para amosar ao cliente e deste xeito poden ser discutidas con el para a súa validación nas continuas revisións.

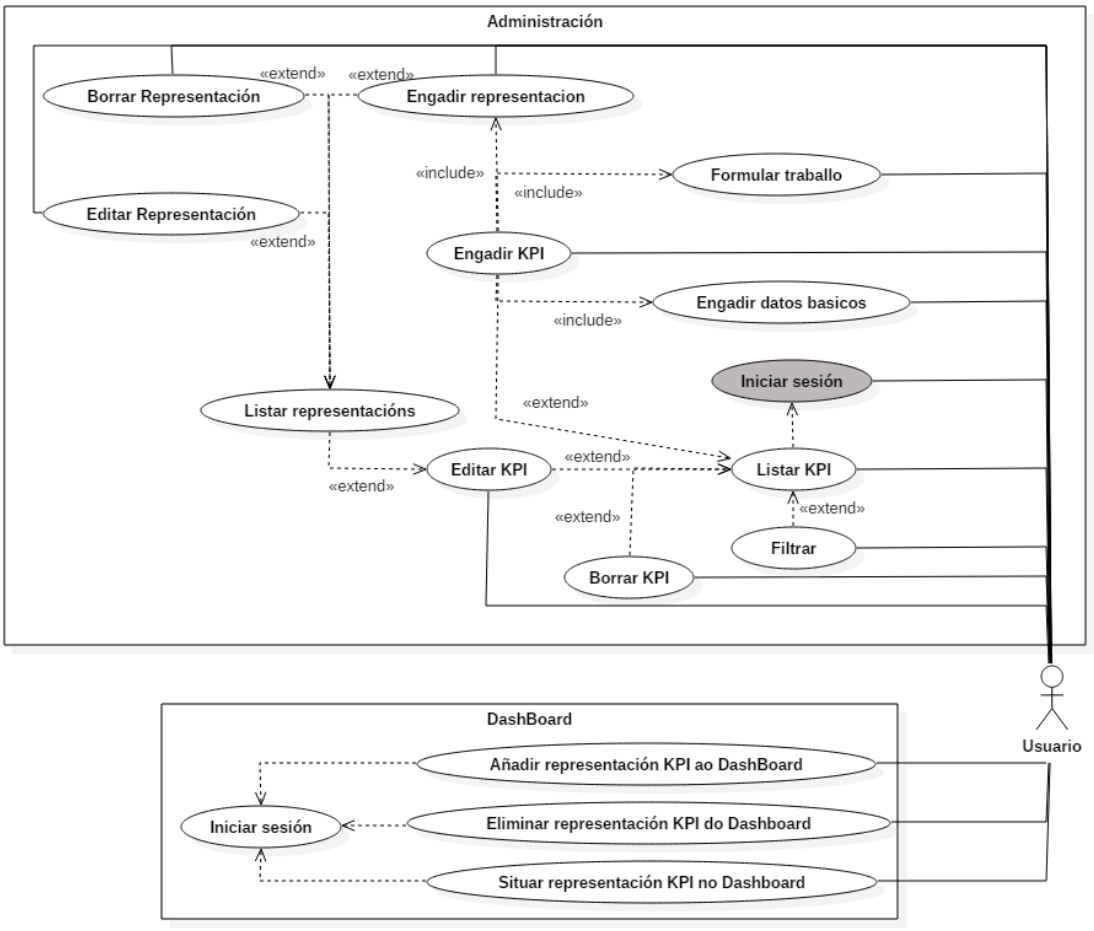

Figura 2.1: Diagrama de casos de uso do sistema.

Na figura 2.1 mostrase unha representación gráfica de forma global dos casos de uso identificados no sistema. A secuencia de actividades esperada polos usuarios expertos do sistema de edición de KPIs, representada na figura 2.1, é o seguinte: Cando un experto desexa empregar o sistema, este accede a aplicación co seu usuario e contrasinal. Dentro da parte administrativa de KPIs pódense realizar as operacións de creación ou edición onde existen dúas partes diferenciadas: a primeira onde se engade a formulación para indicar o que ten que procesar a KPI e outra onde se realiza o tratamento da representación. A segunda parte do editor é o Dashboard do usuario onde elixe as KPIs que quere visualizar, a súa distribución na ventá, etc.

Nos cadros de especificación formal dos casos de uso, inclúese unha pequena descrición de cada un deles e unha clasificación segundo a importancia para o cliente, mediante os seguintes valores:

- **Vital**: Indica que este caso de uso é imprescindible para o cliente e o correcto

funcionamento do sistema.

- **Importante**: Indica que ese caso de uso concreto é necesario pero que non esixe unha implementación inmediata deste.

- **Desexable**: Indica que este caso de uso tería valor para o usuario, debido a que facilita a súa labor ou entendemento, proporcionando un certa funcionalidade extra.

Cadro 2.3: Caso de uso Listar KPI.

ID	CU-001
Nome	Listar KPI
Descrición	O usuario debe ter a posibilidade de listar todas as KPIs que existen no sistema.
Importancia	Vital

Cadro 2.4: Caso de uso Engadir KPI.

ID	CU-002
Nome	Engadir KPI
Descrición	O usuario debe ter a posibilidade de engadir novas KPIs ao sistema. Este caso de uso é unha extensión do caso de uso de Listar KPI.
Importancia	Vital

Cadro 2.5: Caso de uso Engadir datos basicos.

ID	CU-003
Nome	Engadir datos basicos
Descrición	Funcionalidade necesaria para o caso de uso Engadir KPI, onde o usuario introducira os datos básicos da KPI, entre eles, Nome, descrición, palabras chave, tempo de refresco.
Importancia	Vital

Cadro 2.6: Caso de uso Formular traballo.

ID	CU-004
Nome	Formular traballo
Descrición	Funcionalidade necesario para o caso de uso Engadir KPI, onde o usuario debe poder realizar unha formulación (mediante código) para obter os valores requiridos dos rexistros para unha determinada KPI.
Importancia	Vital

Cadro 2.7: Caso de uso Engadir representación.

ID	CU-005
Nome	Engadir representación
Descrición	O usuario debe ter a posibilidade de engadir unha ou varias representacións sobre unha KPI. Esta funcionalidade é necesaria para o caso de uso Engadir KPI, mentres que é unha funcionalidade opcional cando se edita unha KPI.
Importancia	Vital

Cadro 2.8: Caso de uso Listar KPI.

ID	CU-006
Nome	Borrar KPI
Descrición	O usuario debe poder eliminar do sistema unha KPI existente xunto cos seus datos.
Importancia	Vital

Cadro 2.9: Caso de uso Editar KPI.

ID	CU-007
Nome	Editar KPI
Descrición	O usuario debe ter a posibilidade de editar unha KPI existente dentro do sistema. Esta funcionalidade é unha extensión do caso de uso Listar KPI.
Importancia	Importante

Cadro 2.10: Caso de uso Listar representación.

ID	CU-008
Nome	Listar representación
Descrición	Extensión do caso de uso Editar KPI, que permite mostrar todas as representacións que posúe unha KPI.
Importancia	Vital

Cadro 2.11: Caso de uso Editar representación.

ID	CU-009
Nome	Editar representación
Descrición	Extensión do caso de uso Listar representación, que permite modificar os datos asociados a unha representación sen poder modificar a súa forma.
Importancia	Importante

Cadro 2.12: Caso de uso Borrar representación.

ID	CU-010
Nome	Borrar representación
Descrición	Extensión do caso de uso Listar representación, que permite eliminar unha representación dentro das representacións dispoñible dentro dunha KPI concreta.
Importancia	Vital

Cadro 2.13: Caso de uso Filtrar

ID	CU-011
Nome	Filtrar
Descrición	Extensión do caso de uso Listar KPI, que permite procurar ou mostrar as KPI en función da coincidencia co seu nome, descrición ou palabras chave.
Importancia	Desexable

Cadro 2.14: Caso de uso Añadir representación KPI ao Dashboard

ID	CU-012
Nome	Añadir representación KPI ao Dashboard
Descrición	Caso de uso que permite engadir representacións de KPI existentes no sistema para que sexan visualizados dentro do dashboard do usuario.
Importancia	Vital

Cadro 2.15: Caso de uso Eliminar representación KPI ao Dashboard

ID	CU-013
Nome	Eliminar representación KPI ao Dashboard
Descrición	Caso de uso que permite eliminar as representacións que teña o usuario actualmente debuxadas dentro do dashboard.
Importancia	Vital

Cadro 2.16: Caso de uso Situar representación KPI no Dashboard

ID	CU-014
Nome	Situar representación KPI no Dashboard
Descrición	Caso de uso que permite tanto mover como redimensionar as representacións dentro do dashboard.
Importancia	Desexable

Cadro 2.17: Caso de uso Situar representación KPI no Dashboard

ID	CU-015
Nome	Iniciar Sesión
Descrición	Caso de uso que permite acceder a un usuario ao sistema para realizar as operacións dentro del.
Importancia	Importante

2.3. Análise de requisitos

Despois de ter definidos os obxectivos, o alcance e o análise de casos de uso do proxecto a desenvolver levase a cabo, a partir de varas reunións cos clientes, á elaboración da especificación dos requisitos que derivan de ditas reunións.

Esta tarefa do desenvolvemento, de obtención de requisitos, podería ter sido feita unicamente polos desenvolvedores do proxecto, pero tomouse da decisión de realizala conxuntamente co cliente a partir de diversas reunións, partindo sempre dos obxectivos fixados na primeira fase do proxecto, para que por un lado **o cliente sexa parte activa do desenvolvemento**, podendo detectar desta maneira problemas ou elementos que non se axustan a idea que o equipo de desenvolvemento ten sobre o sistema a realizar posibilitando a toma de decisións para que poidan ser solucionado sen ocasionar atrasos e sobrecustos; e por outro lado **que poida aportar novos requisitos** a medida que se van extraendo os relativos aos caso de uso, xa sexa porque non se atopan reflexados directamente nestes, porque non se consideraron no momento da súa definición ou simplemente porque o cliente quixo funcionalidades extra que nun comezo non pensara para o sistema.

Os requisitos obtidos clasifícanse en dous tipos, dependendo de se ofrecen funcionalidades (**requisitos funcionais**) que o sistema debe ter ou se simplemente aportan restricións a considerar na implementación xa sexa en código, condicións de uso, hardware, etc. normalmente sobre a totalidade do sistema (**requisitos non funcionais**).

2.3.1. Requisitos funcionais

A maior parte dos requisitos funcionais obtivéronse de forma directa a partir dos casos de uso definidos anteriormente, e mellorados ou especificados con mais detalle polo cliente e o equipo de desenvolvemento durante as sucesivas reunión, para deste xeito poder ter unha especificación mais elaborada permitindo construír un sistema coa funcionalidade esperada por estes.

Cada requisito segue a plantilla recollida no cadro 2.18 onde se especifica un identificador propio para cada requisito, un nome, unha descrición, unha serie de requisitos aos que se atopa asociado (dependente), as precondicións que se deben dar para poder avaliar o requisito, as secuencias de accións a realizar, tanto a esperada como as alternativas, no caso de habelas, a postcondición que se debe dar ao remate da execución da secuencia de actividades e a importancia do requisito para o cliente. Os requisitos avaliaranse empregando a mesma escala que os casos de uso (vital, importante, desexable) e os criterios de validación que se han de seguir no momento de probar se o requisito se cumpre ou non de forma satisfactoria.

Cadro 2.18: Plantilla para a especificación de requisitos funcionais

ID	RF-XXX
Nome	
Descrición	
Dependencias	
Precondición	
Secuencia normal	
Secuencia alternativa X	
Postcondición	
Importancia	
Validación	

A especificación formal dos requisitos recóllese nos cadros 2.19 a 2.36.

Cadro 2.19: Requisito funcional: Mostrar as KPIs do sistema

ID	RF-001
Nome	Mostrar as KPIs do sistema
Descrición	O sistema debe permitir ao usuario mostrar as KPI que se atopen almacenadas dentro deste.
Dependencias	Ningunha
Precondición	Ningunha
Secuencia normal	1. O usuario accede a parte de administración dentro do Editor de KPI 2. O sistema devolve todas as KPIs.
Postcondición	Recupera a lista con todas as KPI que formen parte do sistema.
Importancia	Vital
Validación	Considerase este requisito satisfeito cando o sistema permita mostrar todas as KPI almacenadas nel.

Cadro 2.20: Requisito funcional: Engadir unha KPI

ID	RF-002
Nome	Engadir unha KPI

Descrición	O sistema debe permitir engadir unha nova KPI dentro deste para que pase a formar parte do conxunto de KPIs existentes nese momento.
Dependencias	RF-001
Precondición	Ningunha
Secuencia normal	1. O usuario accede a parte de administración do sistema. 2. O usuario pulsa sobre o botón para engadir unha nova KPI 3. O usuario introduce os datos básicos (nome, descrición,etc.) dunha KPI. 4. O usuario engade unha formalación da KPI. 5. O usuario engade unha representación a KPI. 6. O usuario preme en finalizar.
Postcondición	O sistema inserta a KPI dentro do sistema
Importancia	Vital
Validación	Considerase este requisito cumprido cando unha vez inserida unha nova KPI esta apareza dentro do sistema con todos os seus datos.

Cadro 2.21: Requisito funcional: Engadir datos básicos

ID	RF-003
Nome	Engadir datos basicos
Descrición	Durante a creación/edición dunha KPI esta conten uns datos básicos como poden ser o seu nome, descrición, palabras clave, etc.
Dependencias	RF-002, RF006
Precondición	O usuario comeza a creación/edición dunha KPI

Secuencia normal	1. O usuario inserta os datos correctamente continua o proceso. 2. O sistema asocia eses datos coa correspondente KPI.
Secuencia alternativa 1	1. O usuario non inserta ou inserta de forma incorrecta algún dos datos básicos. 2. O sistema informa de que dato falta ou esta erroneo.
Postcondición	O sistema asocia eses datos coa KPI
Importancia	Vital
Validación	Considerase este requisito cumprido no momento en que os datos básicos da KPI se asocian a esta.

Cadro 2.22: Requisito funcional: Formular Traballo

ID	RF-004
Nome	Formular traballo
Descrición	Entendese por formular traballo ao código fonte asociado a KPI onde se lle indica que operacións debe realizar esa KPI para obter uns resultados.
Dependencias	RF-002, RF006
Precondición	O usuario comeza a creación/edición dunha KPI
Secuencia normal	1. O usuario engade escribe unha formulación dentro da KPI. 2. O sistema asocia dita formulción a KPI.

Secuencia alternativa 1	1. O usuario compilá a formulación do traballo. 2. O editor mostrara os erros de sintaxe.
Postcondición	O sistema asocia o código de formulación coa KPI.
Importancia	Vital
Validación	Considerase este requisito cumprido cando o sistema asocia dita formulación coa KPI.

Cadro 2.23: Requisito funcional: Engadir Representación

ID	RF-005
Nome	Engadir Representación
Descrición	Permite engadir unha representación asociada a unha KPI.
Dependencias	RF-002, RF006, RF-011
Precondición	O usuario esta creando ou editando unha KPI.
Secuencia normal	1. O usuario preme en engadir representación 2. O usuario escolle unha representación das dispoñibles. 3. O usuario cumprimenta os campos para dar nomes dentro das representacións e as variables para mapear os resultados da KPI a representación. 4. O sistema redirixete a ventá que contén todas as representacións desa KPI

Secuencia alternativa 1	1. O usuario introduce unha representación que xa ten esa KPI. 2. O sistema retorna a ventá que conten todas as representacións da KPI, sen engadila.
Postcondición	O sistema asocia a representación a KPI
Importancia	Vital
Validación	Considerase que se cumpre o requisito cando se asocia unha representación a unha KPI.

Cadro 2.24: Requisito funcional: Editar unha KPI

ID	RF-006
Nome	Editar unha KPI
Descrición	Unha KPI debe poder modificarse, isto implica poder modificar os seus datos básicos, a súa formulación e as súas representacións.
Dependencias	RF-001
Precondición	A KPI ten que existir dentro do sistema
Secuencia normal	1. O usuario escolle dentro da lista de KPIs unha para editar. 2. O usuario edita calquera dato da KPI. 3. O sistema almacena as modificacións e realiza os novos traballos.
Postcondición	O sistema almacena os datos cambiados da KPI.
Importancia	Importante
Validación	Considerase que se cumpre o requisito cando dentro do sistema a KPI se atope modificada logo dun cambio.

Cadro 2.25: Requisito Funcional: Borrar KPI

ID	RF-007
Nome	Borrar KPI

Descrición	Borrar unha KPI refírese a borrar calquera rastro desa KPI dentro do sistema, é dicir eliminar todos os datos tanto os introducidos polo usuario como os xerados polo propio sistema.
Dependencias	RF-001
Precondición	Debe existir KPIs dentro do sistema
Secuencia normal	1. O usuario selecciona o boton de eliminar dunha KPI dentro da lista. 2. O sistema elimina esa KPI do sistema.
Postcondición	O sistema mostra a lista sen a KPI eliminada.
Importancia	Vital
Validación	Considerase o requisito cumprido cando o sistema eliminou unha KPI existente xunto con todos os seus datos.

Cadro 2.26: Requisito funcional: Editar representación

ID	RF-008
Nome	Editar representación
Descrición	Entendese por editar unha representación, cando dentro dunha KPI existe unha ou varias representacións, e se desexa editar as variables de mapeo ou etiquetas desa representación sen modificar o tipo.
Dependencias	RF-002, RF-006
Precondición	O usuario esta editando ou creando unha KPI.
Secuencia normal	1. O usuario esta na ventá de representacións e escolle unha das representacións para editar. 2. O usuario modifica os datos da representación 3. O sistema almacena os cambios asociados de dita KPI.
Postcondición	O sistema modifica a representación escollida da KPI.

Importancia	Importante
Validación	Considerase que se cumpre o requisito cando o sistema cambiou os datos da representación asociada a KPI.

Cadro 2.27: Requisito funcional: Borrar representación

ID	RF-009
Nome	Borrar representación
Descrición	Dentro dunha KPI debese permitir borrar unha das representacións que teña asociadas.
Dependencias	RF-011
Precondición	A KPI debe ter representacións asociadas e o usuario debe estar editando ou creando unha KPI.
Secuencia normal	1. O usuario presiona en eliminar unha representación da lista. 2. O sistema elimina esa representación asociada a KPI.
Postcondición	O sistema elimina a representación asociada a KPI.
Importancia	Vital
Validación	Considerase que se cumpre o requisito cando a representación eliminada non aparece asociada a KPI.

Cadro 2.28: Requisito funcional: Filtrar KPIs

ID	RF-010
Nome	Filtrar KPIs
Descrición	Permite buscar ou delimitar a lista de KPIs en función dun patrón introducido polo usuario.
Dependencias	RF-001
Precondición	Existan KPI dentro do sistema

Secuencia normal	1. O usuario accede a parte de administración de KPIs. 2. O usuario introduce un texto dentro do campo de texto para realizar o filtrado. 3. O sistema devolve a lista de KPIs que se correspondan co patron de texto.
Postcondición	O sistema devolve a lista de KPIs que corresponda coa coincidencia.
Importancia	Desexable
Validación	Considerase que este requisito se cumpre cando a lista de KPIs devoltas se corresponde co patrón de filtrado.

Cadro 2.29: Requisito funcional: Listar representacións

ID	RF-011
Nome	Listar representacións
Descrición	O usuario debe poder ver todas as representacións que formen parte dunha KPI.
Dependencias	RF-001, RF-002, RF-006
Precondición	Debe existir polo menos unha representación
Secuencia normal	1. O usuario esta creando/editando unha KPI e situado na parte de asociación de representacións 2. O sistema devolve todas as representacións que formen parte desa KPI.
Postcondición	O sistema mostra todas as representacións que forma parte da KPI.
Importancia	Importante
Validación	Considerase cumprido o requisito cando ao chegar a parte de representación dunha KPI, este mostra o listado de representacións asociadas a esa KPI.

Cadro 2.30: Requisito funcional: Asistente de xeración de código

ID	RF-012
Nome	Asistente de xeración de código
Descrición	Ademais de crear a formulación para o traballo da KPI manualmente, o sistema debe contar con unha serie de plantillas de diferentes tipos para facilitar a creación dunha formulación de xeito automático.
Dependencias	RF-002
Precondición	O usuario debe estar creando unha KPI e existen KPIs dentro do sistema
Secuencia normal	1. O usuario selecciona unha das plantillas dispoñibles. 2. O usuario cumprimenta os campos solicitados. 3. O sistema debe xerar o código que se mostrara na venta de formulación.
Postcondición	O sistema xera o código mediante a plantilla e permite editalo manualmente.
Importancia	Desexable
Validación	Considerase que se cumpre o requisito cando o sistema xera un código libre de erros sintácticos.

Cadro 2.31: Requisito funcional: Engadir representación KPI ao dashboard

ID	RF-013
Nome	Engadir representación KPI ao Dashboard
Descrición	Para cada usuario, debese permitir engadir de todas as KPIs dispoñibles do sistema, unha ou varias representación para que forme parte do dashboard do usuario.
Dependencias	Ningunha
Precondición	O usuario esta situado no dashboard

Secuencia normal	1. O usuario presiona no boton para engadir unha representación no dashboard. 2. O usuario buscar por nome a KPI. 3. O sistema devolve as KPIs que coincidan co patrón introducido. 4. O usuario escolle unha das KPI mostradas polo sistema. 5. O sistema mostra as represetacións dispoñibles. 6. O usuario escolle a representación. 7. O sistema devolve a ventá do dashboard coa representación escollida, mostrada con datos reais.
Postcondición	O sistema asocia a representación seleccionada co usuario e mostrao por pantalla.
Importancia	Vital
Validación	O requisito cúmprese se ao inserir a representación no dashboard esta aparece correctamente xunto cos seus datos.

Cadro 2.32: Requisito funcional: Eliminar representación KPI do dashboard

ID	RF-014
Nome	Eliminar representación KPI do Dashboard
Descrición	Para cada usuario, este debe ter a posibilidade de eliminar do seu dashboard unha representación que teña debuxada nel.
Dependencias	Ningunha
Precondición	O usuario debe ter representacións no dashboard

Secuencia normal	1. O usuario preme en eliminar dentro da representación. 2. O sistema elimina a representación eliminando a asociación entre a representación e o usuario.
Postcondición	O sistema elimina a representación do dashboard
Importancia	Vital
Validación	Cúmprese o requisito se o sistema elimina a representación do usuario.

Cadro 2.33: Requisito funcional: Mover representación no dashboard

ID	RF-015
Nome	Mover representación no dashboard
Descrición	O usuario, debe ser capaz de mover as representacións seleccionadas dentro do dashboard.
Dependencias	Ningunha
Precondición	Debe haber representacións debuxadas no dashboard
Secuencia normal	1. O usuario situase encima da representación. 2. O usuario fai click sobre él e arrastra a representación até a posición desexada. 3. O sistema cambia a representación de lugar.
Postcondición	A representación cambia de ubicación.
Importancia	Desexable
Validación	Cúmprese o requisito se unha representación do dashboard se pode mover de forma manual e a representación se coloca na posición desexada.

Cadro 2.34: Requisito funcional: Redimensionar representación no dashboard

ID	RF-016
Nome	Redimensionar as representacións no dashboard
Descrición	O usuario, debe ser capaz de redimensionar as representacións dentro do dashboard.
Dependencias	Ningunha
Precondición	Debe haber representacións debuxadas no dashboard
Secuencia normal	1. O usuario situase encima da representación. 2. O usuario fai click na esquina inferior dereita da representación e arrastra a representación sobre a posición desexada. 3. O sistema cambia as dimensións da representación.
Postcondición	A representación cambia de dimensións.
Importancia	Desexable
Validación	Cúmprese o requisito se a representación do dashboard se pode redimensionar de forma manual e a representación obtén as novas dimensións.

Cadro 2.35: Requisito funcional: Gardar cambios do dashboard

ID	RF-017
Nome	Gardar cambios do dashboard
Descrición	O usuario, debe ser capaz de poder almacenar os cambios que executou no dashboard para manter o estado entre sucesivas visitas a interface web.
Dependencias	Ningunha
Precondición	Debe haber representacións debuxadas no dashboard

Secuencia normal	1. O usuario presiona sobre o boton de gardar. 2. O sistema almacena as novas representacións e posicións destas asociandoas ao usuario actual.
Postcondición	Almacenase o novo dashboard dentro do usuario.
Importancia	Desexable
Validación	Cúmprese o requisito se ao recargar a interface web, o dashboard permanece no mesmo estado que cando se gardou.

Cadro 2.36: Requisito funcional: Iniciar sesión

ID	RF-018
Nome	Iniciar sesión
Descrición	O usuario, debe ser capaz de poder acceder o sistema mediante usuario e contrasinal para realizar as demais funcionalidades
Dependencias	Ningunha
Precondición	Debe haber usuarios rexistrados no sistema.
Secuencia normal	1. O usuario accede a interface web e introduce o seu usuario e contrasinal 2. O sistema verifica se os datos son correctos. 3. O sistema levao a vista principal.
Secuencia alternativa 1	1. O usuario introduce datos incorrectos. 2. O sistema informará do erro
Postcondición	O usuario entra no sistema.
Importancia	Importante

Validación	Cúmprese o requisito se ao iniciar sesión cun usuario válido este accede correctamente e se ao facelo cun usuario inválido este e rexeitado.

2.3.2. Requisitos non funcionais

Neste apartado definiranse as restricións que se aplicarán ao proxecto a crear, o que se denomina requisitos non funcionais, que poden incluír parámetros de calidade, rendemento, tecnolxías a empregar, etc. Estes requisitos son extraídos a partir do entorno onde se vai a executar o software e a partir das sucesivas reunións cos clientes.

Para este caso os cadros quedan con menos información se os comparamos coa especificación dos requisitos funcionais. A plantilla empregada para a especificación atopase no cadro 2.37 onde se recolle un identificador, un nome, unha descrición do requisito, a importancia que ten ese requisito para o cliente, e por ultimo o criterio de validación.

Cadro 2.37: Plantilla para a especificación de requisitos non funcionais

ID	RNF-XXX
Nome	
Descrición	
Importancia	
Validación	

A especificación formal dos requisitos non funcionais atópase nos cadros 2.38 a 2.47.

Cadro 2.38: Requisito non funcional: Empregar MongoDB

ID	RNF-001
Nome	Empregar MongoDB
Descrición	O cliente esta interesado en empregar como forma de almacenamento este tipo de base de datos para a elaboración do sistema.
Importancia	Importante
Validación	Este requisito cumprese se todas as bases de datos do sistema están creadas e corren sobre MongoDB

Cadro 2.39: Requisito non funcional: Empregar servizos web

ID	RNF-002
Nome	Empregar servizos web
Descrición	Para acceder desde a interface web aos distinto elementos (Base de datos, Hadoop) e necesario empregar un sistema de comunicación que desacople cada un deles debido a que no momento da posta en producción estes poden estar en maquinas separadas.
Importancia	Vital
Validación	Darase por satisfeito este requisito se as comunicacións entre os distintos módulos se realiza mediante servizos web.

Cadro 2.40: Requisito non funcional: Ferramentas OpenSource

ID	RNF-003
Nome	Ferramentas OpenSource
Descrición	As ferramentas empregadas durante o desenvolvemento do proxecto deben ser gratuitas e de código aberto.
Importancia	Desexable
Validación	Darase por validado este requisito se simplemente todo o desenvolvemento do proxeto se fai so con ferramentas OpenSource.

Cadro 2.41: Requisito non funcional: Tempo máximo de desenvolvemento de 4 meses

ID	RNF-004
Nome	Tempo máximo de desenvolvemento de 4 meses
Descrición	O desenvolvemento do proxecto debe de estar finalizado nun prazo máximo de 4 meses.
Importancia	Vital
Validación	Darase por validado este requisito se o desenvolvemento do proxecto se finaliza antes dos 4 meses.

Cadro 2.42: Requisito non funcional: Aplicación simple e usable

ID	RNF-005
Nome	Aplicación simple e usable
Descrición	O sistema deberá presentar unha interface gráfica que permita ser usada de forma simple e intuitiva evitando que provoque perdidas de tempo para o usuario
Importancia	Importante
Validación	Este requisito cumprese cando a interfaz web pase os test de usabilidade.

Cadro 2.43: Requisito non funcional: Obter os resultados das KPI nun tempo razoable

ID	RNF-006
Nome	Obter os resultados das KPI nun tempo razoable
Descrición	O sistema deberá permitir obter os resultados das KPI e representalos o máis rápido posible.
Importancia	Desexable
Validación	Este requisito cumprirase se os resultados son obtidos e representados nun tempo razoable de 20 segundos.

Cadro 2.44: Requisito non funcional: Documentación técnica simple

ID	RNF-007
Nome	Documentación técnica simple
Descrición	Os manuais tanto de usuario como de instalación deben ser o máis completa e sinxela posible para que usuarios e administradores non invirtan tempo adicional no proceso.
Importancia	Desexable
Validación	Darase por validado este requisito se ambas documentacións son claras e simples para o cliente.

Cadro 2.45: Requisito non funcional: Implementar unha arquitectura módular

ID	RNF-008
Nome	Implementar unha arquitectura módular
Descrición	Para favorecer o incorporación de melloras e o mantemento do software crearase un deseño da arquitectura do sistema modular basado nalgún patrón como pode ser o Modelo-Vista-Controlador (MVC) ou algún derivado.
Importancia	Importante
Validación	Darase por validado este requisito se a interface web está implementada seguindo algún dos patróns mencionados anteriormente.

Cadro 2.46: Requisito non funcional: Empergar Apache Hadoop

ID	RNF-009
Nome	Empregar Apache Hadoop
Descrición	O cliente impón empregar Apache Hadoop como framework para o análise e filtrado de grandes volumes de datos.
Importancia	Importante
Validación	Darase por validado este requisito se o sistema emprega Apache Hadoop para dito uso.

Cadro 2.47: Requisito non funcional: Uso de React e Redux na inferface web

ID	RNF-010
Nome	Uso de React e Redux na interface web
Descrición	O cliente impón o uso da tecnoloxía React e Redux para o desenvolvemento da interface web.
Importancia	Vital
Validación	Darase por cumprido este requisito se simplemente se desenvolve a interface web mediante esta tecnoloxía.

2.3.3. Matriz de trazabilidade

No cadro 2.48 atopase a matriz de trazabilidade dos requisitos funcionais contra os casos de uso, que será de axuda durante o proceso de obtención destes. A finalidade da matriz de trazabilidade é a de asegurarnos de que todos os casos

de uso están cubertos por medio dos requisitos a implementar. É importante que todos os casos de uso teñan asociado polo menos un requisitos.

Cadro 2.48: Matriz de trazabilidade Casos de uso - Requisitos funcionais

	CU-001	CU-002	CU-003	CU-004	CU-005	CU-006	CU-007	CU-008	CU-009	CU-010	CU-011	CU-012	CU-013	CU-014	CU-015
RF-001	X														
RF-002		X													
RF-003		X	X				X								
RF-004		X		X			X								
RF-005		X			X		X	X							
RF-006	X						X								
RF-007	X					X									
RF-008		X					X		X						
RF-009								X		X					
RF-010	X							X			X				
RF-011	X	X					X								
RF-012		X													
RF-013												X			
RF-014													X		
RF-015														X	
RF-016														X	
RF-017															
RF-018															X

Capítulo 3

Xestión do proxecto

A hora de levar a cabo un proxecto informático dunhas dimensións considerables é preciso especificar e definir como se van a levar a cabo certos aspectos do desenvolvemento, entre eles poden ser, xestión dos obxectos creados, modificados ou eliminados ao longo de todo o desenvolvemento e as súas correspondentes versións, o control de custos e riscos, as comunicacións entre os diferentes integrantes e implicados no desenvolvemento do proxecto software.

Como se pode observar, esta parte do proxecto é considerada unha fase chave para conseguir que o proxecto finalice con éxito. Neste capítulo tratarase todos eses aspectos comentados con maior profundidade.

3.1. Alcance

Neste apartado comentaranse as principais seccións presentadas no PMBOK [5] relativos o alcance dun proxecto.

3.1.1. Definición do Alcance

O proxecto a desenvolver permitirá ao usuario poder crear as súas propias *KPIs*, para medir o rendemento daqueles elementos que considere oportuno, e poder visualizar os resultados obtidos para obter unha conclusión a través da información proporcionada.

3.1.2. Entregables do proxecto

A continuación listase o conxunto de entregables ao finalizar o proxecto:

- **Memoria do proxecto** formado polos apartados necesarios para recoller todo o proceso de enxeñaría desenvolto.

- **Manual de usuario** formada polas indicacións necesarias para que o usuario final sexa capaz de utilizar o software sen axuda externa.

- **Manual técnico** deseñado para que calquera usuario poida levar a cabo toda a instalación e configuración necesaria para poñer en funcionamento toda a infraestrutura ou arquitectura desenvolvida.

- **CD** formado por todo o código de desenvolvemento así como os diferentes manuais (citados anteriormente)

3.1.3. Restricións do proxecto

A principal restrición deste proxecto é o tempo de desenvolvemento debido as características do mesmo. Neste senso o número total de horas máximo de dedicación por parte do desenvolvedor é de 420 horas aproximadamente.

3.1.4. Supostos do proxecto

O proxecto desenvolveuse baixo o suposto de que todas as fontes de datos externos como poden ser os servizos web externos, para a obtención de workflows, tarefas e propiedades dun workflow, non varia o esquema de datos devolta en cada chamada, é dicir, o resultado json manten a mesma estrutura de campos (nome dos campos, cantidade de campos por item, etc.), debido a que estes servizos poderían variar o longo do desenvolvemento do proxecto.

3.2. Xestión de Riscos

En calquera proxecto informático de certa envergadura existen elementos que se poden escapar do noso control, xa sexa por erros propios ou de terceiras persoas implicadas no proxecto, como poden ser erros na hora de realizar as estimación no tempo de realización das tarefas, as asuncións falsas no deseño e implementación ou cousas que poden parecer simples como a especificación de requisitos. Estes riscos deberían ser controlados polo equipo de desenvolvemento do proxecto. Pero algúns riscos que se poden producir non poden ser controlados facilmente, como poden ser: Os cambios na lexislación que afecten os tipos de datos que se manexan na aplicación, catástrofes naturais, aparición de competidores, etc.

Que estes riscos aparezan o longo de todo o desenvolvemento do proxecto pode dar lugar a que este se teña que deter e fracasar, por moi boa que fose a idea inicial ou a habilidade do equipo de desenvolvemento, polo que é necesario tomar as medidas oportunas para evitar ou diminuír que os posibles riscos se poidan manifestar.

A continuación realizase un análise dos riscos máis importantes para este proxecto, onde se analizarán as posibles medidas a adoptar para evitalos ou reducir o seu impacto no caso de que se manifesten.

3.2.1. Especificación de riscos

Neste apartado mostrarase a especificación formal dun conxunto de riscos que estará formado polo nome do risco, unha breve descrición, a probabilidade de aparición, o impacto que pode ter no caso de producirse así como o plan levado a cabo para xestionar o risco. Cada risco disporá dun identificador para simplificar, no caso de que fose necesario, a súa referencia.

Tanto para a probabilidade como para o impacto decidiuse elixir unha escala de clasificación formada polo seguintes valores. Na táboa seguinte mostrase unha descrición destes valores para cada caso:

Cadro 3.1: Especificación de riscos: Escala de probabilidade

ESCALA DE PROBABILIDADE	
Alta	Probabilidade estimada de aparición do risco igual ou superior a un 75 %.
Media	Probabilidade estimada de aparición dun risco entre o 25 % e 75 %.
Baixa	Probabilidade estimada de aparición dun risco igual ou inferior a un 25 %.

Cadro 3.2: Especificación de riscos: Escala de impacto

ESCALA DE IMPACTO	
Alta	O impacto provocado polo risco pode provocar a cancelación do proxecto.
Media	O impacto provocado polo risco pode repecutir nunha peor calidade do produto ou provocar modificacións na planificación.
Baixa	O impacto provocado polo risco pode provocar unha lixeira modificación na planificación ou nin sequera afectar en nada á planificación

Unha vez definida a escala utilizada, mostrase a lista de riscos contemplada para a realización deste proxecto:

Cadro 3.3: Risco RSG-01: Baixa dalgún dos titores

ID	RSG-01
Nome do risco	Baixa dalgún dos titores
Descrición	É posible que por algún motivo xa sexa ecónimo, de saúde ou físico, algún dos titores, ou todos, deixen de traballar no proxecto.
Probabilidade de aparición	Baixa
Impacto sobre o proxecto	Medio
Plan de continxencia	No caso de que a baixa sexa dun dos titores, continuarase traballando cos titores restantes. No caso de ser todos, será necesario buscar a outro titor que faga as labores dos titores que están de baixa até finalizalo.

Cadro 3.4: Risco RSG-02: Atraso na planificación

ID	RSG-02
Nome do risco	Atraso na planificación
Descrición	É posible que por algún motivo de inexperiencia por parte do desenvolvedor ou por erros a hora de abordar o proxecto, a planficación deseñada nun primeiro momento, non se axuste a realidade e se produzan pequenos atrasos.
Probabilidade de aparición	Baixa
Impacto sobre o proxecto	Medio
Plan de prevención	Realizar reunións périodicas, sobre todo nas primeiras etapas do deseño e análise, para que o desenvolvedor adquira un coñecemento do ámbito o máis amplo posible.
Plan de continxencia	Redefinir o alcance do proxecto reducindo a carga de traballo e os obxectivos.

Cadro 3.5: Risco RSG-03: Ámbito do proxecto descoñecido

ID	RSG-03
Nome do risco	Ámbito do proxecto descoñecido
Descrición	O tratarse dun proxecto innovador, é posible que o ámbito e o alcance do proxecto sexan descoñecidos.
Probabilidade de aparición	Alta
Impacto sobre o proxecto	Alto
Plan de prevención	Realzación de reunión periódicas, especialmente nas primeiras etapas do desenvolvemento para obter unha definición e unha idea máis completa.

Cadro 3.6: Risco RSG-04: Perda dalgún dos produtos xerados

ID	RSG-04
Nome do risco	Perda dalgún dos produtos xerados
Descrición	A perda dalgún dos produtos xerados durante o proceso de desenvolvemento, entendidos estes como arquivos de código e documentación, poden levar a un atraso da planificación inicial.
Probabilidade de aparición	Baixa
Impacto sobre o proxecto	Alto
Plan de prevención	Para evitar este risco empregranse sistemas de xestión de arquivos que proporcionen historiais de cambios e copias de seguridade automáticas (no caso do código fonte a plataforma empregada será GitHub e o caso do resto de arquivos Google Drive).

Cadro 3.7: Risco RSG-05: Requisitos cambiantes

ID	RSG-05
Nome do risco	Requisitos cambiantes
Descrición	É posible que alguns dos requisitos cambie ao longo do proceso de desenvolvemento do proxecto, tanto por novas necesidades dentro do proxecto como por limitacións atopadas que impiden a súa realización.
Probabilidade de aparición	Alta
Impacto sobre o proxecto	Medio
Plan de prevención	Realizar un deseño modular que permita reducir de forma considerable o impacto sobre o software ao aparecer un cambio

Cadro 3.8: Risco RSG-06: Problemas cos servizos externos

ID	RSG-06
Nome do risco	Problemas cos servizos externos
Descrición	É posible que algún dos servizos externos empregados para o desenvolvemento do sistema, como a obtención de datos dos workflows ou das propiedas, non esten dispoñibles nun momento determinado.
Probabilidade de aparición	Baixa
Impacto sobre o proxecto	Medio
Plan de prevención	Almacenar os datos devoltos polos servizos mantendo a estructura para en caso de fallo ou problema poder usar estes datos para continuar co desenvolvemento.

Cadro 3.9: Risco RSG-07: Enfermidade por parte do desenvolvedor

ID	RSG-07
Nome do risco	Enfermidade por parte do desenvolvedor
Descrición	É posible que o desenvolvedor sufra algunha enfermidade que provoque a súa baixa do proxecto durante unha gran cantidade de días.
Probabilidade de aparición	Baixa
Impacto sobre o proxecto	Alta
Plan de continxencia	Aplazarase o proxecto até a sua futura reincorporación.

Cadro 3.10: Risco RSG-08: Interface pouco usable

ID	RSG-08
Nome do risco	Interface pouco usable
Descrición	É posible que a interfaz web sexa pouco intuitiva para o usuario final impedindo que este quede satisfeito durante a realización das súas tarefas ou que abandone o seu uso debido a súa dificultade de uso.
Probabilidade de aparición	Media
Impacto sobre o proxecto	Alto
Plan de prevención	Antes de finalizar o proxecto, debese facer unha serie de test que permitan detectar fallos de usabilidade.

Cadro 3.11: Risco RSG-09: Mal deseño das probas

ID	RSG-09
Nome do risco	Mal deseño das probas
Descrición	É posible que as probas deseñadas non permitan encontrar ningún erro no software desenvolto.
Probabilidade de aparición	Media
Impacto sobre o proxecto	Medio
Plan de prevención	Crear un plan de probas que permita comprobar que se cumple cada requisito e caso de uso definido.

Cadro 3.12: Risco RSG-10: Fallo hardware

ID	RSG-10
Nome do risco	Fallo hardware
Descrición	É posible que se produzan fallos hardware no equipo no que se esta a desenvolver o proxecto software, imposibilitando a realización do mesmo co consecuente atraso na planificación.
Probabilidade de aparición	Baixa
Impacto sobre o proxecto	Alta
Plan de prevención	Dispoñer de un segundo equipo onde poder continuar coas tarefas a desenvolver e ter sempre unha copia de seguridade de todo o proxecto noutro lugar.

A continuación mostrase a matriz de impacto fronte a probabilidade do conxunto de riscos seleccionados, onde se poderán observar os riscos principais a ter en conta o longo deste proxecto. Estes son os situados na esquina superior esquerda da matriz:

Cadro 3.13: Matriz Impacto - Probabilidade

Impacto/ Probabilidade	**Alta**	**Media**	**Baixa**
Alto	**RSG-03**	**RSG-08**	RSG-04, RSG-07, RSG-10
Medio	**RSG-05**	RSG-09	RSG-06, RSG-02, RSG-01
Baixo			

3.3. Xestión da configuración

Segundo o INCIBE[1] a xestión da configuración [7] é:

> *Un proceso cuxo propósito é establecer e manter a integridade dos produtos de traballo a través da identificación dos elementos/produtos que van a ser a controlados, a definición dun procedemento para o control dos produtos, o rexistro, o informe do estado dos produtos e as auditorías de configuración.*

O xeito máis eficaz para implementar e controlar todo isto pasa pola utilización de ferramentas e sistemas que permitan un correcto control e almacenaxe de versións dos diferentes elementos do proxecto, que permita tanto poder volver atrás dentro das diferentes versións en caso de cometer erros nalgún produto ao ser modificado. A maiores, estas ferramentas permiten un continuo seguimento dos cambios levados sobre os diferentes elementos, permitindo saber cando e quen fixo cada cambio e o que se cambiou con respecto as versións anteriores.

3.3.1. Xestión do código xerado

Para a xestión dos cambios realizados sobre os ficheiros de código fonte do proxecto, empregouse o popular software de control de versións **git**. Este software, nacido en 2005, esta baseado en BitKeeper e Monotone, foi deseñado pola comunidade de desenvolvedores do kernel Linux, en especial polo seu creador Linus Torvalds, permite levar un control de versión de forma rápida e sinxela.

O git pódese empregar de forma local, pero para evitar riscos de poder ter unha perda de información empregouse dito sistema de control de versión xunto coa plataforma web gratuíta GitHub[2], que é unha forxa para aloxar proxectos tanto públicos como privados.

[1] www.incibe.es

[2] https://github.com/

3.3.2. Xestión da documentación

Para xestionar a documentación empregáronse diferentes ferramentas dependendo do tipo de ficheiro do que se estivese a falar. Para os arquivos de texto plano, é dicir, a presente memoria xunto con todos os seus apartados (especificación de requisitos, planificación, etc.), escribíronse empregando o potente software de creación de linguaxe enriquecida LaTeX, que permite a creación de documentos científicos con formatos complexos. A ferramenta empregada para realizar esta memoria foi Overleaf[3], unha aplicación web gratuíta, para calquera usuario, que permite compilar os arquivos en formato latex a PDF de forma rápida, evita todo o proceso de instalación do entorno no sistema do desenvolvedor. Ademais permite a compartición de proxectos, debido a que foi pensada para traballar varias persoas sobre un documento e permite ver unha vista preliminar en todo momento de forma automática.

Por outro lado, para os documentos que non están compostos por texto plano así como as imaxes (diagramas de deseño e planificación, gráficos, gantt, etc.) empregouse a plataforma de almacenamento na nube Google Drive [4], que permite gardar un histórico dos diferentes ficheiros (creación, modificacións, etc.) e ter sincronizados os diferentes ficheiros en varios computadores, ademais empregando esta ferramenta diminúese a probabilidade de que estes se poidan perder por diferentes motivos.

3.3.3. Control dos cambios

Ao ser un único programador o encargado de todo o desenvolvemento do proxecto software, non resulta moi necesario nin eficiente establecer un sistema formal para realizar o control dos cambios, pois será practicamente imposible que existan conflitos entre as distintas versións dun mesmo ficheiro.

No caso dos documentos escritos empregando a plataforma Overleaf, esta permite facer distintas versións do documento, polo que en cada edición do mesmo se creara unha versión indicando brevemente, que cambios se realizaron no documento. Ademais pódense realizar comparacións entre as diferentes versión para ver os cambios que houbo entre elas.

No caso dos ficheiros de código fonte, ao utilizarse un git indicarase en cada commit a tarefa realizada, por outra parte dentro do código fonte poderanse especificar etiquetas en forma de comentarios que indicaran:

- **TODO**, indica que a funcionalidade aínda non esta implementada.
- **CORRIXIR**, indica que a funcionalidade esta realizada pero non é a definitiva, xa sexa a problemas por resolver, faltan tarefas por implementar, etc.

[3]https://www.overleaf.com
[4]https://drive.google.es

- **COMPROBAR**, indica que a implementación actual proporciona a funcionalidade e que podería ser a definitiva, pero que se necesita comprobar se funciona de forma esperada e sen erros.

Unha vez que o cambio se estableza como definitivo, o comentario será eliminado facendo o correspondente commit indicando o que se engadiu ou se eliminou, deste xeito poderase ver en calquera momento as versións anteriores polas que pasou determinada funcionalidade até ser aprobada.

3.4. Selección da metodoloxía

É moi importante definir unha boa metodoloxía de desenvolvemento para levar a cabo o proxecto no tempo planificado conseguindo abordar os obxectivos marcados, citados en apartados anteriores e deste xeito evitar que o proxecto fracase. É necesario realizar unha análise para considerar que ciclo de vida se axusta mellor a este tipo de proxecto.

O proxecto a desenvolver ten un enfoque individual, debido a que so unha persoa desenvolverá o software, aínda así é preciso considerar aos titores do proxecto como clientes e directores do proxecto. Desta forma foi necesario nas reunións iniciais dedicar un pouco de tempo para escoller a metodoloxía óptima. Outro factor a ter en conta é que os requisitos en proxectos deste tipo poden ser cambiantes, en parte porque en etapas iniciais non se sabe exactamente ou simplemente non se entende todo o que abarca o proxecto, novas funcionalidades que non estaban contempladas, etc. Aínda así, e tendo en conta os obxectivos marcados, podemos considerar que o desenvolvemento pode ter marxe de liberdade a hora de encarar o proxecto.

Por outra parte, ademais do citado no parágrafo anterior, búscase un avance rápido para que o cliente poida avaliar canto antes o produto que se está a construír, ademais non se dispoñerá de toda a información sobre os requisitos do proxecto no comezo do desenvolvemento, debido a que algunha das partes ten que ser ben definido polo cliente antes de implementalo. Ademais disto os clientes non estarán dispoñibles para manter reunións constantemente, polo que sobresaturar a estes, pode facer que perdan interese no proxecto. Tendo en conta todo o anterior destacando, o requirimento do avance rápido e a falta de precisión na definición dalgúns requisitos nas fases iniciais lévanos a descartar a maior parte das metodoloxías tradicionais de desenvolvemento e seleccionar unha das coñecidas como **metodoloxiás áxiles**.

Dentro destas metodoloxías destacan dúas: **scrum** e a **programación extrema (XP)**. De estas dúas escolleremos a de programación extrema, debido a que scrum require un número bastante alto de reunións que o cliente non esta disposto a asumir e adicar o seu tempo.

3.4.1. Programación extrema

Este ciclo de vida caracterizase por considerar un proxecto como a liberación de varias versións, até chegar a versión final. Este feito ten vantaxes sobre outras metodoloxías de desenvolvemento:

- Construír un sistema pequeno ten menos ricos que construír un moi grande (problema encontrado no cascada), debido a que o cliente pode ver as diferentes partes xa funcionais antes de terminar todo desenvolvemento e polo tanto pode dar a súa aceptación sobre o resultado que espera.
- Esta metodoloxía é eficiente en proxectos pequeno ou de corta duración como é neste caso, debido a que simplifica o deseño para axilizar o desenvolvemento e permitir obter os produtos en maior brevidade.
- Ao desenvolver en pequenos fragmentos (iteracións), é fácil detectar se os requisitos son correctos. Non é necesario esperar ao final do proxecto (problema atopado en cascada).
- Apoiase na refactorización do código para deste xeito mantelo sempre simple e manexable.

Tendo en conta as vantaxes que supón esta metodoloxía fronte a outras e o ámbito no que se vai a desenvolver este proxecto, é a máis adecuada principalmente debido as dubidas iniciais, polo se debe mostrar o proceso e ter realimenatación dos titores, a inexperiencia do desenvolvedor e sobre todo a limitación de tempo.

3.5. Planificación temporal

Este apartado mostrará a planificación levada a cabo para este proxecto. A primeira fase consiste en identificar os bloques de traballo requeridos para completar o proxecto. Para a súa identificación fíxose uso da EDT (Estrutura de Descomposición de Tarefas). Unha vez realizado este paso, realizase o cronograma do proxecto que ofrece unha representación temporal das tarefas identificadas no diagrama anterior, mostrando máis en detalle os bloques de traballo.

3.5.1. EDT

Unha EDT (Estrutura de Descomposición de Tarefas) é unha ferramenta que consiste na descomposición xerárquica, do traballo a ser executado polo equipo de desenvolvemento, para cumprir os obxectivos do proxecto. Na figura 3.1 pódese observar a EDT xerada para este proxecto. A continuación mostrase a definición de cada paquete de traballo (nodos folla da EDT):

1.1 **Definición do proxecto**.

1.1.1 **Definición de obxectivo e alcance**: Tarefa encargada de definir que é o que se quere construír e para que.

1.1.2 **Análise de casos de uso**: Tarefa encargada de realizar unha análise das actividades que se deberán realizar no sistema.

1.1.3 **Análise de requisitos**: Tarefa mediante a cal se realiza unha análise do problema para obter os requisitos necesarios para o correcto funcionamento deste.

1.1.4 **Análise de riscos**: Tarefa encargada de obter, analizar e dar unha solución aos riscos que poderían facer que o proxecto fracase.

1.1.5 **Análise de alternativas**: Tarefa encargada de estudar as mellores solucións e ferramentas dispoñibles actualmente que se adecúen a este proxecto software.

1.2 **Deseño de alto nivel do sistema**: Paquete que consiste no deseño da arquitectura xeral do sistema a desenvolver

1.3 **Interface web**

1.3.1 **Deseño da arquitectura**: Tarefa encargada do deseño da arquitectura mediante a cal se construirá o sistema.

1.3.2 **Deseño lóxico**: Tarefa encargada do deseño lóxico do sistema.

1.3.3 **Deseño do aspecto gráfico**: Tarefa mediante a cal se dará un aspecto a interface web para que sexa sinxela de manexar e agradable.

1.3.4 **Implementación**: Tarefa encargada de levar o deseño a unha linguaxe de programación.

1.3.5 **Probas unitarias**: Tarefa que consta dun conxunto de probas para asegurar que o software responde como debe.

1.4 **Servizos**

1.4.1 **Tratamento dos datos**: Tarefa encargada de producir os servizos para ter acceso e tratamento dos datos necesarios para o correcto funcionamento do sistema.

1.4.2 **Hadoop**: Tarefa encargada de xerar os servizos que comuniquen o software a desenvolver coa plataforma Hadoop.

1.4.3 **Probas unitarias**: Tarefa que consta dun conxunto de probas para asegurar que os servizos responden como se esperara.

1.4.4 **Integración coa interface web**: Tarefa que consiste na incorporación dos servizos coa interface web.

1.4.5 **Probas de Integración**: Tarefa que consta dun conxunto de probas para asegurar que os servizos xunto coa interface web funcionan como se espera.

1.5 **Xestión**: Paquete formado por un conxunto de tarefas necesarias para poder supervisar o correcto desenvolvemento do TFG.

1.6 **Documentación**: Tarefa que consiste na elaboración da memoria final do TFG así como a documentación relativa a instalación do software e uso deste.

1.7 **Peche do proxecto**: Tarefa na cal se realizan os pasos parar dar por finalizado o desenvolvemento do proxecto.

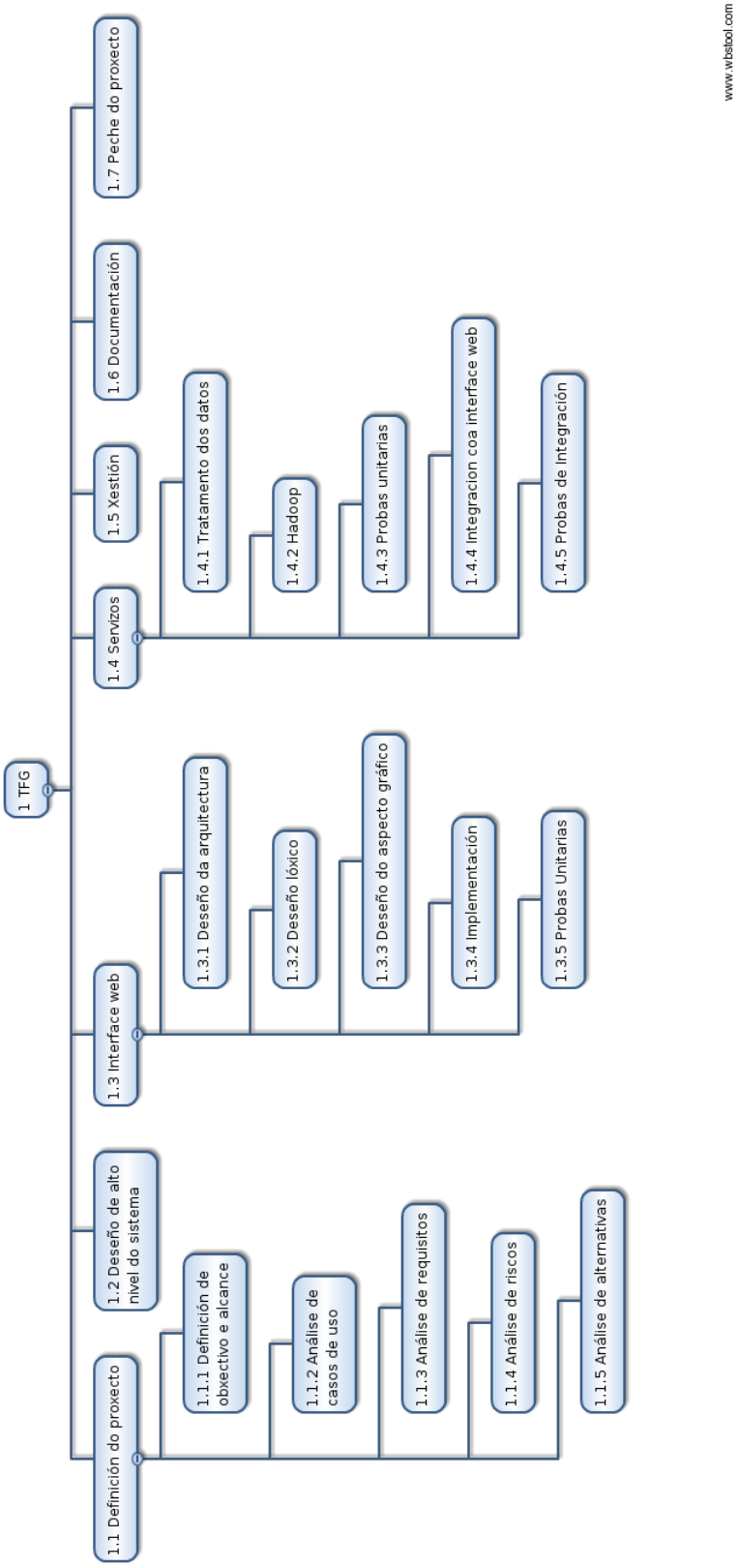

Figura 3.1: EDT do proxecto

3.5.2. Diagrama de Gantt

A continuación inclúese a distribución temporal estimada para o desenvolvemento do proxecto. En primeiro lugar, na figura 3.2 inclúese un cronograma coas iteracións principais nas que se divide o desenvolvemento do proxecto. En paralelo a todas estas levarase a cabo unha tarefa de documentación necesaria para amosar todo o proceso de desenvolvemento de forma fiable e revisable. Tamén se poden ver outras tarefas de xestión que consisten en verificar que ningún risco se produza ou os cambios que se realicen no proxecto se fagan de forma eficiente e eficaz. Para rematar o proxecto realizase un proceso de peche deste e conclúe coa súa impresión e entrega.

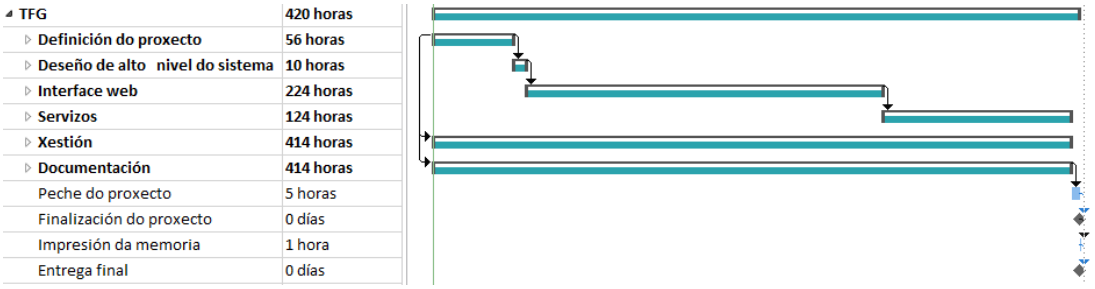

Figura 3.2: Cronograma xeral do proxecto

A primeira fase do proxecto é a etapa de definición, cunha duración de 56 horas, distribuídas como se pode observar na figura 3.3. Seguida dunha primeira fase de deseño da arquitectura do sistema, cunha duración de 10 horas (figura 3.4). Estas dúas primeiras fases constitúen a fase de inicialización do proxecto.

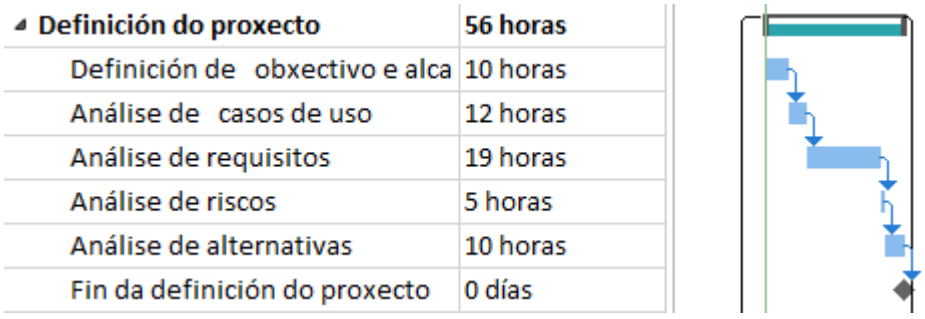

Figura 3.3: Cronograma da fase de definición do proxecto

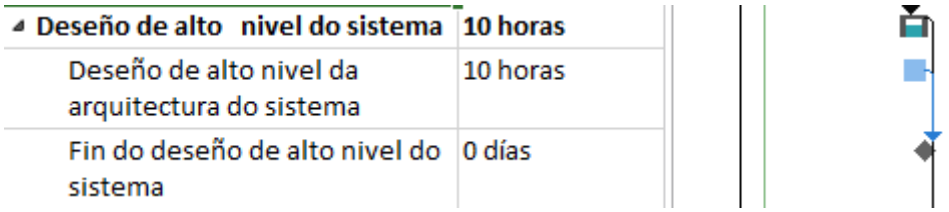

Figura 3.4: Cronograma da fase de deseño da arquitectura de alto nivel do sistema

A continuación da fase de inicialización comezan as fases con maior carga de traballo, ao proceder coa implementación da interface web, tarefa representada na figura 3.5 cunha duración de 224 horas.

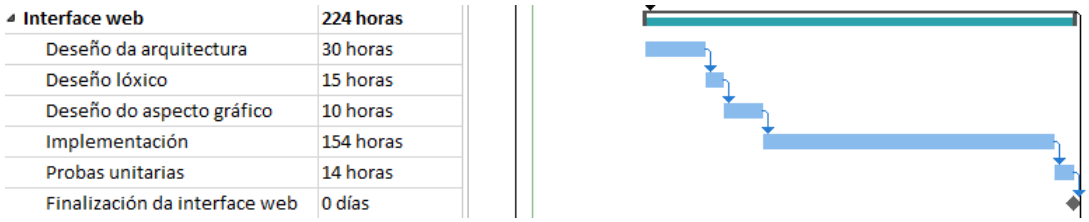

Figura 3.5: Cronograma da fase de implementación da interface web do proxecto

Tras finalizar a implementación da interface web, procedese a implementar os diferentes módulos ou servizos que terán acceso aos datos que necesita a interface web así como a súa integración con Apache Hadoop para realizar os traballos oportunos (figura 3.6).

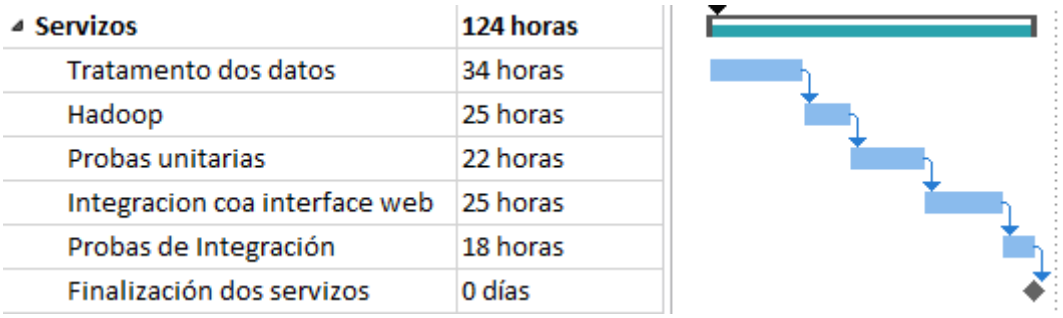

Figura 3.6: Cronograma da fase de implementación dos servizos do proxecto

Finalmente atópanse as tarefas encargadas de levar todo o proceso de xestión de control de cambios, control de riscos e seguemento do plan (figura 3.7) e a de elaborar a presente memoria (figura 3.7):

Figura 3.7: Cronograma da fase de xestión do proxecto

Figura 3.8: Cronograma da fase de documentación do proxecto

3.6. Análise de Custos

É recomendable realizar un análise de custos a pesar de ser un proxecto académico debido a que diversos informes xustifican que a maioría dos proxectos informáticos que fracasan debese a sobrecustos no desenvolvemento e con frecuencia os que finalizan sobrepasan o presupostado de forma alarmante [9] [11].

Para tratar este problema os custos califícanse en custos **directos**, sendo estes os custos que se produciran de forma constante o longo do proxecto, e os custos **indirectos** que son repartidos entre diferentes proxectos e que teñen unha influencia tanxencial no desenvolvemento, como pode ser a auga, o material de oficina, a luz, etc. Cabe destacar que os valores dos custos, que se mostran nas táboas das seccións posteriores están en euros.

3.6.1. Custos directos

Dentro deste apartado recóllense os custos que se poden atribuír facilmente ao proxecto, como poden ser os soldos dos desenvolvedores ou os equipos necesarios para a execución do sistema a desenvolver. Para este proxecto, ao non ser necesario adquirir ningún equipamento novo, os únicos custos directos son os relacionados cos recursos humanos e cos soportes (documentación e código).

O equipo de desenvolvemento esta composto so por un rol de analista-programador júnior (A.P). Os directores do proxecto considerarase que non teñen ánimo de lucro neste proxecto debido a que é un proxecto académico.

Por outra parte pódense cargar ao presuposto deste proxecto o custo da impresión en papel de 4 copias desta memoria e de 4 CDs para a entrega dunha versión dixital xunto con todo o código desenvolto da aplicación.

Cadro 3.14: Custos directos do proxecto destinados a salarios

	Bruto	**Seg. Soc. (30 %)**	**Total anual**	**Total Mensual**	**Custo/hora**
A.P	16.000	4.800	20.800	1.485,71	10,40

Cadro 3.15: Custos do proxecto destinado a materiais

	Uds.	**Custo/ud.**	**Total**
Copias impresas da memoria	4	25	100
CDs	4	0,30	1,20

No cadro 3.14 preséntase os custos do persoal que esta relacionado co proxecto. A referencia destes salarios foi adquirida a través do *Estudio salarial do sector TIC en Galicia 2015-2016* [10]. Tamén se inclúe neste cálculo unha porcentaxe estimada do 30 % destinada a Seguridade Social, para cubrir accidentes de traballo, enfermidades, desemprego, etc. No cadro 3.15 pódense observar os custos directos relacionados coa distribución do sistema e documentación (copias impresas da memoria e CDs). Finalmente no cadro 3.16 un resumo dos custos directos estimados para o proxecto. Para todas as táboas os importes atópanse en euros.

Cadro 3.16: Resumo dos custos directos do proxecto

	Importe
Soldo A.P (custe por hora x número de horas)	4368,20
Gastos de materiais	101,20
TOTAL	**4469.40**

3.6.2. Custos indirectos

Como se comentou na introdución deste apartado, os custos indirectos son os que non son atribuíbles directamente o proxecto, senón que se comparten entre varios proxectos e polo tanto teñen un impacto tanxencial no custo total deste. Entre estes destacan a amortización dos equipos de desenvolvemento. Neste caso o único equipo empregado foi un computador cun custo de **900 €** onde se estima unha durabilidade de **5 anos**. O cálculo do custo do equipo atribuíble ao proxecto reflíctese no cadro 3.17

Cadro 3.17: Amortización do equipo de desenvolvemento

Custo total	**Amortización/hora**	**Horas de uso**	**Custo atribuíble ao proxecto**
900	0,1	420	42

Por outra banda non so temos o custo dos equipos como custos indirectos se non que tamén dentro dese equipo temos que contar os produtos software que se empreguen para o desenvolvemento do proxecto, como as licenzas, aínda que existen licenzas para usos académicos, nun proxecto real non se póderia facer

uso delas polo que tamén se incluirán. Os custos para o software empregado atopase desglosado no cadro 3.18. Ao igual que na táboa 3.17, estímase un tempo de amortización para as aplicacións empregadas, pero neste caso dependerá da funcionalidade do software en cada caso.

Cadro 3.18: Custos do software empregado no desenvolvemento aplicable ao proxecto

Aplicativo	**Custo total**	**Horas até amortizalo**	**Custó por hora**	**Custo atribuible ao proxecto**
Sublime Text	63	2.000	0,03	3
Microsoft Project	769	2.000	0,38	4
StarUML	60	1.000	0,06	3
Highcharts	350	900	0,38	4
Netbeans	0	-	-	-
MongoDB	0	-	-	-
Parabola GNU/Linux-Libre	0	-	-	-
SBT	0	-	-	-
Scalding	0	-	-	-
React	0	-	-	-
Material-UI	0	-	-	-

Para finalizar, os últimos custos indirectos que se terán en conta son os derivados dos servizos, necesarios para levar a cabo a tarefa de desenvolvemento (auga, luz, internet...). Estes estimaranse como un 15 % do total, estimado á súa vez no cadro 3.19. No cadro 3.20 presentase un resumo dos custos indirectos atríbuibles ao proxecto.

Cadro 3.19: Custos totais asociados aos servizos básicos do lugar de traballo

	Custo mensual(estimado)	**Atribuíble ao proxecto**
Luz	50	20
Auga	30	11,50
Gas	20	8
Telefono	30	11,50
Internet	40	15,50
Cafe	30	11,50

Cadro 3.20: Resumo dos custos indirectos atribuibles ao proxecto

	Importe
Amortización equipos desenvolvemento	42
Software	14
Servizos básicos	78
TOTAL	**134**

3.6.3. Custos totais

Para calcular os custos totais debemos acumular os custos directos e indirectos. Por outra banda debido á inexperiencia do desenvolvedor, decídese reservar un 15 % do custo total deste para posibles problemas que poidan ocorrer durante o desenvolvemento. O desglose atópase recollido no cadro 3.21.

Cadro 3.21: Resumo dos custos totais

	Importe
Gastos directos	4469,4
Gastos indirectos	130
Bolsa para posibles problemas (+15 % do total)	689,91
TOTAL	**720,99**

3.7. Xestión das comunicacións

Un dos factores clave, para que o desenvolvemento do proxecto software sexa exitoso, é unha boa xestión das comunicacións entre os diferentes membros implicados no proxecto no caso de estar formado por máis de un. Para iso a continuación defínese un plan de xestión das comunicacións no que se establecen os protocolos e medios de comunicación a empregar nas comunicacións entre eles, para que se realicen dun xeito o máis eficaz e eficiente posible.

3.7.1. Identificación dos interesados

Os interesados ou stakeholders [5] dentro dun proxecto software, segundo o PMBOK, son:

> *Os interesados no proxecto son as persoas e organizacións que participan de forma activa no proxecto ou cuxos intereses poden verse afectados como resultado da execución do proxecto ou da súa conclusión. Un interesado pode xestionar a súa influencia en relación cos requisitos para asegurar un proxecto exitoso.*

A identificación de todos os interesados no proxecto resulta de vital importancia para ter en conta os requisitos e restricións de todos eles á hora de definir un proxecto. No cadro 3.22, atópanse recollidos os datos identificativos dos interesados no proxecto. No cadro 3.23, aparecen recollidos os principais intereses e funcións dos interesados identificados, e no cadro 3.24 a clasificación destes segundo a súa procedencia e postura de cara o proxecto. Unha postura de apoio indica que o interesado fará todo o posible para que o proxecto se leve a cabo con éxito, mentres que unha postura de oposición ao proxecto indica que o interesado fará todo o posible para que o proxecto fracase. Unha postura neutral, indica que non lle importa se o proxecto finaliza de forma satisfactoria ou fracasa.

Cadro 3.22: Identificación dos interesados no proxecto

ID	**Nome**	**Rol**	**Localización**
IN-001	Manuel Lama Penín Juan C. Vidal Aguiar Victor J. Gallego Fontenla	Titores do proxecto / Clientes	Campus Vida, Santiago de Compostela
IN-002	Jorge López Seijas	Desenvolvedor	Campus Vida, Santiago de Compostela
IN-003	Usuarios	-	-

Cadro 3.23: Intereses e funcións dos interesados

ID	**Función**	**Expectativas**	**Influencia**	**Fase de maior interese**
IN-001	Resolver dúbidas e proporcionar axuda durante o proceso de desenvolvemento do proxecto	Finalización exitosa do proxecto	Elevada	Todo o proxecto
IN-002	Desenvolver todas as fases do proxecto	Finalización exitosa do proxecto	Elevada	Todo o proxecto
IN-003	Testear o correcto funcionamento do sistema xerado	Obtención dun sistema completa-mente funcional e usable	Media	Probas

Cadro 3.24: Clasificación dos interesados segundo a súa orixe e nivel de apoio ao proxecto

ID	Orixe	Nivel de apoio ao proxecto
IN-001	Externo	Apoio
IN-002	Interno	Apoio
IN-003	Externo	Neutral

3.7.2. Planificación das comunicacións

Para que todas as comunicacións sexan efectivas e evitar perdidas de tempo e distraccións innecesarias, no cadro 3.25 defínense os obxectivos das comunicacións, os medios a empregar e os interesados implicados. Como se indica no cadro mencionado anteriormente, levaranse a cabo unha serie de reunións cos directores e clientes durante todo o proceso de desenvolvemento do proxecto. Estas reunións realizaranse no edificio CITIUS, lugar onde traballan, sito no Campus Vida da Universidade de Santiago de Compostela, nunha sala reservada previamente para tal fin. Non se define cada canto tempo terán lugar as reunións senón que se poderán convocar cando se considere oportuno e necesario. Sen embargo, si que se espera unha frecuencia máis elevada nas fases iniciais do desenvolvemento (definición de obxectivos, alcance e requisitos) e deseño do proxecto que no resto de fases.

Cadro 3.25: Matriz de planicación das comunicacións

IDs	Obxectivo	Nivel de detalle	Tipo de información	Idioma	Formato
IN-001 ↔ IN-002	Coñecer a evolución do proxecto, resolver dúbidas e proporcionar os requisitos	Alto	Cronogramas. Técnica. Descritiva.	Galego, Castelán	Correo electrónico, Reunións
IN-002 ↔ IN-003	Proporcionar información sobre a interface gráfica, usabilidade e posibles melloras.	Medio	Descritiva	Galego, Castelán.	Reunións.

Capítulo 4

Análise das tecnoloxías, ferramentas e estratexias

Neste capítulo farase unha análise das tecnoloxías dispoñibles actualmente, as ferramentas empregadas para realizar o desenvolvemento e as posibles estratexias para a implementación do sistema.

Para cada unhas das sección propóñense unha serie de alternativas coas súas vantaxes e desvantaxes, que se deben analizar para escoller a opción que mellor se adapte permitindo alcanzar os obxectivos do proxecto.

4.1. Servizos Web

Cando se precisa crear unha aplicación onde se precisa traballar cunha arquitectura distribuída o equipo de desenvolvemento pode escoller entre multitude de tecnoloxías: paso de mensaxes, aplicacións de memoria compartida, chamadas a procedementos remotos, etc.

Co incremento da popularidade das tecnoloxías web creouse un novo sistema para este tipo de arquitecturas chamadas, **servizos web**. Un servizo web é unha nova forma de acceder aos procedementos remotos das aplicacións empregando un conxunto de protocolos e estándares que cubran un gran abano de posibilidades cando se precisa construír unha arquitectura distribuida.

No proxecto a desenvolver, todas as comunicacións entre os diferentes módulos (base de datos, hadoop, etc.) realizaranse empregando este tipo de arquitecturas. Para realizar a implementación dos servizos web disponse de varias alternativas, pero entre elas as máis destacables e as máis usadas son as arquitecturas SOAP ou REST.

4.1.1. Servizo web SOAP

A idea principal de SOAP (Simple Object Access Protocol) basease en proporcionar un protocolo estándar para a creación de servizos web a través do uso de diferentes tecnoloxías.

Por un lado, defínese mediante a utilización da linguaxe declarativa XML a estrutura das mensaxes. Ademais disponse da linguaxe WSDL, tamén baseada en XML, que ten o cometido de definir as diferentes interfaces a empregar.

SOAP é un protocolo stateless, polo que non mantén o estado do sistema. Se se precisa coñecer o estado do sistema é necesario empregar outros recursos adicionais. No cadro 4.1 amósanse as vantaxes e desvantaxes desta tecnoloxía:

Cadro 4.1: Vantaxes e devantaxes do protocolo SOAP

Vantaxes	***Desvantaxes***
- Permite a **definición de interfaces** que proporcionan os servizos ao utilizar WSDL	- Complexidade das mensaxes enviadas e da súa definición
- Maior **seguridade** no envío e recepción de datos ao ter que encapsular e desencapsular	- Dependencia do XML como única linguaxe para definir as mensaxes
- Control da **calidade do servizo**	- O tamaño das mensaxes é significativo
- **Non depende** de protocolos nin de implementacións	

4.1.2. Servizos web REST

No caso de REST, este non define un protocolo, senon unha arquitectura de servizos web baseada en protocolos estándares e moi estendidos. Os servizos web REST teñen as seguintes características:

1. Os recursos dispoñibles son identificados mediante **URIs** (Identificador de recursos uniforme), que proporciona un espazo de nomes inequívoco por recurso.

2. Emprega unha interface común, xa que ao empregarse o protocolo HTTP este dispón, unicamente de 4 métodos (GET para recuperar datos, PUT para modificalos, DELETE para eliminalos e POST para crealos).

3. Como HTTP non mantén o estado, podemos afirmar que REST tampouco mantén o estado e polo tanto é stateless, polo que en cada petición se deben mandar os datos necesarios para que o servidor o poida tratar.

O cadro 4.2 recolle unha lista de vantaxes e desvantaxe deste tipo de servizos:

Cadro 4.2: Vantaxes e desvantaxes dos servizos REST

Vantaxes	***Desvantaxes***
- Simplicidade, ao empregar protocolos lonxevos e simples en materia web (HTTP, XML...)	- Dificultade para saber o que realiza cada operación ao utilizar sempre a mesma interface (GET, PUT, POST, DELETE), ademais de non existir ningunha esixencia de que método hai que usar en cada caso.
- Todo os servizos web poden ser probados directamente desde o navegador web, o que facilita as comprobacións de funcionamento.	- Dependencia do protocolo HTTP.
- Permite multitude de mensaxes para definir os datos (JSON, XML, CSV...)	- Imposible implementar comunicacións seguras directamente debido a que HTTP é un protocolo inseguro polo que se debe facer mediante outros protocolos.

4.1.3. Conclusión

Observando detidamente as vantaxes e desvantaxes de ámbolos dous tipos de servizos, tendo en conta en todo momento o contexto no que se vai a desenvolver o proxecto, chegase a conclusión de que a arquitectura REST é a ideal para realizar dito desenvolvemento. Os motivos que levaron a tal escolla foron os seguintes:

- Como a comunicación entre o cliente e o sistema vai ser a través dunha interface web a través de **AJAX** é máis sinxelo realizar esta comunicación empregado servizos REST a través de URIs para chamar os diferentes métodos. Ademais ao tratar con datos en formato JSON este facilita moito a labor de manipulalos dentro da aplicación web.

- Algúns servizos externos necesarios para levar a cabo dito desenvolvemento xa estaban creados usando unha arquitectura REST polo que é importante manter a uniformidade.
- Como programar servizos REST é máis doado debido a facilidade para testealo, o formato das mensaxes, etc. foi recomendado para o programador debido a súa inexperiencia con este tipo de tecnoloxías.

4.2. Patróns de deseño

Os patróns de deseño consisten en elementos de deseño que foron propostas por expertos na materia e probados durante un longo período de tempo, que aportan unha solución sinxela sobre problemas comúns que foron vistos ao longo da historia do desenvolvemento software. Nos apartados seguintes móstranse os patróns de deseño que se utilizaron para o deseño deste proxecto.

4.2.1. Patrón singleton

A idea que existe detrás deste patrón e moi sinxela; consiste en asegurar que nunca existe máis dunha instancia dunha clase que o implemente en memoria. Na figura 4.1 pódese observar a súa representación en UML:

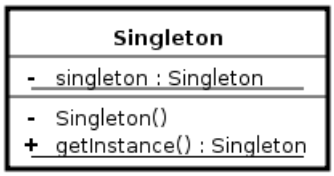

Figura 4.1: Diagrama do patrón de deseño: Singleton

Este patrón será usado no módulo de xestión de datos, para garantir que non existan dúas instancias dun obxecto que conteña a conexión coa base de datos.

4.2.2. Patrón template method

Este patrón nace da necesidade de extender determinados comportamentos dentro dun mesmo algoritmo por parte de diferentes entidades, é dicir, diferentes entidades teñen un comportamento similar pero que difiren en determinados aspectos puntuais da entidade concreta. Unha posible solución sería copiar o algoritmo en cada entidade cambiando a parte concreta pola que difiren. Esta solución tería a desvantaxe de que existe código duplicado. A solución pasa por abstraer todo o comportamento que comparten as entidades nunha clase (abstracta).

Esta clase conterá o esqueleto do algoritmo e delegará determinadas responsabilidades nas clases fillas, mediante un ou varios métodos abstactos que deberán implementar. Na figura 4.2 pódese observar o deseño de clases:

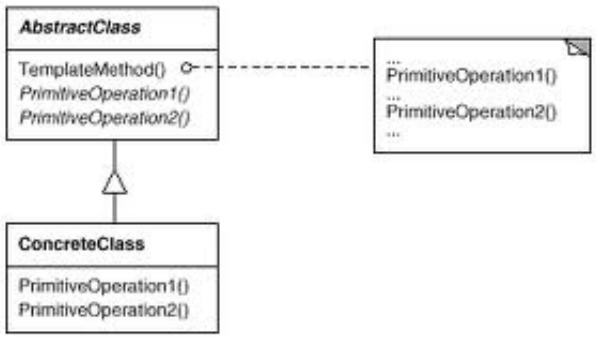

Figura 4.2: Diagrama do patrón de deseño: Template Method

Este patrón será empregado na interface web para o mapeo das variables nas correspondentes representacións, debido a que este mapeo segue sempre unha implementación base para obter datos e varia lixeiramente en función do tipo de representación.

4.2.3. Patrón observer

A implementación deste patrón ten a finalidade de servir como método probado de notificación de cambios entre compoñentes. Para isto, o obxecto observado debe implementar unha serie de métodos que se encarguen de notificar aos observadores cando este é modificado, co obxectivo de que estes coñezan os cambios.

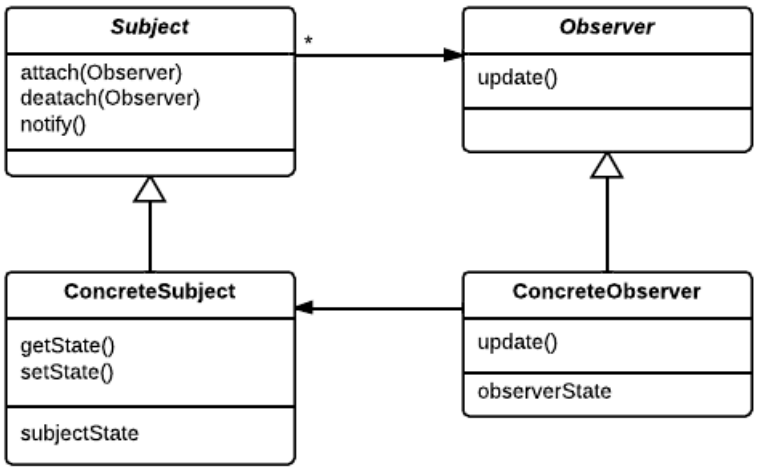

Figura 4.3: Diagrama do patrón de deseño: Observer

No caso deste proxecto, o patrón *observer* ven implementado tanto polo DOM como polo framework de desenvolvemento empregado para a creación da interface web, e permite realizar a captura e tratamento de eventos e notificacións. O diagrama correspondente con este patrón atópase na figura 4.3

4.2.4. Data Transfer Object

O Data Transfer Object, DTO é un obxecto que transporta datos entre procesos. O seu uso ten especial interese cando se realiza unha comunicación entre procesos mediante interfaces remotas onde a chamada é unha operación custosa.

Os DTOs son obxectos simples que non deben conter máis lóxica que a de almacenar e entregar os seus propios datos, deste xeito aforrase custos a hora de encapsular, desencapsular e ancho de banda xa que o seu tamaño e menor que o obxecto orixinal. Dentro deste proxecto, será empregado nas comunicacións entre cos servizos web, para deste xeito delimitar o número de datos a enviar e recibir.

4.3. Modelos de arquitectura

Neste apartado comentaranse os modelos de arquitectura empregados para o deseño da parte da interface web.

4.3.1. Modelo-Vista-Controlador (MVC)

A arquitectura MVC, pretende separar o sistema en tres partes. Por un lado atópanse os **modelos**, que son os encargados de ter toda a lóxica de negocio. Por outra parte temos as **vistas**, que son as encargadas de representar os datos de forma amigable para os usuarios e para enlazar estes dous compoñentes temos os controladores, que enrutan as vistas en función do que se teña que mostrar:

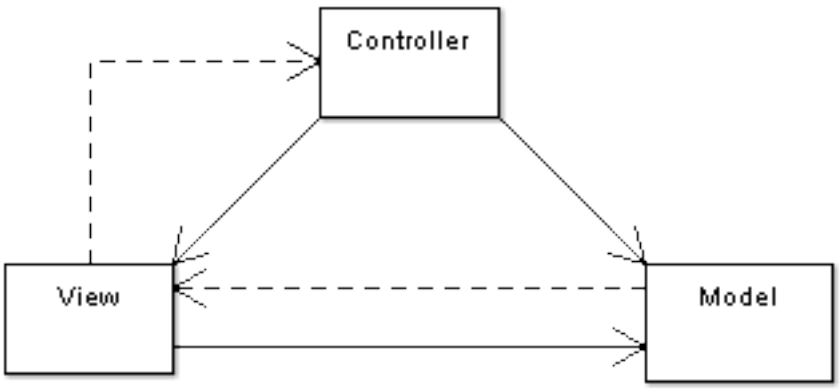

Figura 4.4: Diagrama de exemplo de MVC

4.3.2. Modelo-Vista-Modelo de Vista (MVVM)

Con este enfoque, prescíndese de controladores para pasar a crear **modelos de vista**, que implementan todo o estado das vistas e enlazan os datos obtidos dos modelos con estas. Ademais, os modelos son independentes, polo que non teñen constancia da interface, senon que son as vistas as que teñen constancia dos modelos de vista, optando así por un enfoque algo diferente, ao anterior, no que o controlador sempre sabia da interface.

4.3.3. Modelo-Vista-Controlador-Modelo de vista (MVCVM)

Por último, este enfoque resulta ser un híbrido resultado de mesturar os puntos fortes dos modelos de arquitectura MVC e MVVM. A descomposición básica ideal desta arquitectura sería a seguinte:

- As vistas encarganse de representar os datos, sen saber de onde proceden ou como se obtiveron.
- Os modelos da vista conteñen unha representación dos datos a presentar na interface, asi como a implementación da lóxica relativa ao comportamento desta, pero seguen sen coñecer a procedencia dos datos.
- Os modelos conteñen a representación real dos datos e os métodos de acceso a estes.
- Os controladores encárganse de decidir e enrutar a vista que se debe amosar en cada momento.

4.3.4. Flux

Flux [8] é unha arquitectura proposta por Facebook que proporciona unha solución os cambios de datos dentro do modelo forzando a que estes se leven a cabo un a un e nunha única dirección, solucionando o problema do patrón MVC que non se consegue saber en que orde se producen as actualizacións dentro do modelo.

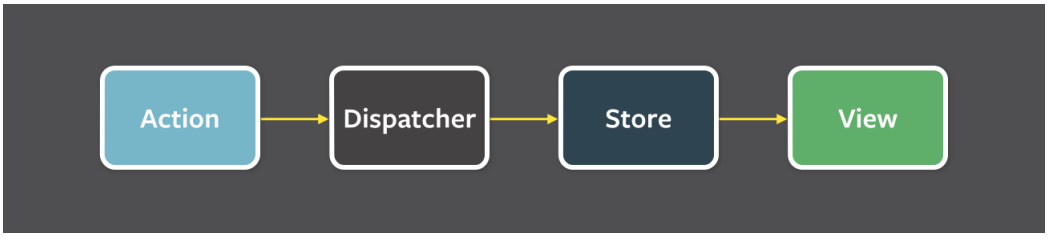

Figura 4.5: Diagrama do funcionamento de Flux

Como se pode observar na figura 4.5 cando se vai a producir un cambio sobre o **Store** a dirección do fluxo de datos vai nunha soa dirección, deste xeito pódese saber en cada momento quen levou a cabo a acción e sobre que se levou a cabo ao ser o almacén de datos (store) único. Dentro deste proxecto será empregado mediante Redux para actualizar os datos dentro da interface web.

4.3.5. Conclusión

Tras analizar as alternativas dispoñibles para a definición da arquitectura da interface web, decidiuse empregar unha arquitectura MVCVM, xa que nos aporta a simplicidade do modelo MVC xunto coa posibilidade de implementar un enlazado inmediato dos datos do modelo da vista coa propia vista sen complicar moito o deseño coñecido da aplicación.

Cabe destacar que para a parte do modelo, como se comentou no subapartado anterior, seguirase unha arquitectura de tipo flux, mediante Redux, que proporcionara que as accións se leven a cabo nunha soa dirección.

4.4. Tecnoloxías

Neste apartado trataranse e introduciranse dun xeito breve e explicativo as tecnoloxías e linguaxes empregadas no desenvolvemento do sistema, tanto para a parte de interacción web como para as demais partes do sistema.

4.4.1. HTML5

HTML (Linguaxe de marcas de hipertexto), que define unha estrutura básica e un código para a definición do contido dunha páxina web. Esta desenvolvido pola W3C[1] e pola WHATWG[2]. HTML5 é a última especificación desta linguaxe para a realización deste traballo, que incorpora moitas diferenzas con respecto as versións anteriores, como por exemplo novas etiquetas que aportan maior valor semántico ao código e polo tanto máis comprensible a hora de léelo, tales como *header* para incluír encabezados, *nav* para a barra de navegación, *article* para os articulos ou o *footer* para indicar os pés de páxina, mentres que tamén se engadirón atributos as etiquetas que xa existían e se simplificaron outras.

4.4.2. CCS3

CSS (Folla de estilo en cascada) é unha linguaxe de formato aberto para definir e crear a presentación ou deseño dun documento estruturado como pode ser HTML ou XML. O encargado de formular e crear as especificacións é ,como no

[1] https://www.w3.org/
[2] https://whatwg.org/

caso do HTML, a W3C. A principal idea é a de separar o que é a estrutura definida do documento da súa representación permitindo realizar varias representacións para unha mesma estrutura (móbil, tv...).

A diferenza coas versións anteriores, CSS3 foi a única especificación que definía varias funcionalidades e que esta dividida en varios documentos chamados módulos. Cada módulo engade novas funcionalidades sobre as especificacións anteriores, deste xeito permite a compatibilidade coas especificacións anteriores. Na figura 4.6 pódese ver a evolución ao longo do tempo e a inclusión de módulos para amplialo, obtido do sitio web da fundación Mozilla[3], onde se pode ver como continua a evolución deste.

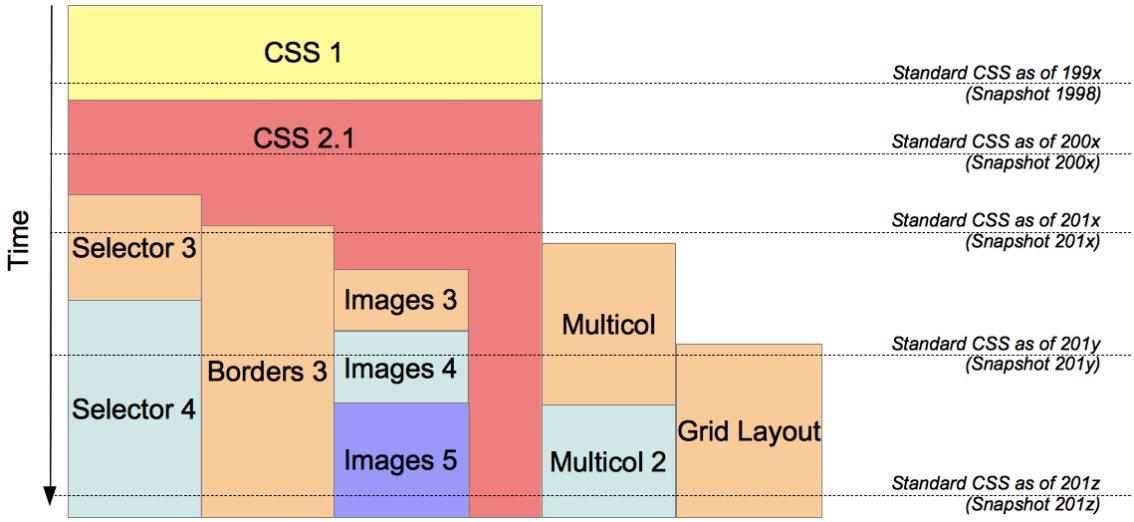

Figura 4.6: Evolución do estándar CSS

4.4.3. JavaScript

Ademais de HTML para estruturar o documento e CSS para dar estilo, temos tamén JavaScript que constitúe gran parte do desenvolvemento deste proxecto. É unha linguaxe interpretada que que permite construír sitios web dínamicos. Ademais conta cun tipado débil e ten bastante influencia de linguaxes do tipo de C e Java.

Naceu a mediados da década dos 90 da man de Netscape Communications Corp. é unha linguaxe baseado no estándar ECMAScript que se emprega no lado do cliente, aínda que existen outras alternativas, tanto por parte de Microsoft (JScript) como de Adobe (ActionScript).

Actualmente JavaScript segue o estándar ECMAScript 6[4][5] recentemente novo

[3]https://developer.mozilla.org/es/docs/Web/CSS/CSS3
[4]http://www.ecma-international.org/ecma-262/6.0/
[5]http://www.campusmvp.es/recursos/post/ECMAScript-6-es-ya-un-estandar-cerrado.aspx

polo que é o usado neste proxecto xa que conta con bastante soporte por practicamente todos os navegadores. Ademais incorpora bastantes melloras como poden ser promesas para programación asíncrona, verdadeiras clases para programación orientada a obxectos, iteradores, xeradores, etc. Por outra banda é posible facelo compactible con navegadores antigos, xa que permite usar algún tipo de traductor como babel, que permite traducir o código de ES6 a ES5, para poder ser executado. JavaScript, xunto coa API DOM de HTML, proporcionara dinamismo na nosa interface web.

4.4.4. JSON

JSON (JavaScript Object Notation) é un formato sinxelo para representar datos que viaxan pola web, que usa a sintaxe dos obxectos JavaScript para codificar os datos. Ademais é un formato no que é fácil construír un parseador de forma sinxela, posto que segue os estándares ECMA-404[6] e RFC7159[7].

Este formato componse de pares chave:valor, onde a chave é unha cadea de texto que se emprega como identificador e o valor poden ser calquera tipo de datos javascript.

As vantaxes que ten este formato con respecto a outros son a súa simplicidade, e polo tanto é comprensible por case calquera linguaxe de programación. Ademais, ao ser tan simple aforra ancho de banda, tempo de transmisión... o que produce que sexa moi boa elección sobre todo en entornos web.

4.4.5. AJAX

AJAX (Asynchronous JavaScript And XML) consiste nunha técnica de desenvolvemento web, baseada en JavaScript e XML, para crear aplicacións interactivas. Este método permite levar a cabo comunicacións asíncronas co servidor.

A comunicación asíncrona permite enviar unha serie de peticións ao servidor que devolven o contido necesario para ser utilizado na construción ou representación na aplicación web.

Inicialmente esta proposta estaba pensado para que usara XML como formato de intercambio, aínda que na actualidade permite numerosos formatos, JSON é un dos mais empregados.

4.4.6. Java

Java é unha linguaxe de programación de propósito xeral, concorrente e orientado a obxectos desenvolvido a mediados da década dos 90 por Sun Microsystems, cunha forte influencia de C++, Pascal e Objetive-C.

[6] http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[7] https://tools.ietf.org/html/rfc7159

Actualmente o desenvolvemento pertence a Oracle Corporation tras a adquisición de Sun Microsystems en 2009.

O éxito de Java basease no principio WORA (Write Once, Run Anywhere), é dicir, escribir o código do programa unha soa vez e podelo executar sobre calquera plataforma compatible. Para conseguir isto inclúese dentro da distribución Java un compilador a bytecode, que non deixa de ser unha representación intermedia do código que toma a máquina virtual de Java. Unha vez compilado o código fonte en formato bytecode pode ser lanzado en calquera máquina virtual de Java sen importar a arquitectura que teña o sistema onde se este a executar sempre que exista unha máquina virtual que funcione en dita arquitectura.

Java é actualmente unha das linguaxes de programación máis empregadas a nivel mundial en parte pola facilidade de uso sobre outros linguaxes de programación como C++ e tamén pola gran dispoñibilidade de librarías. Na figura 4.7 pódese observar a evolución en materia de características e posibilidades novas engadidas:

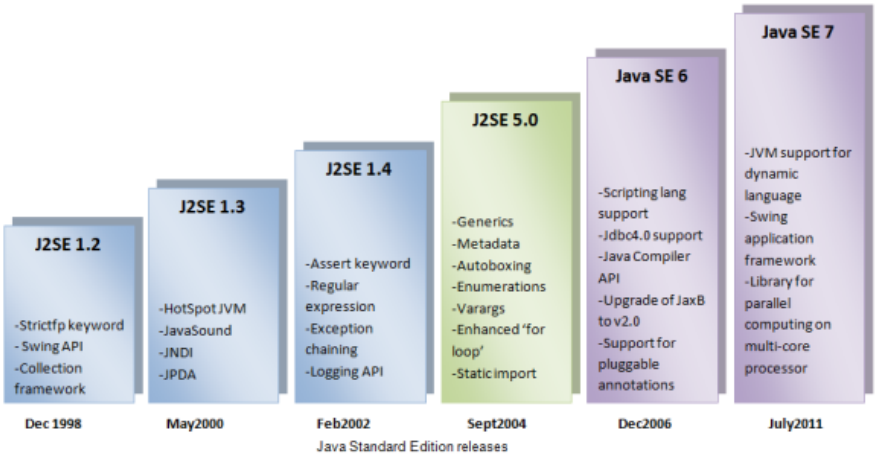

Figura 4.7: Evolución de Java

4.4.7. MongoDB

MongoDB é un xestor de base de datos NoSQL, orientada a documentos, lanzado a finais dos anos 2000 e situase como un dos xestores de bases de datos NoSQL máis empregados na actualidade.

Esta escrito en C++, multiplataforma e conta con drivers para ser empregado nun gran abano de linguaxes de programación (Scala, C, Python, NodeJS...). Ademais conta cunha licencia libre GNU AGPL v3.0

MongoDB garda a estrutura dos datos en documentos de tipo JSON que se almacenan en disco mediante un formato binario chamado BSON, o que implica que non existe un esquema prefixado para almacenar os datos.

4.4.8. Scala

Scala é unha linguaxe de programación multiparadigma que combina as características de linguaxes funcionais xunto coas características da programación orientada a obxectos ademais de incorporar unha forma de programar concisa, elegante e con tipos seguros.

Apareceu a comezo dos anos 2000 deseñado por Martin Odersky no Laboratorio de métodos de programación de la EPFL. A súa implementación actualmente funciona sobre a máquina virtual de Java e é compatible coas aplicacións de Java existentes.

4.5. Ferramentas

Neste apartado inclúense todas as ferramentas empregadas durante o desenvolvemento do presente proxecto realizando unha breve descrición sobre elas.

4.5.1. Maven

Maven é unha ferramenta de software para a xestión e construción de proxectos Java creada por Jason van Zyl a inicios dos anos 2000, que conta cunha licencia Apache 2.0.

O seu funcionamento basease na construción dun ficheiro en formato XML onde se definen as tarefas a realizar e as dependencias do proxecto a desenvolver. Cando chega o momento de construír o software Maven automaticamente descarga as dependencias necesarias e constrúe a aplicación cos arquivos compilados, aforrando deste xeito moito traballo aos desenvolvedores. Ademais esta ferramenta conta con moitos plugins adicionais, proporcionados pola comunidade, que amplían en gran medida a súa funcionalidade.

4.5.2. Netbeans

Netbeans é un IDE para o desenvolvemento de proxectos empregando diferentes tecnoloxías e linguaxe de desenvolvemento. Creado por Sun Microsystems no ano 2000 e continuado por Oracle Corporation desde a súa compra.

Netbeans proporciona utilidades para os programadores, como son as funcións de depuración e navegación a través do código para corrixir erros, un motor de autocompletado moi potente, integración con Maven, dispón de control de versión...

Ademais dispón dunha comunidade moi activa onde se proporcionan numerosos plugins para aumentar a súa funcionalidade. Este software e totalmente gratuíto, e conta coa licencia de código aberto GPLv2

4.5.3. Sublime Text

Sublime Text é un editor de texto escrito en código C++ e Python por Jon Skinner a finais dos anos 2000. Aínda que a idea orixinal era que fose unha extensión de Vim, finalmente foi collendo unha identidade propia de editor. Este editor pode ser probado de forma gratuíta, aínda que sexa software privativo. Este software esta dispoñible para os sistemas Windows, Mac e GNU/Linux.

Ten soporte para unha gran cantidade de linguaxes de programación, entre as que se atopan por defecto HTML, CSS, JavaScript, Scala, Python... aínda que tamén posibilita a descarga de módulos para ter compatibilidade con máis linguaxes.

Proporciona ao desenvolvedor as capacidades de autocompletado do código, de movemento polo código fonte a través dun mapa, de multicursor e multiselección, de consola propia, etc.

4.5.4. Microsoft Project

Microsoft Project é un software de xestión de proxectos creado por Microsoft. Este software é empregado pola maior parte de xestores de proxecto arredor do mundo, debido a que proporciona un gran número de funcionalidades que abarcan dende a planificación temporal do proxecto até a xestión dos recursos. Aínda que existen solucións baseadas en software libre como é o ProjectLibre, non conseguen alcanzar o nivel de calidade que este software proporciona.

4.5.5. StarUML

StarUML é un software de creación de diagramas en UML empregada durante a fase de deseño, esta desenvolvido por MKLAB. A súa licencia era de software libre coa GPLv2 até 2014 onde co lanzamento da versión 2.0 do software foi trocada por unha licencia privativa.

A versión empregada para a realización é a última (v2.7), lanzada o 20 de abril, no momento da realización do proxecto, que posúe soporte para todo tipo de diagramas baseados en UML2 e diagramas de entidade-relación.

Ademais de todo isto permite a xeración automática de código a partir dos diagramas, aínda que este servizo non vai ser empregado no proxecto actual. Tamén permite a incorporación de gran documentación dentro dos diagramas creados.

4.6. Frameworks e bibliotecas

Nesta sección recóllense unha serie de bibliotecas e frameworks necesarios ou empregados para desenvolvemento do sistema. Entendese por biblioteca como un conxunto de funcións, métodos, procedementos, obxectos, etc. destinados a

realizar unha serie moi ben definidas de operacións, mentres que un framework abarca un campo bastante mais amplo, xa que ademais de realizar o mesmo que unha biblioteca, proporciona o esqueleto sobre a cal se constrúe toda a lóxica [12].

4.6.1. React

React[8] é unha biblioteca JavaScript creada en 2013 para desenvolver interfaces de usuarios que é usada por grandes compañías como Facebook ou Instagram. O desenvolvemento desta plataforma pretende solucionar un problema que consiste na construción de grandes aplicacións con datos que cambian a través do tempo [13].

React permite combinarse con outras librarías como pode ser Node.js ou AngularJs, ademais de permitir de forma sinxela empezar con un proxecto React. Outra característica importante é que permite construír compoñentes reutilizables, permitindo reutilizar código e testealo de forma máis sinxela.

Ademais React incorpora un Virtual DOM, en vez de confiar so no DOM do navegador o que permite máis facilidade a biblioteca para comparar os contidos entre a versión vella e a almacenada no virtual DOM, deste xeito determina como actualizar eficientemente o DOM do navegador [14].

Cabe destacar que React proporciona unicamente as vistas (V) dentro dun módelo de arquitectura MVCVM ou MVC, xa que como se comentou anteriormente, mediante esta libraría podemos construír compoñentes gráficos personalizados e reutilizar compoñentes xa existentes.

4.6.2. Material-UI

Material-UI[9] é unha biblioteca de compoñentes para React que permiten dar estilo os sitios web construídos con React, baseada na especificación Material design, que esta enfocada na visualización do sistema operativo Android, ademais da web, etc. Esta normativa foi desenvolvida por Google e anunciada o 25 de xuño do 2014.

4.6.3. Apache Hadoop

Apache Hadoop[10] é un framework creado por Apache que permite o procesamento distribuído de grandes conxunto de datos a través dun conxunto de computadoras, clusters, utilizando modelos de programación sinxelos. O seu deseño esta pensado para permitir pasar de poucos nodos a miles de nodos de forma áxil.

[8] https://facebook.github.io/react/
[9] http://www.material-ui.com/#/
[10] http://hadoop.apache.org/

Apache Hadoop é un sistema distribuido usando unha arquitectura Master-Slave que emprega o sistema de ficheiros Hadoop Distributed File System (HDFS) como almacenamento, baseado en Google GFS. Ademais emprega algorimos Map-Reduce para realizar o procesamento de alta cantidade de información, utilizado por google. É software libre, xa que posúe a licencia Apache License 2.0 e esta programado en Java.

4.6.4. Scalding

Scalding[11] é unha librería para a linguaxe de programación Scala que fai sinxelo especificar traballos para Apache Hadoop empregando o paradigma MapReduce. Esta construído por encima de Cascading, unha libraría Java que abstrae os detalles de baixo nivel de Hadoop (xa que como se comentou nun apartado anterior Scala é compatible con Java). Scalding é comparable a Pig, pero ofrece unha integración con Scala.

Foi creada por twitter e actualmente esta na versión 0.16.0, polo que aínda esta en fase de desenvolvemento alpha pero que xa comeza a ser amplamente empregada, pola súa facilidade de uso e integración.

4.6.5. Scalakata

Scalakata[12] é un editor de código en Scala, que esta escrito en Scala e foi creado por Massé Guillaume que permite dun xeito sinxelo comprobar a sintaxe do código introducido, compilalo e observar os resultados de operacións sinxelas e directas a través do código.

Este editor permite introducir facilmente novas librerías, como por exemplo a de Scalding, e deste xeito poder incluír a súa sintaxe, atallos a hora de seleccionar métodos, etc.

4.6.6. React-Router

React-Router[13] é unha libraría para React que permite manter a interfaz web en sincronía coa URL e deste xeito enrutar as diferentes partes da aplicación React. Este posúe unha API aínda que sinxela conta con bastantes características, como redireccións, interpretación dos parametros na URL, etc. Dentro do noso proxecto esta libraría englobariase dentro do controlador (C) do modelo de arquitectura MVCVM.

[11] https://github.com/twitter/scalding
[12] https://github.com/MasseGuillaume/ScalaKata2
[13] https://github.com/reactjs/react-router

4.6.7. Redux

Redux[14] é un contedor de estado predecible para aplicacións JavaScript que axuda a construír aplicacións consistentes, que se poden executar en diferentes entornos (cliente, servidor, etc.). Esta baseada na arquitectura Flux, polo que promete que os datos vaian nunha soa dirección, aínda que con algunha diferencia como pode ser que manten o seu estado inmutable para conseguir un comportamento da aplicación repetible e reproducible. O estado so pode ser cambiado empregando e disparando accións, onde cada unha delas produce un novo estado, (o estado nunca se cambia, se non que se produce un novo estado e ese pasa a ser o estado da aplicación) a través de un ou mais reducers. Dentro deste proxecto tal e como se comentou dentro do apartado da arquitectura Flux, Redux sera o módelo (M) dentro da arquitectura MVCVM.

[14]https://github.com/reactjs/redux

Capítulo 5

Deseño e implementación da aplicación

O paso seguinte durante o proceso de desenvolvemento dun proxecto software pasa pola definición formal dun deseño sobre o cal traballar até o momento de inicializar a implementación. Este deseño pódese afrontar de moi diversas maneiras, polo que non existe unha única solución correcta se non que existen moitas e diversas formas de darlle solución dende o punto de vista de deseño o problema presentado. A fase de deseño vaise afrontar seguindo unha aproximación segundo o nivel de abstracción que se precise e o destinatario ao que vaian destinados ditos diagramas, isto implica que non é necesario o mesmo nivel de detalle nun diagrama destinado a amosarlle ao cliente que o necesario nun diagrama que vai a ser utilizado para o equipo de desenvolvemento para basear a codificación ou implementación do sistema na linguaxe que se determine oportuna.

Resulta especialmente interesante realizar varios deseños con diferentes niveis de abstracción que complementen o código fonte, pois axuda a favorecer o mantemento e extensibilidade deste no futuro, debido a que sempre resultará máis sinxelo basearse nun gráfico que modele todo o conxunto para entender como está feito o software e de que elementos esta composto, posibilitando a incorporación de melloras ou correccións de forma máis eficiente, ao poder ver a simple vista, e percibir onde é o mellor lugar para a levar a cabo a súa incorporación.

Ademais, de ser útiles para clarificar o funcionamento interno do software ou realizar unha descomposición en pequenos elementos, resulta interesante en desenvolvemento de proxectos onde se solicite a construción dunha interface gráfica de usuario, a realización dun deseño das distintas vistas necesarias antes de comezar coa implementación da lóxica que as ha de controlar. Deste xeito resultará máis sinxelo aplicar os cambios que o cliente necesite antes de ter a súa implementación debido a que o cliente pode ter unha primeira impresión de como vai ser a interface.

Nos seguintes apartados atópanse os diferentes diagramas nos que se especifican todos os deseños mencionados anteriormente.

5.1. Deseño da arquitectura do sistema

Neste apartado pretendese dar unha definición, cun elevado nivel de abstracción, dunha arquitectura que permita separar o deseño da implementación. Deste xeito poderíanse realizar varias versións da aplicación sen ter necesariamente que modificar o deseño inicial. Para realizar o modelado do software farase uso de UML (Unified Modeling Language) [15]. Os diagramas que se mostran máis abaixo buscan aportar unha visión de alto nivel dos diferentes elementos que compoñen o sistema a desenvolver e de como estes se relacionan entre si.

Resultou de especial interese facer un enfoque deste estilo, debido a que a sinxeleza deste diagrama permite, por unha banda mostrar para o cliente as distintas divisións dentro do conxunto do sistema a desenvolver, e por outra banda permite o equipo de desenvolvemento diferenciar os distintos módulos ou compoñentes, facilitando gran parte do traballo en etapas posteriores, xa que o descompoñer o sistema en subsistemas máis pequenos resulta máis sinxelo a hora de deseñalos.

Para deseñar a arquitectura do sistema, ademais de ter en conta os requisitos funcionais, tamén será de especial interese ter en conta aqueles requisitos non funcionais que se poidan cumprir nesta etapa, se sobre todo, se aportan un alto valor para o cliente como poden ser:

- **RNF-002** - *Empregar servizos web*
- **RNF-008** - *Implementar unha arquitectura modular*
- **RNF-009** - *Empregar Apache Hadoop*
- **RNF-001** - *Empregar MongoDB*

Na figura 5.1 descríbese a interación de moi alto nivel entre un cliente e os diferentes elementos que compón o sistema real, onde o cliente fai as peticións mediante a interface web que se comunica cos servidores que conteñen os diferentes módulos de xestión. O módulo que aparece nun servidor adicional consiste nos servizos externos empregados neste proxecto. Mediante esta arquitectura, o usuario pode empregar a interface web para a creación, edición e borrado de KPIs e a súa representación no dashboard, que actuan tanto sobre a base de datos como a plataforma Apache Hadoop, durante o proceso de envio da formulación, a través dos servizos web que están aloxados no servidor.

Na figura 5.2 pódese observar un diagrama da arquitectura do sistema completo de alto nivel, utilizado simplemente para amosar o contexto no cal se vai a desenvolver o proxecto software, e os diferentes módulos no que esta descomposto o software.

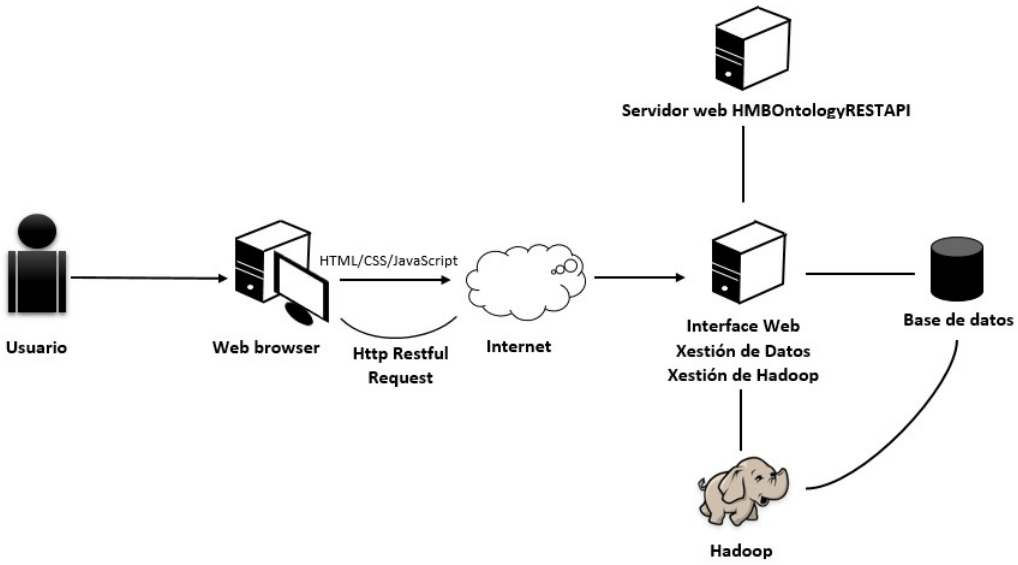

Figura 5.1: Diagrama de toda a interación co sistema

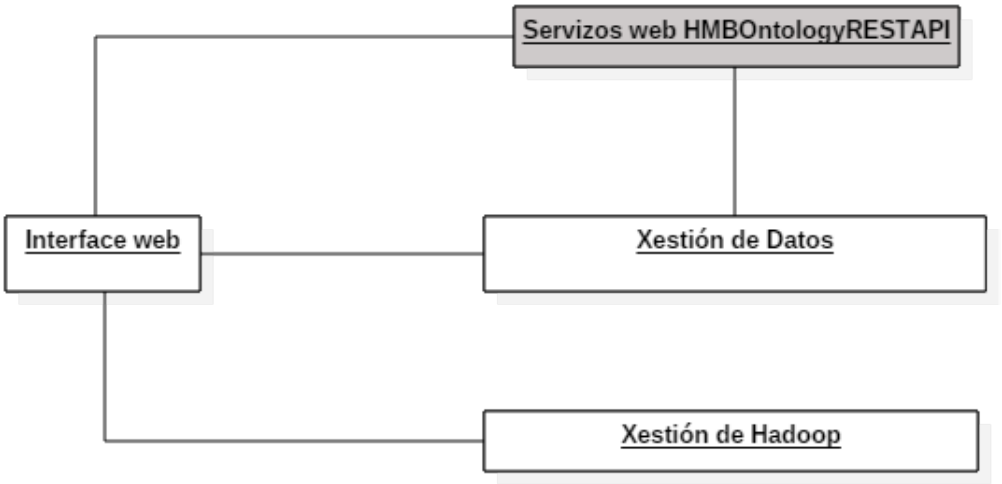

Figura 5.2: Diagrama da arquitectura do sistema completo

Dentro do diagrama 5.2, os diferentes cadros representan os distintos módulos polo que esta composto o sistema, mentres que as liñas de unión representan as comunicación entre os diferentes módulos. As entidades de cor máis escura representan os módulos actualmente dispoñibles e totalmente funcionais que serán

empregados polos módulos de fondo branco, que son os compoñentes a desenvolver neste proxecto.

Na figura 5.3 inclúese a mesma arquitectura que a mostrada na figura anterior pero con maior nivel de detalle. Como se pode observar no diagrama, mostrase o sistema a desenvolver descomposto en tres módulos:

1. **Módulo de Xestión de Datos**: que permitirá o tratamento dos datos das KPIs, plantillas de código para a creación das KPI, dos usuarios dispoñibles no sistema, tratando o seu login e o dashboard de cada un deles.

2. **Módulo de Xestión de Hadoop**: que permitirá a comunicación co framework de Apache Hadoop para realizar as correspondentes operacións dependendo da KPI creada e que deste xeito poida este procesar os diferentes ficheiros de log ou rexistros para obter un resultado e almacenalo.

3. **Módulo da Interface web**: este módulo permitirá a iteración por parte do usuario con todos os demais módulos, tanto os módulos mencionados anteriormente (Xestión de Datos, Xestión de Hadoop) como os servizos xa existentes no momento de desenvolver o proxecto (HMBOntologyRESTAPI).

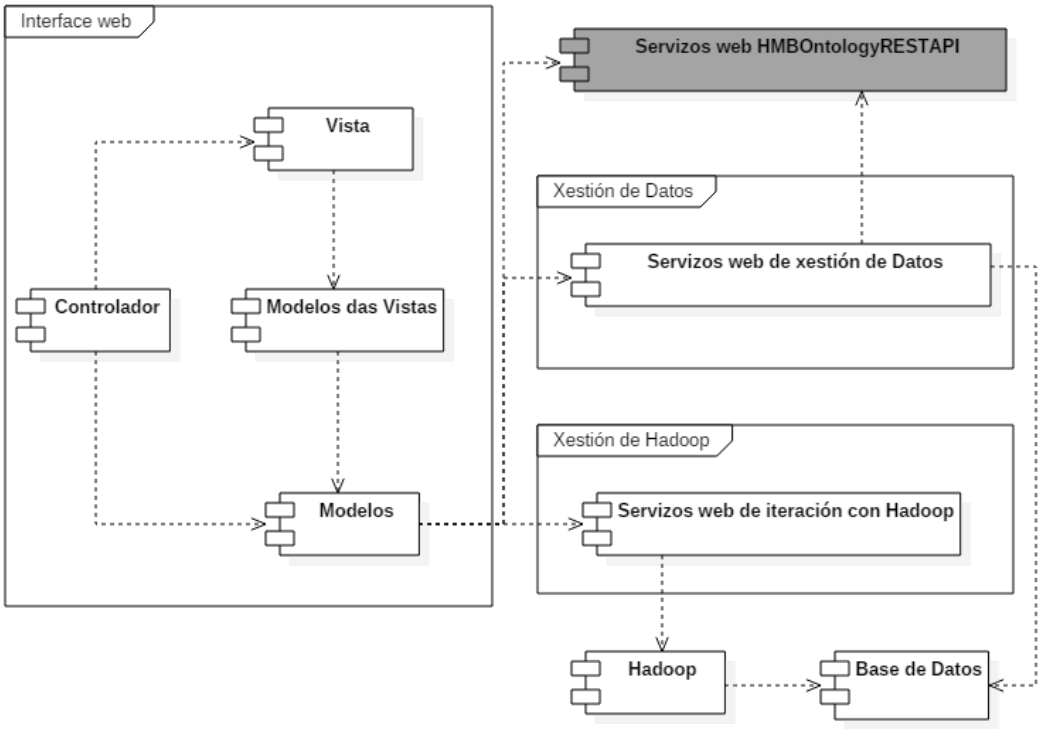

Figura 5.3: Diagrama da arquitectura do sistema completo con maior nivel de detalle

Con respecto ás comunicacións entre os diferentes módulos, finalmente decidiuse empregar os servizos web RESTful, debido a sinxeleza de traballar con datos en formato JSON, concretamente para o desenvolvemento dos módulos de Xestión de Datos e para o de Xestión de Hadoop, debido a que o módulo HMBOntologyRESTAPI xa expón o acceso a todos os seus métodos e funcionalidades empregando esta tecnoloxía de xeito que non é necesario implementar unha capa superior que englobe este módulo.

Para a arquitectura da interface web esta emprega un modelo de arquitectura Modelo-Vista-Modelo de Vista (MVCVM) un pouco modificado onde React proporciona as Vistas, React-Router proporciona o Controlador e Redux proporciona a parte do Modelo empregando o patrón Flux. É importante destacar que todas as tecnoloxías e estratexias de deseño mencionadas aparecen explicadas máis detalladamente no capítulo 4 da presente memoria.

5.1.1. Arquitectura da interface web

Como se pode observar no diagrama da figura 5.4 a arquitectura da interface web divídese en catro compoñentes esenciais:

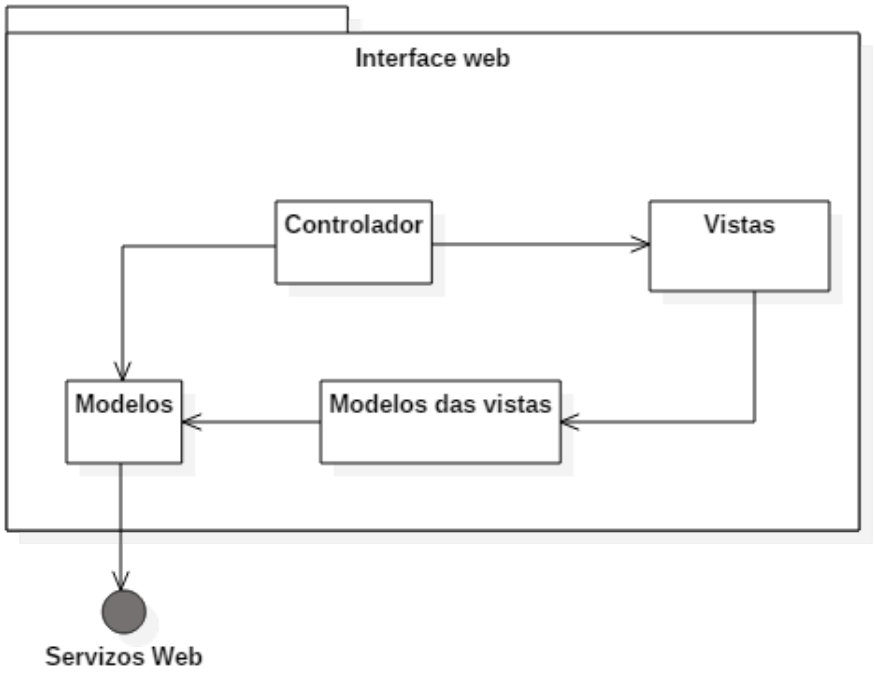

Figura 5.4: Diagrama simplificado da arquitectura da interface web

- **Vistas**: Estas son as encargadas de realizar toda a iteración co usuario, mos-

trando información, permitindo realizar accións, etc. Estas comunícanse cos modelos das vistas, facilitando a introdución dos datos como a visualización deles.

- **Modelos das vistas**: Estas son as encargadas de implementar a lóxica de actualización e manexo das funcionalidades dentro de cada vista, comunicándose cos diferentes modelos que levan a cabo a lóxica nun nivel máis baixo.

- **Modelos**: Son os encargados de levar a cabo toda a lóxica das chamadas dos Modelos das vistas até a actualización do estado dos elementos da Interface web. Tamén son os encargados de realizar as chamadas aos servizos web, cando os necesite a interface web.

- **Controladores**: Son os encargados de procesar os datos e decidir as rutas da aplicación, invocando as correspondentes vistas e modelos en función das accións tomadas polo usuario.

Fora do que é a interface web en si mesma, atopámonos con outro compoñente, composto polos servizos web, tanto o do módulo xa proporcionado, como os dos diferentes módulos a desenvolver. Estes servizos son chamados a través dos modelos, mediante a realización de chamadas asíncronas AJAX.

Se continuamos desenvolvendo máis en detalle o diagrama anterior, isto lévanos a obtención do diagrama da figura 5.5, onde se mostra cun maior nivel de detalle a especificación da arquitectura da interface web vendo como se descompón cada un dos submódulos definidos anteriormente.

Como se pode observar a primeira variación clara é a de incorporar os **Compoñentes Personalizados**. Isto facilita a creación de novos compoñentes que permiten modularizar o desenvolvemento de interfaces web até un punto que non estaba pensado até fai uns poucos anos. Nesta arquitectura as vistas, están formadas por diferentes compoñentes personalizados, que se inicializan dende un controlador, pero son os modelos da vista definidos para cada un destes os encargados de enlazar os datos e de implementar as función propias de cada un destes compoñentes, mentres que os modelos son os encargados da actualización dos estados dos propios datos e das chamadas aos servizos web dos diferentes módulos.

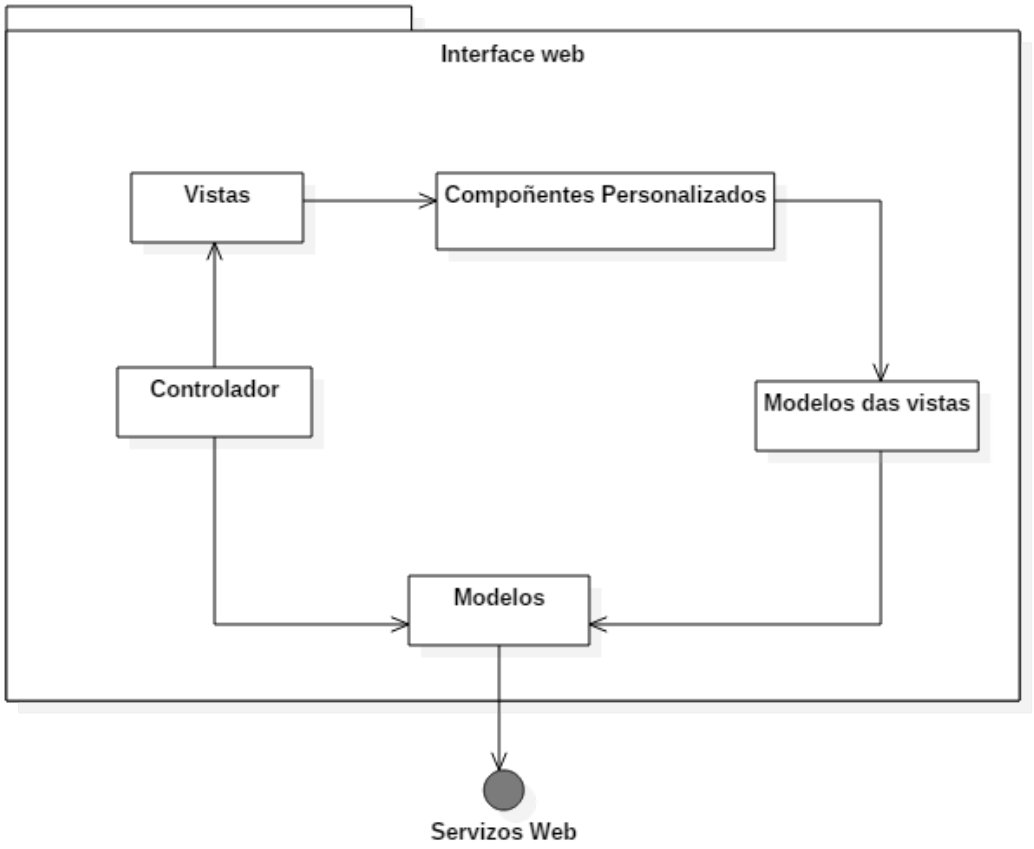

Figura 5.5: Diagrama completo da arquitectura da interface web

5.1.2. Arquitectura do módulo de xestión de datos

Para o módulo de xestión de datos atopámonos cunha arquitectura moito mais sinxela e básica que a da interface web, formado soamente por dúas compoñentes, como se pode observar na figura 5.6 e que esta composto por:

- **Servizos web**: Un submódulo que empregará a interface web para comunicarse cos métodos de manipulación da base de datos, obtendo a través deste os resultados necesarios ou aplicando os cambios desexados.

- **Acceso a datos**: Neste submódulo delegase a responsabilidade do acceso a base de datos tanto para o almacenamento de datos como para o acceso aos datos necesarios que precise a interface web.

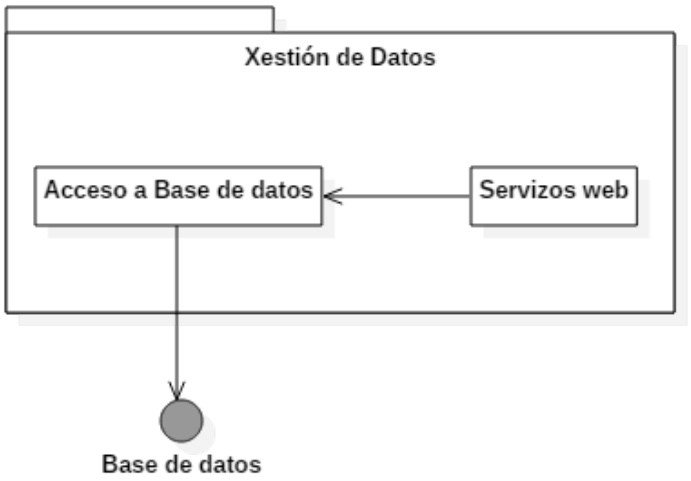

Figura 5.6: Diagrama da arquitectura do módulo de xestión de datos.

5.1.3. Arquitectura do módulo de xestión de Hadoop

Neste caso atopámonos cunha arquitectura que posúe a mesma complexidade que o anterior desde un punto de vista de representación, xa que o módulo esta formado practicamente polos mesmos elementos, como se pode observar na figura 5.7:

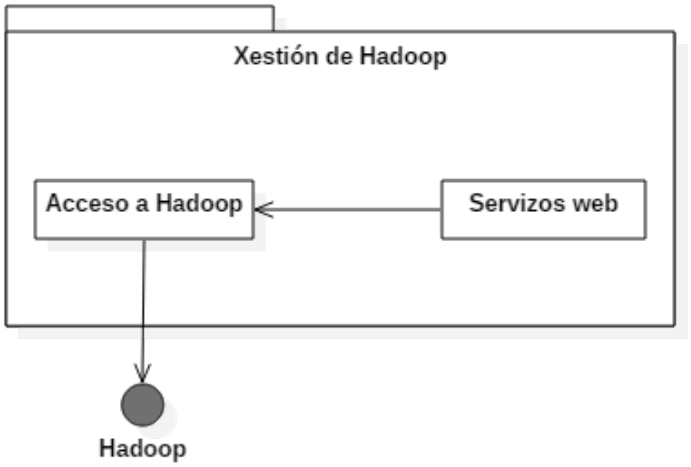

Figura 5.7: Diagrama da arquitectura do módulo de xestión de hadoop.

Onde por unha parte temos os **servizos web** que serán consultados pola

interface web, como no caso do módulo de xestión de datos e, por outra banda temos o submódulo no que se delega a parte de comunicación con Apache Hadoop para enviarlle os traballos a realizar.

5.2. Deseño das clases

O deseño dos diagramas de clase, aportan maior nivel de detalle sobre os diagramas de arquitectura no deseño do software que se vai a construír. Con isto alcánzanse un dos grandes obxectivos que é aportar as ferramentas para que os desenvolvedores encargados de realizar, ampliar e manter o sistema poidan facer o seu traballo de forma máis simple. Por este motivo, os diagramas deben ser o suficientemente claros e fiables á implementación final para que permitan comprender o sistema creado sen ter que observar o código fonte.

Este proceso é recomendable realizalo antes de comezar co proceso de implementación do software, aínda que non se debe temer á realizar modificacións sobre eles a medida que se vai implementando o código, seguindo sempre un control, para adaptalos aos problemas que poidan ir aparecendo, xa que é moi normal que no deseño non se plasme ao 100 % todo o que pode suceder a medida que se vai realizando a codificación.

Coa finalidade de que se poidan seguir os diagramas facilmente e saber de que parte é cada un, realizouse este deseño empregando a descomposición en módulos comentada no diagrama 5.2 da arquitectura do sistema. Todos os diagramas que se mostran nas seguintes seccións se corresponden coas últimas versión do desenvolvemento do produto.

5.2.1. Deseño das clases da interface web

Para a realización deste módulo empregáronse unha serie de diagramas, co fin de amosar a estrutura xerárquica dalgún dos compoñentes que forman o módulo. Concretamente creáronse 3 diagrama ben diferenciados:

- No diagrama da figura 5.8 pódese observar todos os compoñentes personalizados, que herdan do compoñente Component proporcionado polo framework React.

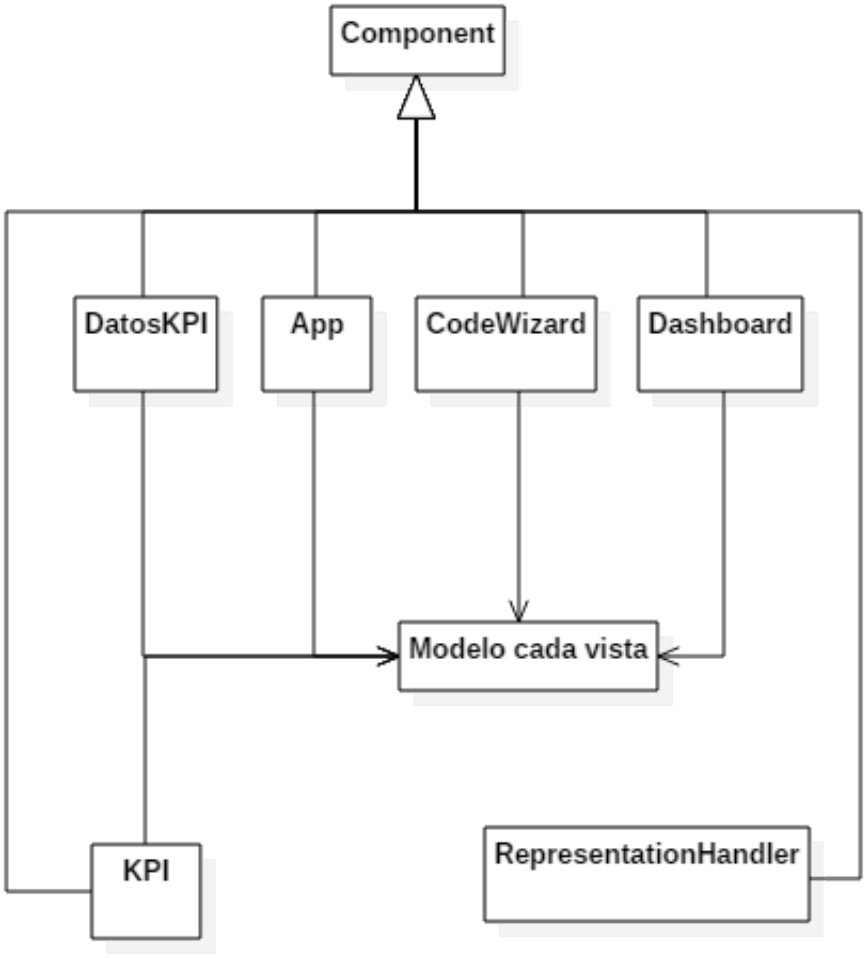

Figura 5.8: Diagrama de clases para a xerarquia de elementos personalizados da interface

- Na figura 5.9 pódese observar con mais detalle o modelo tanto das vistas como o que manipula as chamadas a AJAX, que se comunican cos servizos web dos outros modelos, e o store de Redux. Ademais pódese observar a relación entre os diferentes compoñentes, debido a que en React uns compoñentes pódense anidar sobre outros dentro da interface web formando unha estrutura en árbore como sucede en HTML.

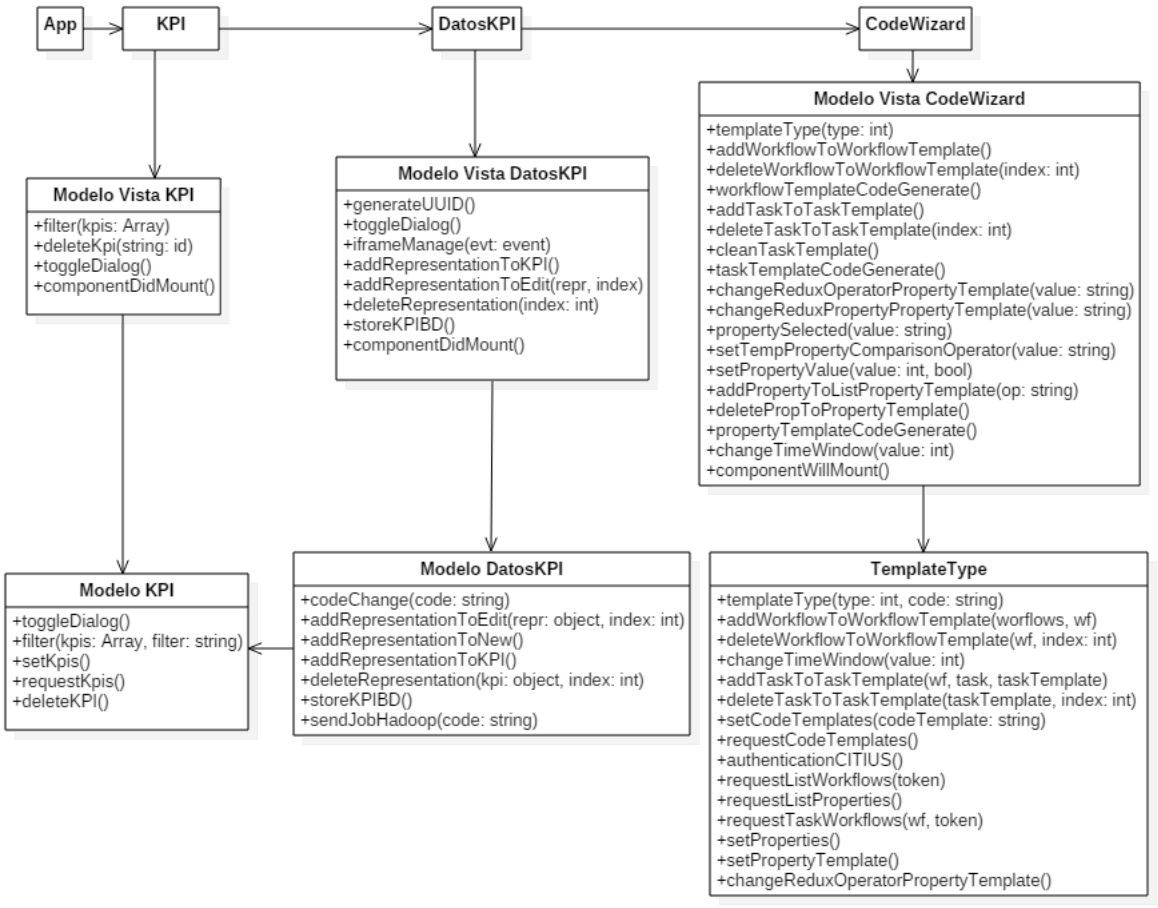

Figura 5.9: Diagrama de clases para os compoñentes KPI, DatosKPI e CodeWizard

- No diagrama da figura 5.10 pódese observar un deseño similar ao diagrama 5.9, aínda que neste caso o compoñente RepresentationHandler non posúe un modelo propio se non que posúe soamente un modelo de vista. Isto é debido a que este compoñente so apoia ao compoñente Dashboard para seleccionar a representación de cada KPI a mostrar por este sen modificar o estado en ningún momento nin realizar chamadas a AJAX.

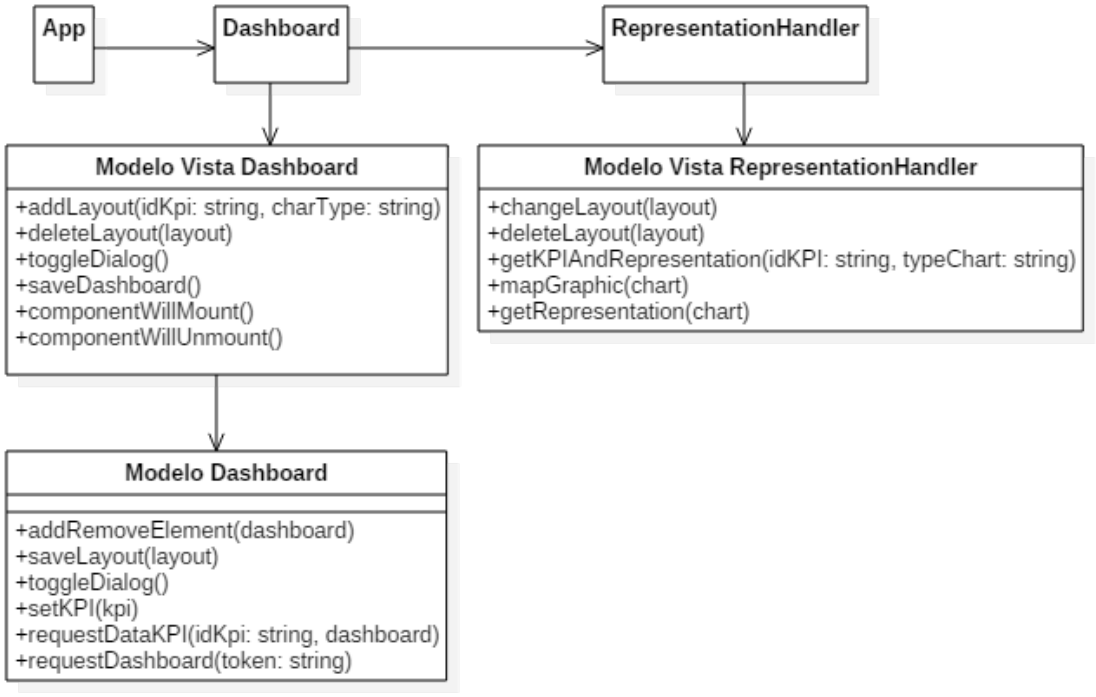

Figura 5.10: Diagrama de clases para os compoñentes Dashboard e RepresentationHandler

5.2.2. Deseño das clases do módulo de xestión de datos

O diagrama da figura 5.11 recolle a descomposición en diferentes clases do módulo do seguinte xeito:

- O paquete **ws** contén todas as clases encargadas de implementar os diferentes servizos web, ademais dunha clase que é a encargada de obter a conexión para establecer a comunicación coa Base de Datos MongoDB, implementando deste xeito o patrón singleton. Neste caso disponse de 4 clases para formar o conxunto de servizos web, onde se implementará en cada unha delas as súas respectivas rutas para recibir as peticións HTTP.
    - A clase **CRUDKPI** permite o acceso a base de datos para obter, crear, modificar e eliminar as KPIs.
    - A clase **CodeTemplate** contén os métodos para traballar coas plantillas de código que van a ser empregadas para a realización da formulación asociada a unha KPI no momento da súa creación.
    - A clase **HadoopData** proporciona acceso aos valores calculados por Apache Hadoop que almacenou en base de datos para as KPIs en función da formulación de cada unha delas.

- A clase **Usuarios** permite realizar as accións de login e logout sobre a base de datos, ademais de permitir manipular o dashboard, tanto para almacenalo como para obtelo e que sexa manipulado pola interface web.

- O paquete **helpers** contén as clases de apoio ou axuda das clases do paquete *ws*. Neste caso so forma parte del unha clase (**TasksHelper**) que é empregada para obter os nomes das tarefas a partir dos UUIDs dos rexistros.

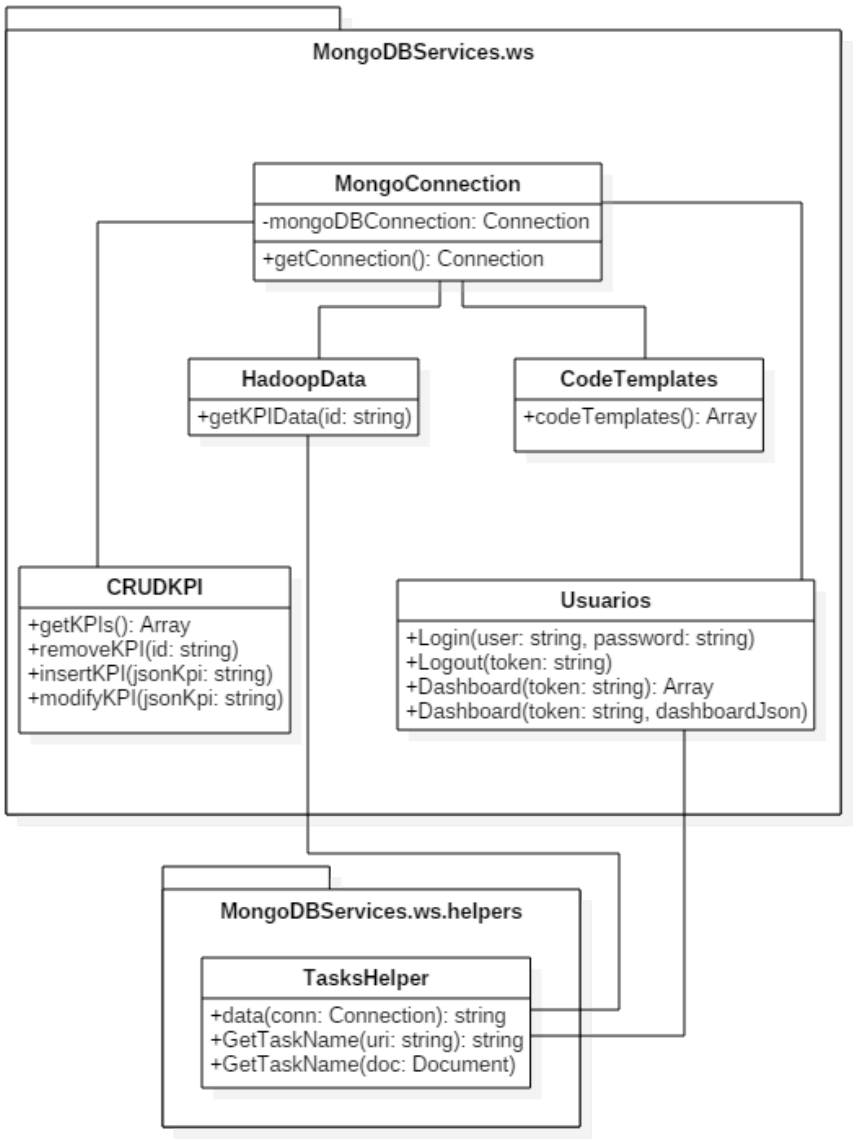

Figura 5.11: Diagrama de clases para o modulo de xestión de datos

Como se pode observar mediante este deseño queda validado o requisito **RNF-001** - *Empregar MongoDB*

5.2.3. Deseño do módulo de xestión de Hadoop

Como se pode observar no diagrama figura 5.12, este diagrama é aínda máis sinxelo que o anterior debido a que so conta cun paquete (**ws**), que a súa vez soamente ten unha clase (**Hadoop**). Esta clase dispón do servizo web, onde se implementa as rutas para ter o acceso a el mediante HTTP, con este servizo enviase a formulación da KPI a Apache Hadoop para que realice o filtrado dos rexistros en función de dita formulación. Con este deseño queda cumprido o requisito non funcional **RNF-009** - *Empregar Apache Hadoop.*

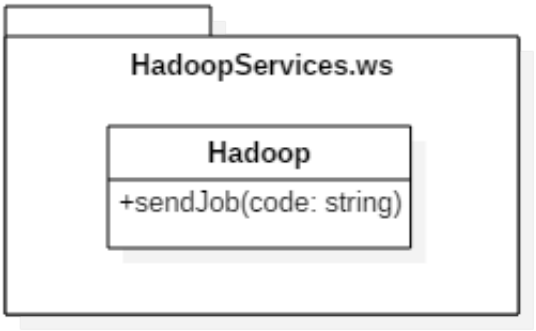

Figura 5.12: Diagrama de clases para o módulo de xestión de Hadoop

5.3. Deseño da interación

Nesta parte do documento recóllense un conxunto de diagramas de secuencia nos que se amosan as series de chamadas e fluxos de datos que ocorren durante a execución da aplicación para diferentes operacións. Cabe destacar que nos apartados seguintes non se atopan todos os diagramas, se non que se decidiu mostrar os que conlevan unha interación máis complexa, descartando a realización das operacións que sucederán dun xeito moi sinxelo ou similar á estrutura dalgún dos mostrados.

5.3.1. Diagramas do módulo de xestión de datos

Dentro deste módulo decidiuse realizar os diagrama para catro operacións principais: Creación dunha KPI, Eliminación dunha KPI (figura 5.14), Obtención dos datos devoltos por Hadoop asociados a unha KPI (figura 5.15) e Obtención do

dashboard (figura 5.16). A continuación, descrbiremos cada un destes diagramas máis polo miúdo.

Creación dunha KPI

Na figura 5.13 describimos o diagrama de secuencia para a creación dunha KPI. Os pasos son os seguintes:

1. A secuencia de interacións comeza coa chamada ao servizo web, que se chama insertKPI, onde recibe a KPI en formato JSON como string para ser inserida.

2. Para realizalo, a clase CRUDKPI realiza unha chamada a clase MongoConnection para desta forma obter a conexión. En caso de que está xa fora creada anteriormente devolvese a conexión existente, en caso contrario crea unha nova e devólvella a clase CRUDKPI.

3. A clase CRUDKPI realiza a operación de inserción na base de datos, concretamente na colección *kpis*, facendo uso da conexión citada anteriormente e devolve o resultado para ser mostrada na interface web.

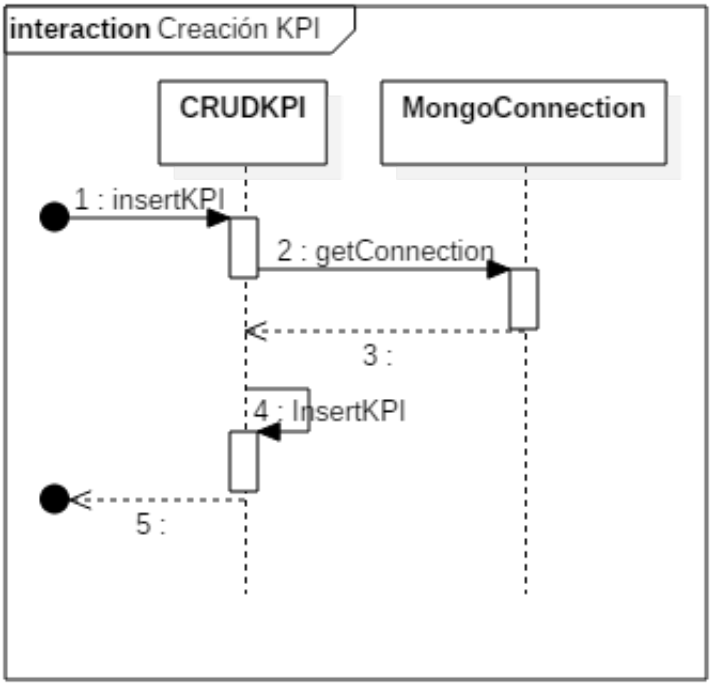

Figura 5.13: Diagrama de secuencia: Creación dunha KPI

Eliminación dunha KPI

Na figura 5.14 describimos o diagrama de secuencia para a eliminación dunha KPI. Os pasos son os seguintes:

1. A execución comeza coa chamada a o servizo web da clase CRUDKPI, removeKPI, onde se indica o ID da KPI a eliminar.

2. Esta clase pide á conexión a base de datos, para poder realizar as correspondentes operacións sobre ela.

3. Unha vez recibida a conexión, a clase CRUDKPI, realiza o borrado da KPI do conxunto de KPIs que formen parte do sistema.

4. Como tamén é preciso eliminar a KPI dos dashboards dos usuarios nos cales estea presente, faise unha solicitude á clase Usuarios para obtelos.

5. Esta clase solicita tamén a conexión na base de datos para poder procurar os dashboards para devolvelos a clase CRUDKPI.

6. A clase CRUDKPI realiza as modificacións do dashboard daqueles usuarios que tivesen representado esta KPI.

7. Finalmente, enviase o dashboard modificado á clase Usuario para que esta a inserte novamente na base de datos.

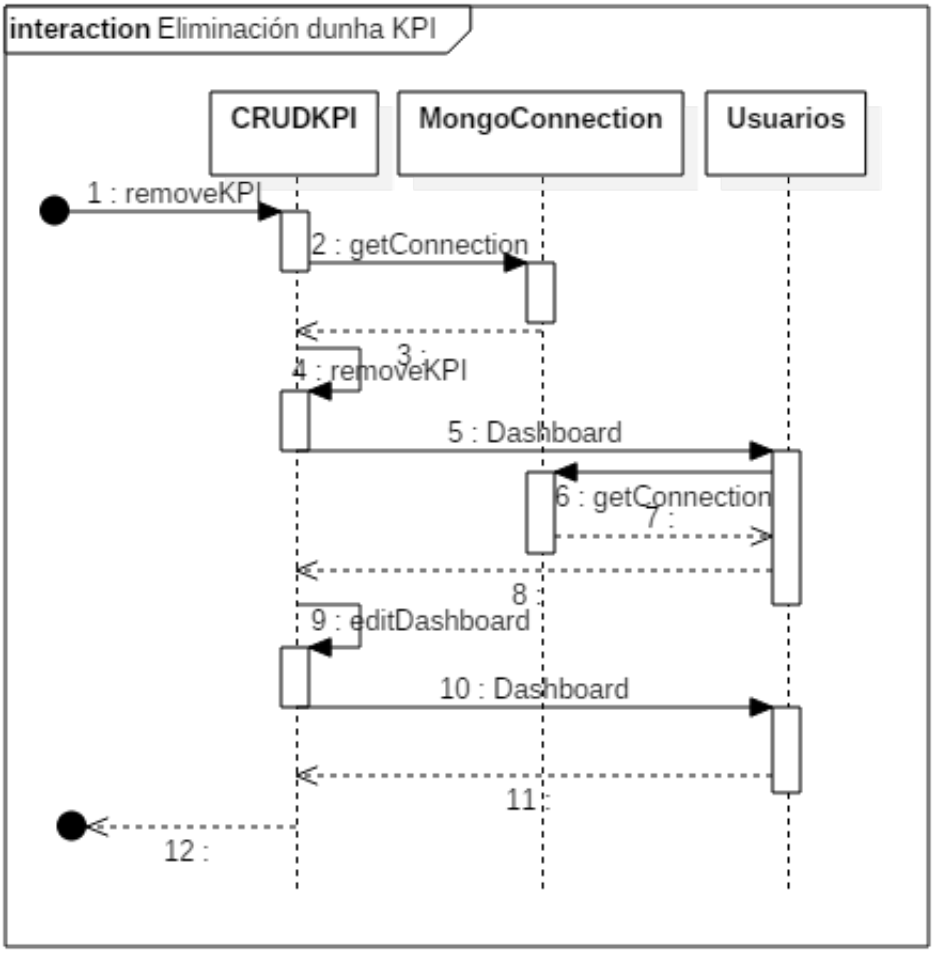

Figura 5.14: Diagrama de secuencia: Eliminación dunha KPI

Obtención dos datos de Hadoop asociados a unha KPI

Na figura 5.15 describimos o diagrama de secuencia para a obtención dos datos de Hadoop asociados a unha KPI. Os pasos son os seguintes:

1. A execución iniciase chamando ao método do servizo web da clase HadoopData, getKPIData.
2. Obtén a conexión da base datos.
3. Realiza as operacións de lectura dos datos das KPI sobre a base de datos e organiza a información.
4. Como os datos asociados a unha KPI poden ser nomes de tarefas en formato de UUID, este son convertidos a nomes lexibles a partir da chamada a clase

TaskHelper co método getTaskName, que devolve esas UUID traducidas. Finalmente devolve o conxunto de elementos do dashboard.

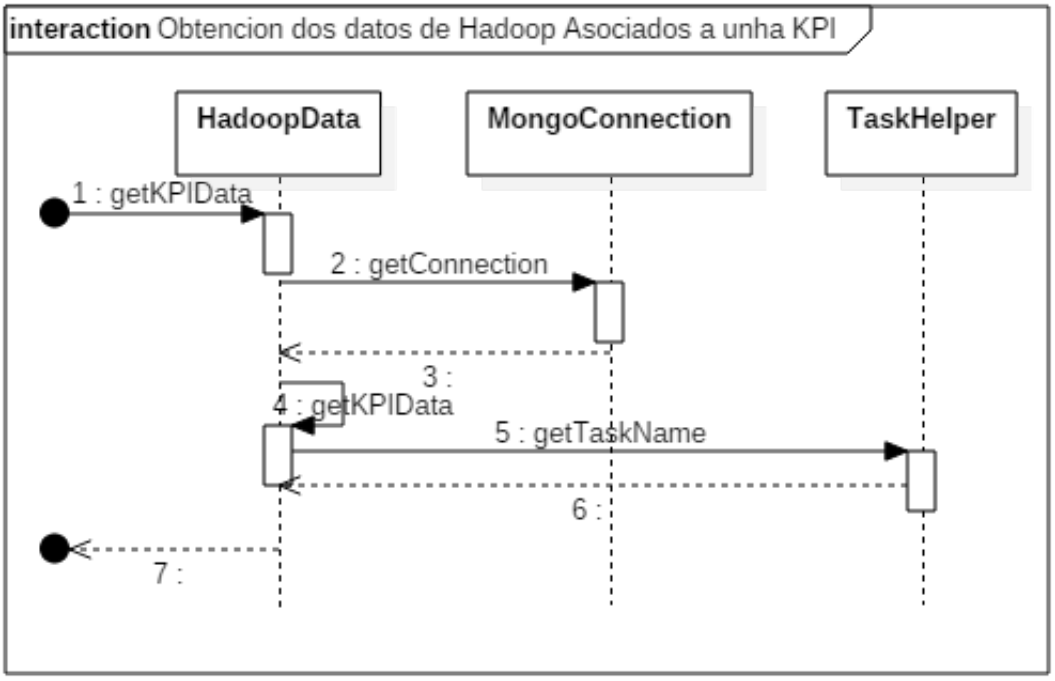

Figura 5.15: Diagrama de secuencia: Obtención dos datos devoltos por Hadoop asociados a unha KPI

Obtención do dashboard

Na figura 5.16 describimos o diagrama de secuencia para a obtención do dashboard dun usuario. Os pasos son os seguintes:

1. O diagrama de secuencia comeza coa execución do servizo web da clase Usuarios mediante o método getDashboard.

2. O método solicita a conexión por parte da base de datos para realizar as operacións sobre ela.

3. Realiza a petición do dashboard á base de datos mediante a conexión recibida e procesa os datos devoltos pola base de datos.

4. Como o nome dalgunha das variables poden ser en formato UUID, estas teñen que ser modificadas facendo unha conversión empregando a clase TaskHelper co método getTaskName que realiza as chamadas aos servizos web, que estaban implementados antes de comezar o proxecto.

5. Finalmente devolvese o dashboard para que sexa representado.

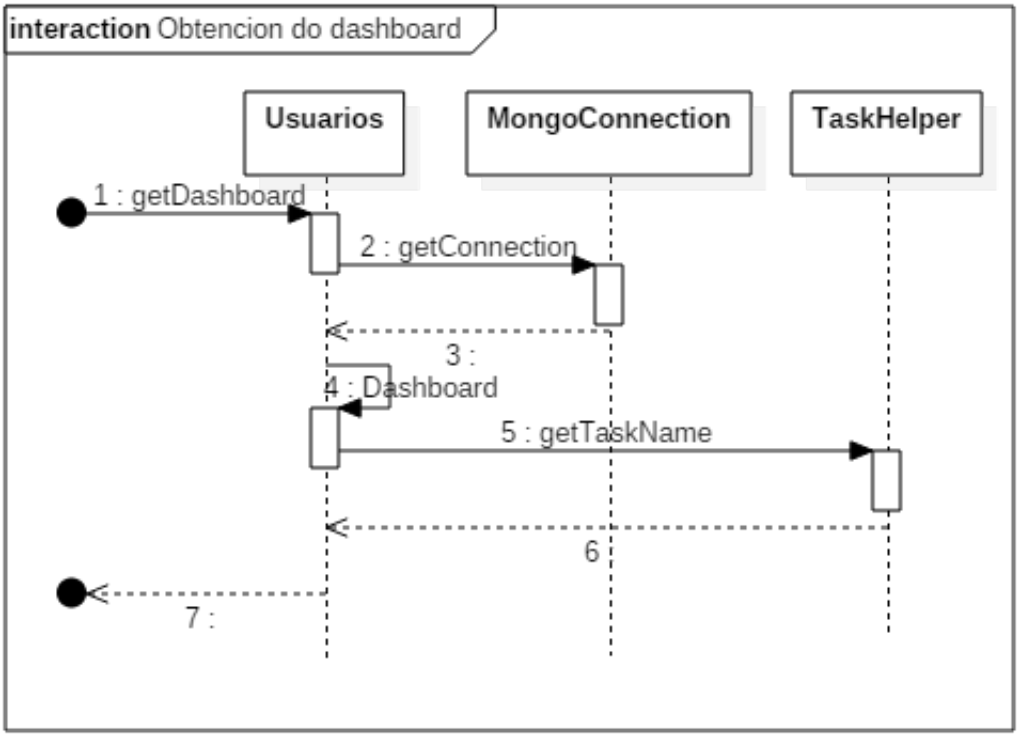

Figura 5.16: Diagrama de secuencia: Obtención do dashboard

5.3.2. Diagramas do módulo de xestión de Hadoop

Para este módulo so existe un diagrama de secuencia moi sinxelo, como se pode observar na figura 5.17. Os pasos son os seguintes:

1. A execución comeza chamando o servizo web da clase Hadoop sendJob.

2. Dentro desa clase realizase o envío do traballo, realizando cambios no formato da formulación que o arquivo executable de Hadoop non admite.

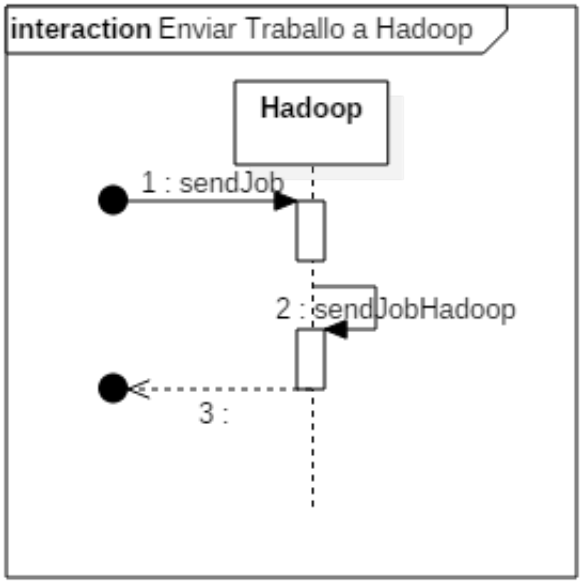

Figura 5.17: Diagrama de secuencia: Enviar traballo a Hadoop

5.3.3. Diagramas da interface web

No caso da interface web, existen unha enorme cantidade de operacións que un usuario pode levar a cabo. A continuación, inclúense os diagramas de secuencia das 4 accións máis complexas a realizar dende a interface web.

Cabe destacar que nestes diagramas se abstraeu moito o funcionamento que se amosa neles con respecto a realidade, xa que dentro do que é a interface web, se agrupou as vistas, compoñentes, modelos das vistas e modelos como unha única entidade por cada compoñente. Por outra parte os servizos web tamén se agruparon como un único módulo para simplificar a complexidade dos diagramas.

As iteracións escollidas foron: Engadir unha representación no dashboard (figura 5.18), Creación dunha kpi mediante plantillas (figura 5.19). Mover unha representación e almacenar o dashboard (figura 5.20) e Redimensionar unha representación e almacenar o dashboar (figura 5.21)

Engadir unha representación no dashboard

Na figura 5.18 describimos o diagrama de secuencia para engadir unha representación no dashboard dun usuario. Os pasos son os seguintes:

1. O usuario preme en engadir unha representación.
2. A interface solicita mediante un servizo web, getKPIs, as KPI do sistema para mostrar ao usuario.

3. Na interface móstranse as KPIs, onde o usuario escolle unha delas.

4. Unha vez seleccionada a KPI, mostráselle unha lista das representacións dispoñibles para esa KPI, onde o usuario selecciona unha delas.

5. A interface web realiza unha chamada ao servizo web para obter os datos filtrados por Hadoop asociadas a dita KPI mediante getDataKPI.

6. Finalmente mapéanse as variables á correspondente representación e devolvéselle ao usuario.

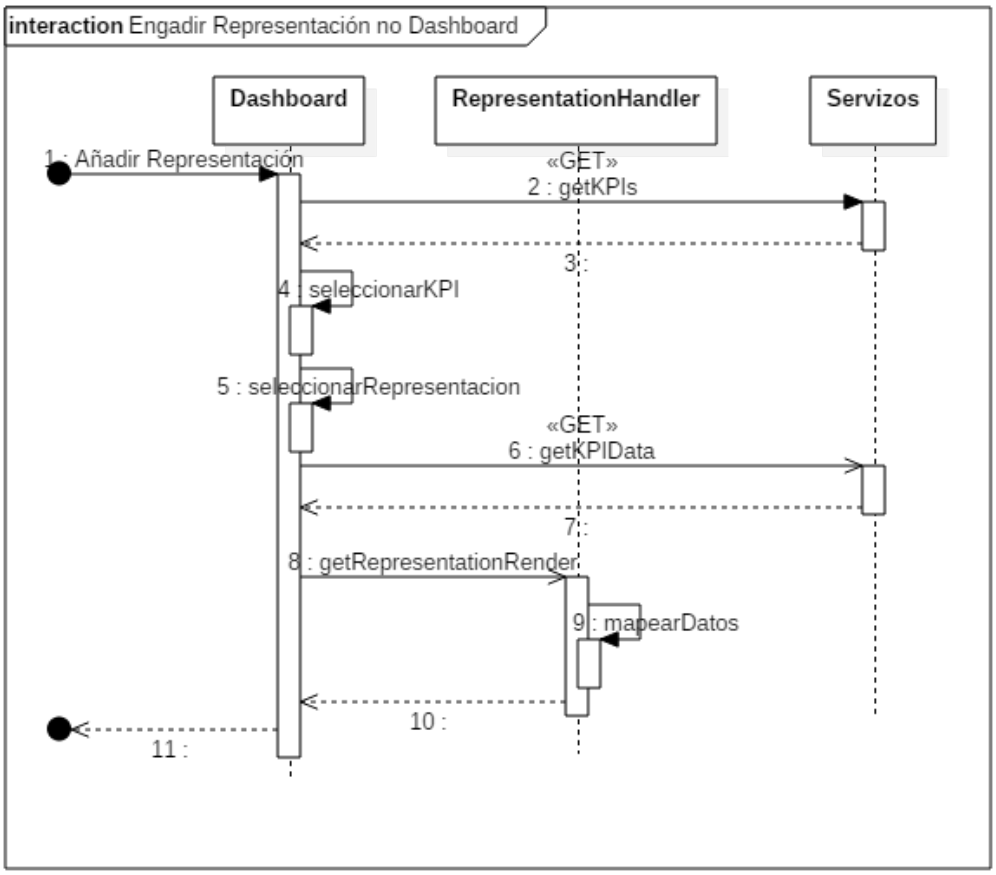

Figura 5.18: Diagrama de secuencia: Engadir representación no dashboard

Creación dunha KPI usando a plantilla de tarefas

Na figura 5.19 describimos o diagrama de secuencia para a creación dunha KPI usando a plantilla de tarefas. Os pasos son os seguintes:

1. O usuario preme sobre a interface web a opción de engadir unha nova KPI.

2. A interface web abre unha nova ventá con un formulario para engadir os datos básicos da KPI.

3. O usuario enche os datos básicos e preme en continuar, mentres que o sistema vai almacenando no estado estes datos.

4. A interface web abre un novo compoñente onde mostra as plantillas dispoñibles e alternativamente a opción de engadir toda a formulación manualmente. O usuario escolle unha plantilla para tarefas.

5. A interface web solicita a un servizo web o nome das tarefas taskNames.

6. O usuario cubre a plantilla mentres o sistema almacena os datos que vai cubrindo no estado. Finalmente o usuario preme en xerar código.

7. O sistema devolve o código xerado e móstrao.

8. O sistema solicita a escolla dunha representación e a cumprimentación dos datos.

9. O sistema mostra as variables obtidas do código para mapear a representación escollida anteriormente.

10. O sistema almacena a KPI chamando ao servizo web encargado desa tarefa.

11. O sistema envía o código xerado a Hadoop para que realice o traballo.

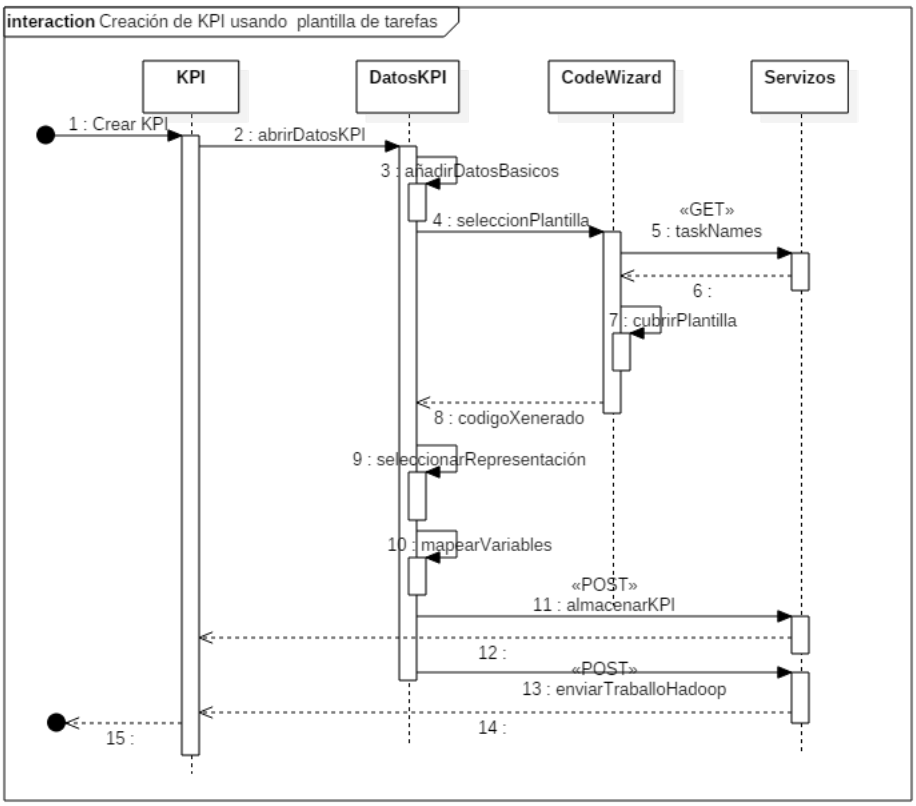

Figura 5.19: Diagrama de secuencia: Creación de KPI usando plantilla de tarefas

Mover unha representación e almacenar o dashboard

Na figura 5.20 describimos o diagrama de secuencia para mover unha representación do dashboard e realizar o seu almacenamento. Os pasos son os seguintes:

1. O usuario move unha das representacións existentes no dashboard a unha nova posición.

2. A interface delega a responsabilidade de actualizar a nova posición para a representación no compoñente RepresentationHandler

3. Obten a representación (que será a mesma que tiña) sobre a cal se vai a mapear os datos de novo e devolve dita representación na nova posición.

4. O usuario preme no botón para salvar o dashboard. A interface web realiza a chamada ao servizo web encargado de realizar o almacenamento.

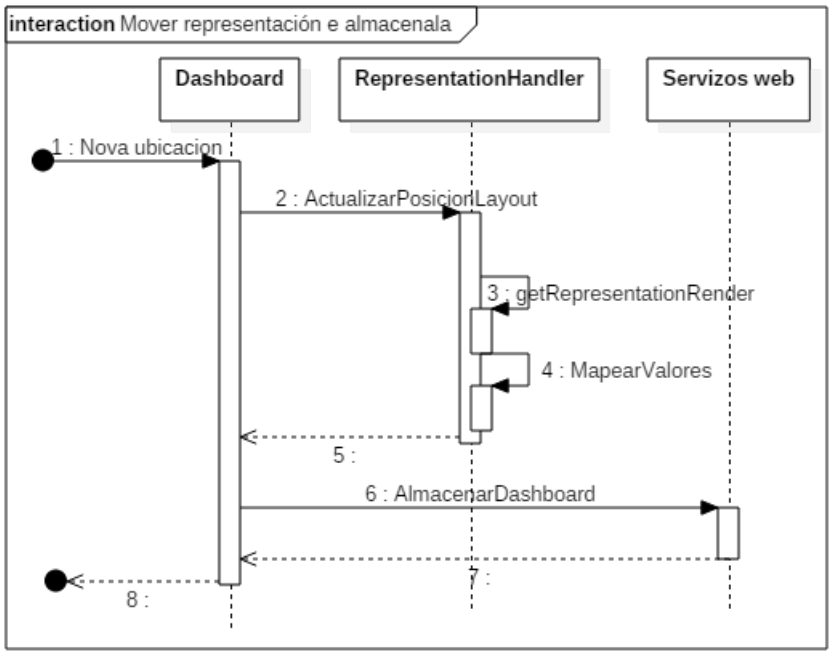

Figura 5.20: Diagrama de secuencia: Mover representación e almacenar o dashboard

Redimensionar unha representación e almacenar o dashboard

Na figura 5.20 amosase o diagrama de secuencia para redimensionar unha representación e alacenar o dashboard dun usuario. Para este caso, realízanse as mesmas operacións que no caso de mover a representación, a única variación é que os parámetros a enviar son os da redimensión (escalado) e non os da ubicación.

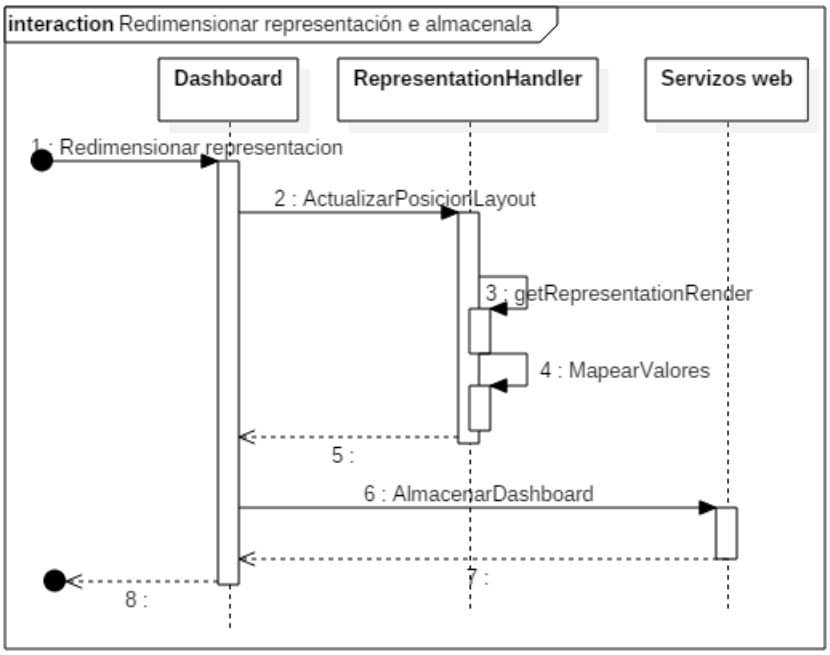

Figura 5.21: Diagrama de secuencia: Redimensionar representación e almacenar o dashboard

5.4. Deseño da interface

Finalmente móstrase aquí o proceso de deseño seguido para a creación das vistas da interface web. Este é un proceso moi importante debido a que se a interface web non se adapta ao usuario pode levar ao proxecto ao fracaso, polo que é importante matizar o deseño e as discusións co cliente para deste xeito podelo adaptar o máximo posible ás súas expectativas.

Actualmente están en continúa expansión o uso das WebAp, que consisten en aplicacións executadas polo navegador, facendo uso de HTML, CSS e JavaScript, e que dan un aspecto de aplicación como as usadas en tablets ou dispositivos móbiles. Neste proxecto empregarase este tipo de tecnoloxías para o seu desenvolvemento.

Inicialmente realizouse un storyboard, unha sucesión de gráficos sen demasiado detalle, para amosar o aspecto ao cliente e poder entender á súa vez se as funcionalidades amosadas son as entendidas realmente, para despois levar a cabo a implementación real desta interface.

5.4.1. Storyboard

O storyboard permite obter un punto de vista de como vai ser a aplicación, para mostrar para ao cliente e realizar cambios a partir dela, sen apenas consumir tempo nin recursos.

Por outra parte permite que o cliente poida opinar sobre o boceto, e pedir os cambios que considere oportunos. Tamén serve como primeira idea para saber se realmente se comprendeu o que quería o cliente que realmente se construira, xa que no comezo dun proxecto software e habitual que haxa equivocos a hora de saber o que se quere realmente construír. As figuras dende a 5.22 a 5.26 recollen as pantallas polas que se debería pasar:

Figura 5.22: Storyboard: Ventá de administración da KPI

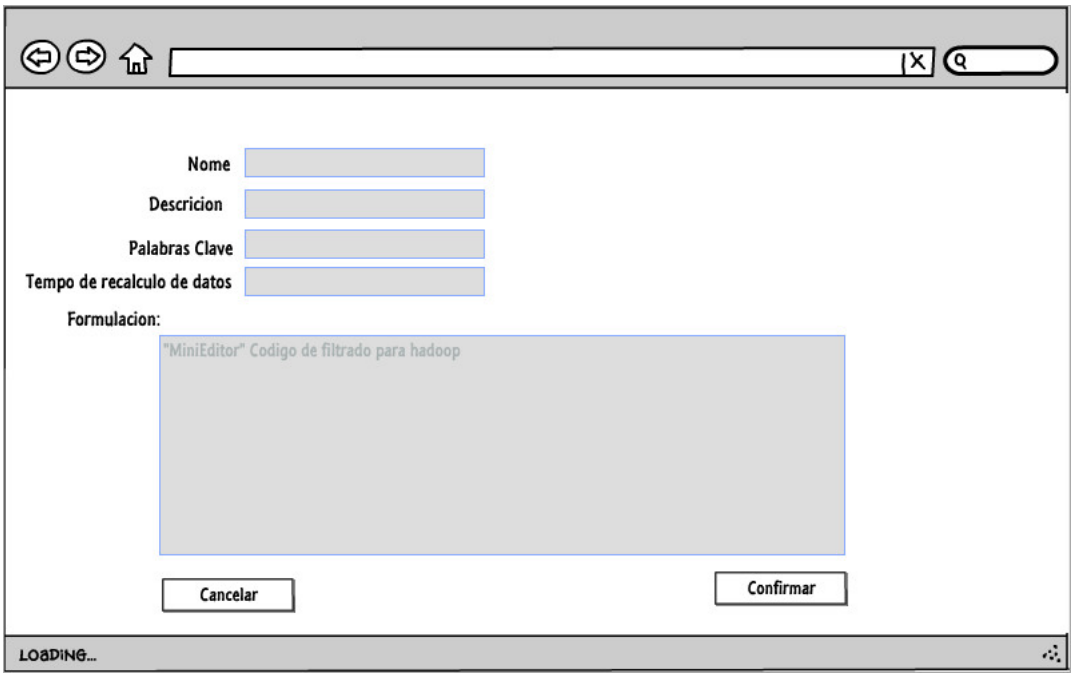

Figura 5.23: Storyboard: Ventá de creación de KPIs Xenericas

Figura 5.24: Storyboard: Lista de KPIs Especificas

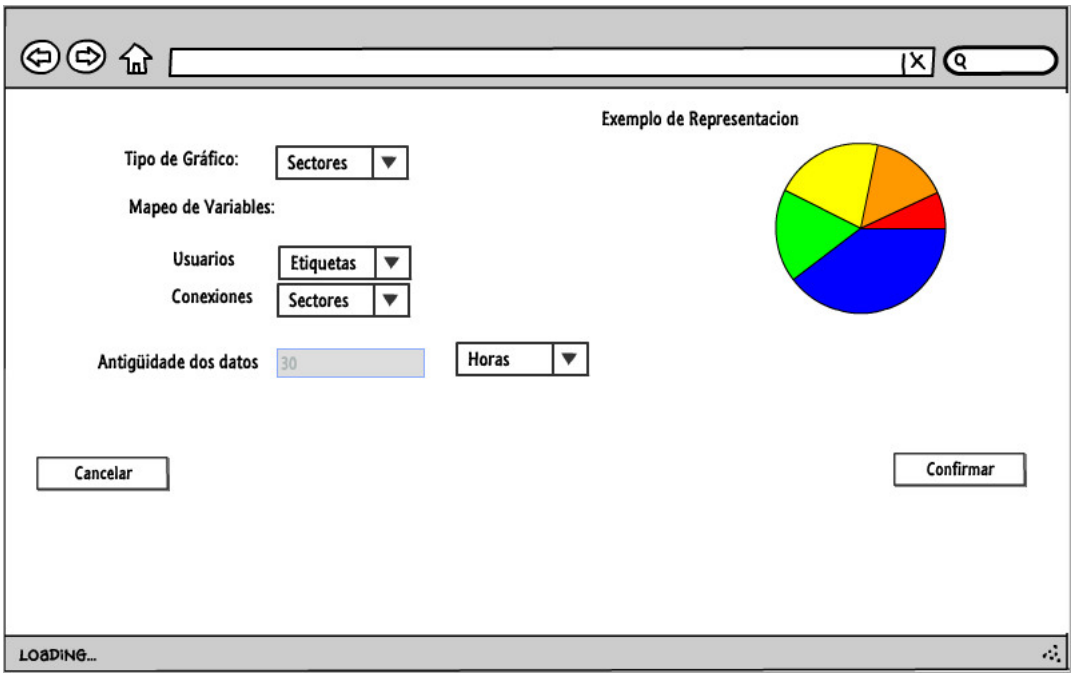

Figura 5.25: Storyboard: Ventá de creación de KPIs Especifica

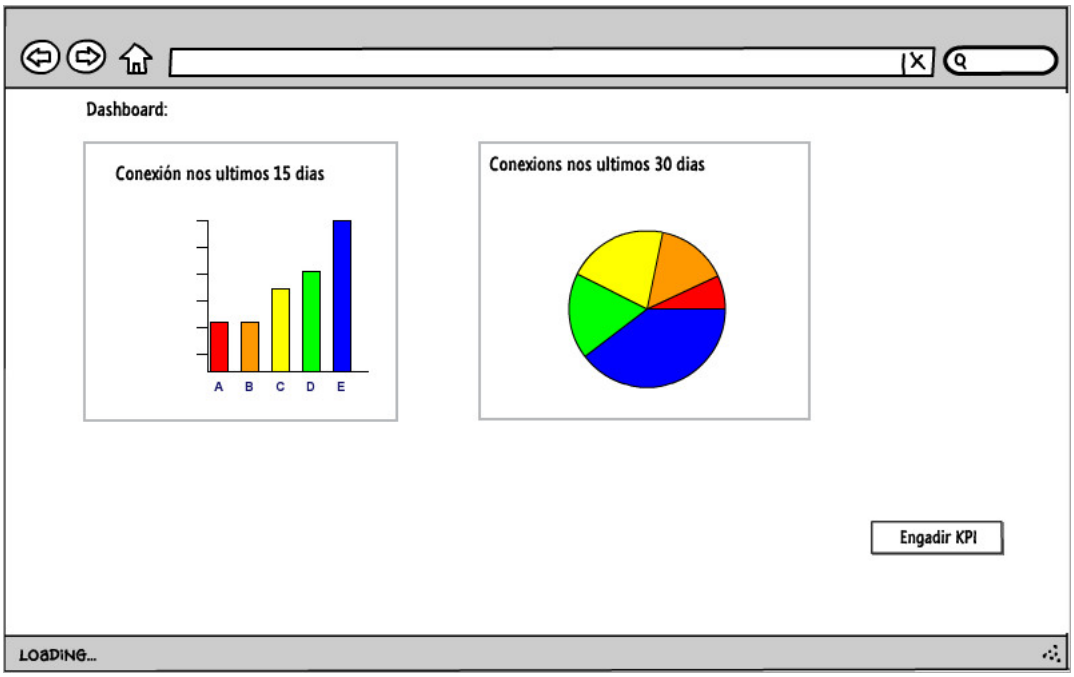

Figura 5.26: Storyboard: Ventá do Dashboard

Unha vez finalizado o storyboard, tivéronse unha serie de reunións cos clientes para aclarar detalles e ver as posibles melloras de cara á implementación da

interface web e sobre todo das correccións sobre as interpretacións dentro do equipo de desenvolvemento. Atopáronse os seguintes problemas:

- Mala interpretación polo equipo de desenvolvemento sobre o que é unha KPI Xenérica e Especifica. A KPI Xenérica refírese á KPI cos seus datos básicos xunto coa formulación, mentres que a KPI Específica se refire ás distintas representacións que pode ter. Pola súa banda, o equipo de desenvolvemento estabao a interpretar como dous conceptos totalmente independentes.
- Dende a parte de administración das KPI non ten sentido decidir que KPI aparece no dashboard polo que o botón de representar na parte de administración non debe aparecer nese lugar, debido a que o interesante é poder engadir dende o propio dashboard e non ter que estar cambiando entre os diferentes apartados da aplicación.

Unha vez analizados os problemas, e entendido mellor en que consiste o proxecto, realizaranse as solucións e melloras na interface web real, sobre todo dende un punto de vista de usabilidade, xa que o cliente reclamou a escaseza de iconas integradas nas táboas por cada KPI para poder editala, eliminala ou simplemente ir a sección de representacións desta KPI para xestionalas.

5.4.2. Interface web implementada

Para a versión implementada adoptouse por un enfoque bastante actual, onde se verá que a maior parte dos compoñentes non son nativos, se non que son recollidos da libraría Material-UI, para dar un aspecto máis próximo ao dunha aplicación móbil. As figuras 5.27 mostra o panel de administración con unha KPI no sistema.

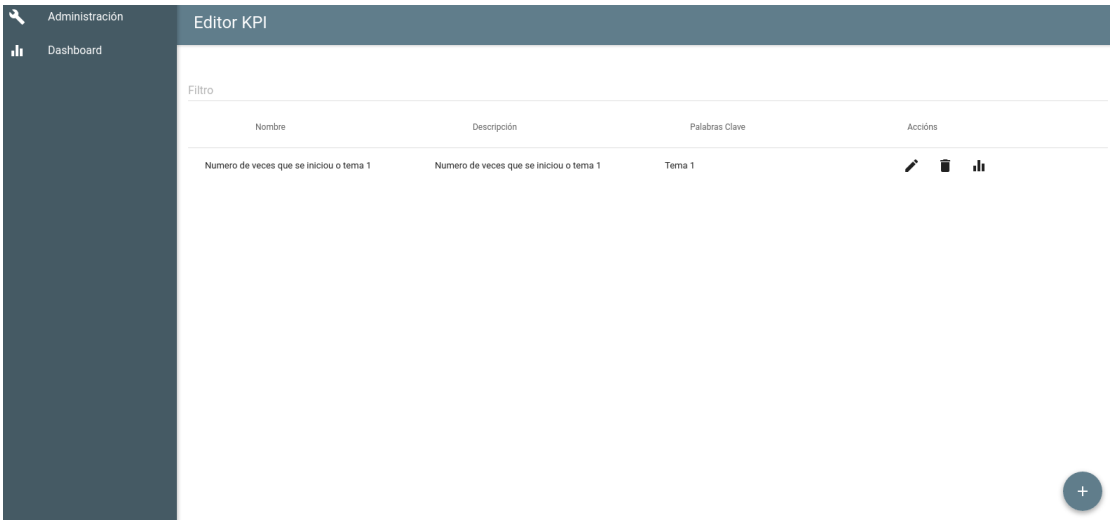

Figura 5.27: Interface: Ventá de Administración

As figuras 5.29 a 5.31 mostrán as vistas durante a primeira parte (datos básicos e formulación) do proceso de creación dunha nova KPI:

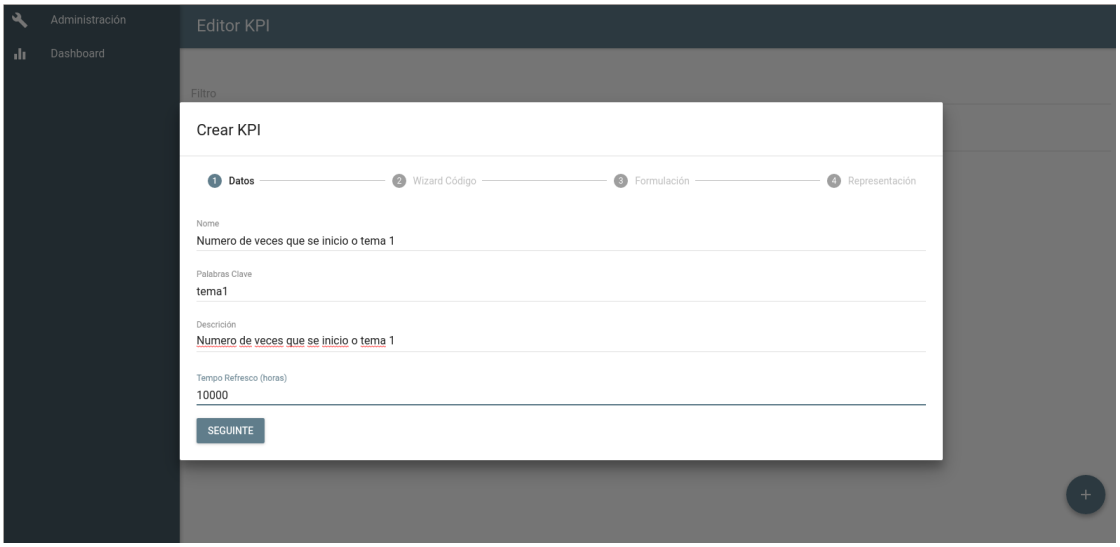

Figura 5.28: Interface: Datos básicos na creación dunha KPI

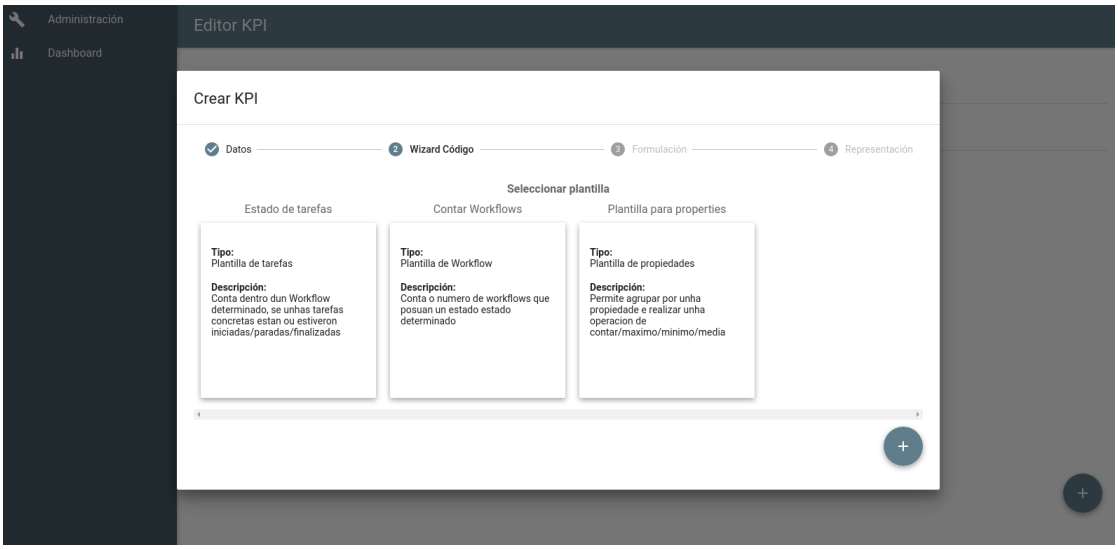

Figura 5.29: Interface: Plantillas para a creación dunha KPI

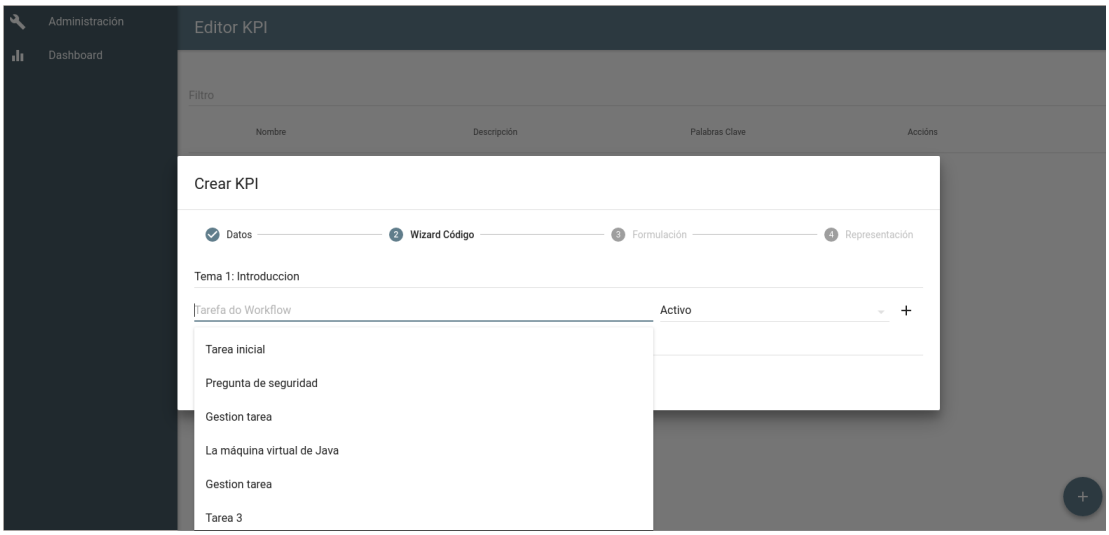

Figura 5.30: Interface: Plantilla de tarefas, autocompletado

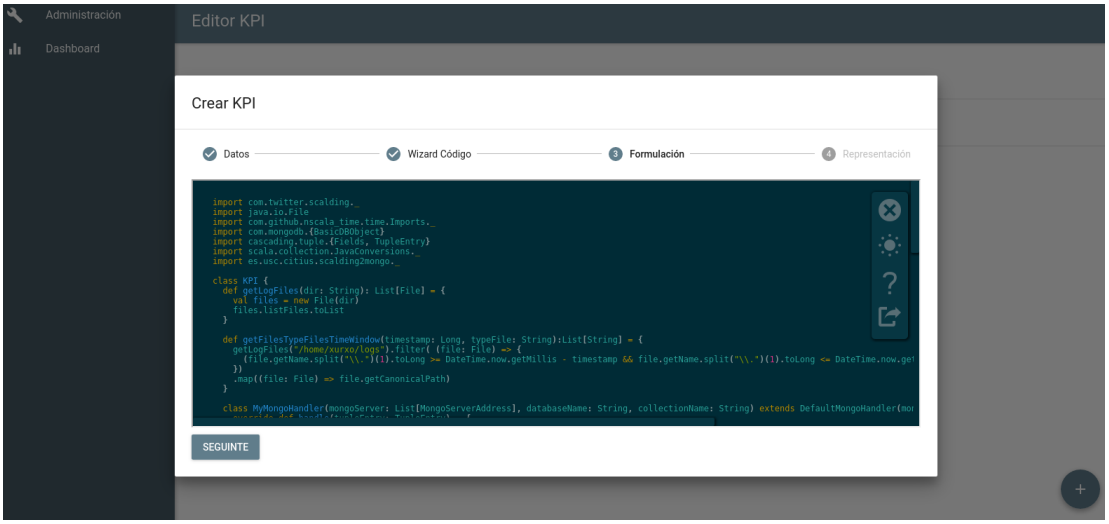

Figura 5.31: Interface: Editor de Scalding integrado, Scalakata

As figuras restantes mostran as partes relativas as representacións dunha KPI. Onde se pode observar a parte da creación e visionado das representacións durante a creación/edición dunha KPI (figura 5.32 e 5.33), mentras que na figura 5.34 se mostra as representacións dentro do dashboard.

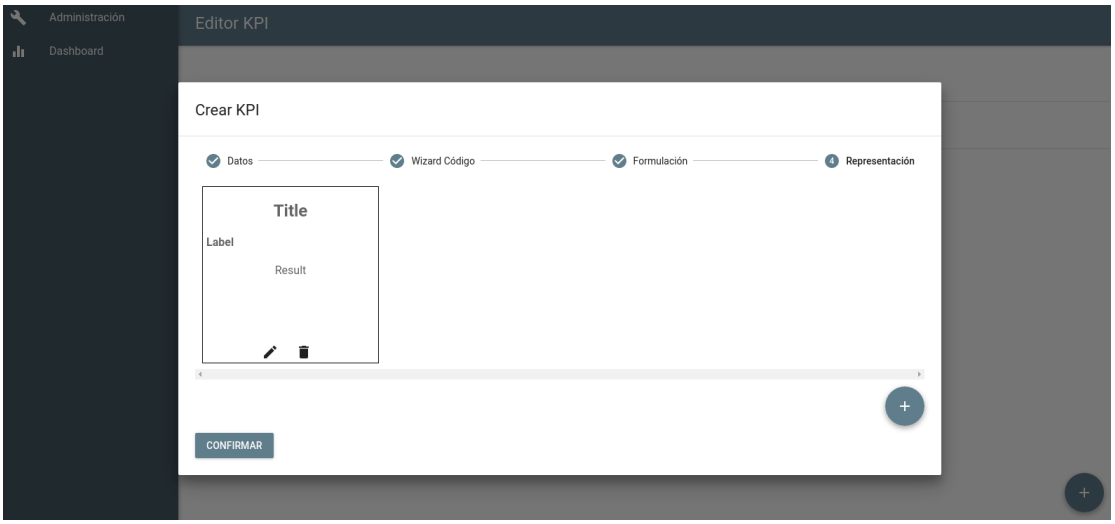

Figura 5.32: Interface: Lista de representacións

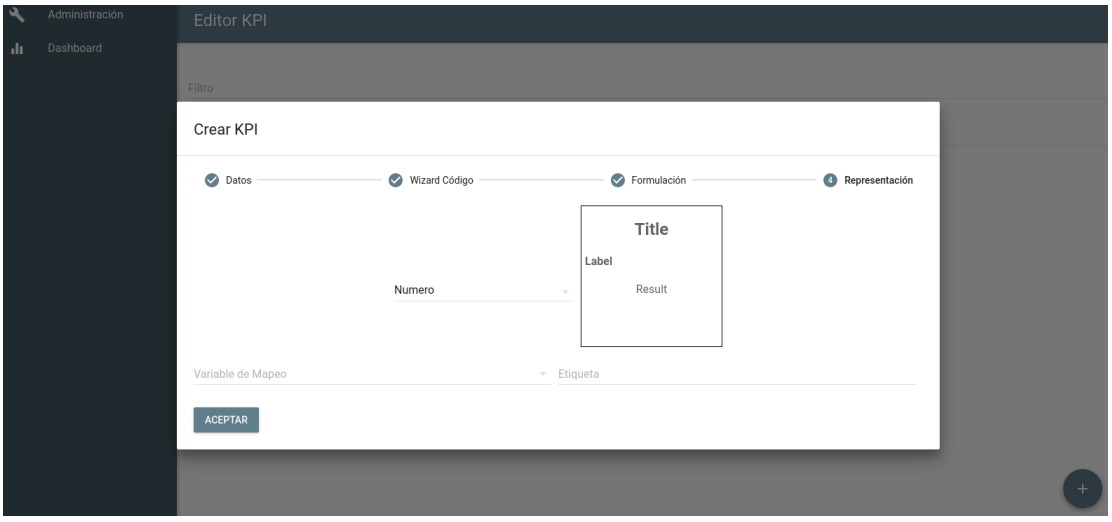

Figura 5.33: Interface: Engadir Representación

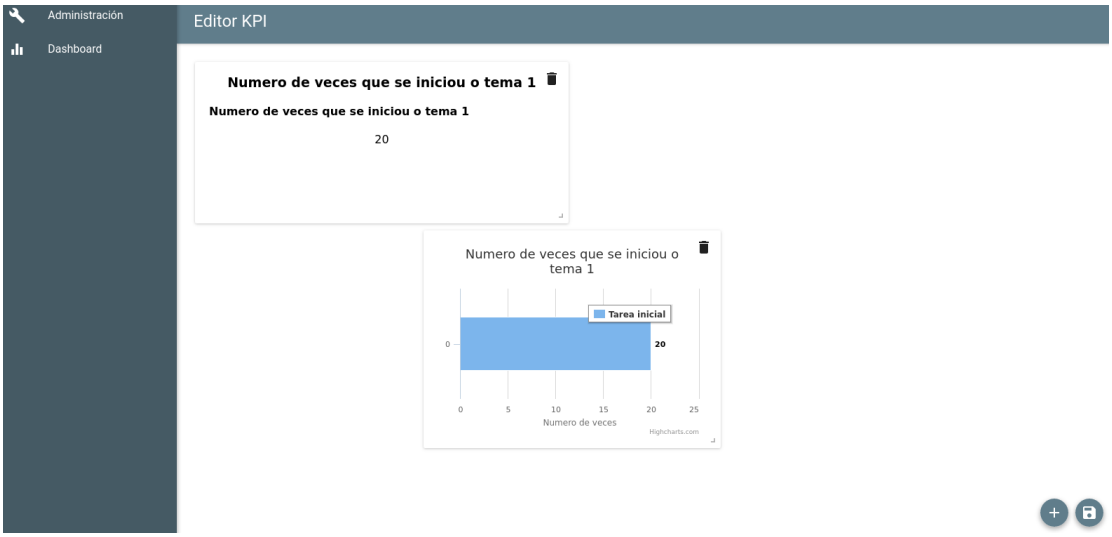

Figura 5.34: Interface: Dashboard

Durante o proceso de implementación da parte gráfica sucederon os seguintes problemas inesperados:

- Integración do Scalakata dentro da interface web, como editor de código escrito en linguaxe Scala, debido a que debería haber unha comunicación entre ambas interfaces para o envío (no caso de estar editando unha KPI) e recepción do código da formulación para enviala posteriormente a Apache Hadoop. Para solucionalo empregouse a integración mediante iframes e o paso de mensaxe entre eles. Para iso foi necesario estudar e manipular o código do editor escrito en Scala, para procesar o paso de mensaxes.

- Outro problema atopado, foi como deducir a partir do código escrito as variables de saída para que o usuario poida realizar o mapeo destas nas correspondentes representacións. Para iso empregáronse expresións regulares xa que se observou que partindo do código da formulación en Scala se pode manipular o nome das variables de saída asignando un nome polo propio usuario dentro do código. En calquera caso, no caso de que esta expresión regular non ofreza ningún resultado, sempre permite ao usuario engadir o nome delas manualmente cando seleccione unha representación.

5.4.3. Validación da usabilidade da interface

Como o desenvolvemento e implementación da interface foi unha das partes do proxecto onde se dedicou máis esforzo e se busca que este sexa o máis usable posible, como se comentou no capítulo de análise de requisitos, en concreto no requisito non funcional **RNF-005** - *Aplicación simple e usable*, realizouse unha fase de validación adicional con algúns usuarios.

Para lograr iso realizouse un cuestionario **SUS** [16], que consiste na obtención dunha puntuación baseada nunha escala de valores. Este cuestionario destaca por tratarse dun tipo de cuestionario sinxelo de facer que non debe provocar o aburrimento do usuario a hora de realizalo.

O proceso a seguir e bastante sinxelo: O usuario contesta a pregunta con un valor entre 1 e 5 antes de haberlle informado de ningún aspecto da aplicación. A contribución de cada unha valerá entre 0 e 4. Para as preguntas impares (positivas) a contribución calculase como o valor da resposta menos 1, mentres que as preguntas pares (negativas) a contribución será 5 menos o valor da resposta. Multiplicase a suma dos resultados por 2,5 para obter o valor global do SUS. O resultado estará entre 0 (nada usable) e 100(moi usable). A continuación móstrase o conxunto de preguntas realizadas no cuestionario:

1. Foi sinxelo engadir unha nova KPI.
2. Sentinme en algún momento perdido.
3. Non me resultou complexo atopar as accións que buscaba.
4. Non me sinto cómodo empregando a aplicación.
5. A visualización do dashboard resulta moi agradable a vista.
6. O uso do sistema resultoume demasiado complexo e sobrecargado de accións.
7. Pareceume un sistema moi sinxelo de empregar.
8. Non volvería a entrar nesta aplicación.
9. Fun capaz de completar as accións realizadas.
10. Non empregaría este sistema habitualmente.

Cadro 5.1: Resultado do cuestionario SUS

	1	2	3	4	5	6	7	8	9	10
Usuario 1	4	1	5	1	5	2	5	1	5	1
Usuario 2	4	2	5	1	5	1	4	1	4	3
Usuario 3	4	2	5	2	5	1	4	1	4	2
Contribución	9	10	12	11	12	10	10	12	11	9
Media	3	3.3	4	3.6	4	3.3	3.3	3.3	3.6	3
Resultado	87.72 %									

As probas de usabilidade SUS establecen que para que unha interface supere a proba de usabilidade esta ten que ter unha porcentaxe de usabilidade superior

ao 80 %. Neste caso pódese observar que a proba foi superada con éxito ao ter como resultado un 87.72 % de usabilidade.

Capítulo 6

Validación e probas

Este capítulo ten como obxectivos comprobar que a aplicación non ten fallos, que o produto finalmente construído está acorde cos desexos do cliente e o que este quería que realmente fixera o software. A programación extrema posúe un enfoque orientado ás probas, polo que aínda que as probas vaian todas a vez neste apartado, compre indicar que se foron realizando durante todo o proceso de desenvolvemento.

6.1. Probas unitarias

Este tipo de probas teñen o cometido de verificar que o software creado funciona correctamente por separado e que o seu comportamento é o esperado. Ademais deben cubrir a maior parte posible do código e debsen ser repetibles e independentes entre si.

Os cadros 6.1 a 6.31 recollen as probas unitarias realizadas ao código. No cadro 6.32 pódese ver a matriz de trazabilidade dos requisitos fronte as probas. Como se pode observar, cumpríronse de forma satisfactoria o 100 % das probas unitarias establecidas para comprobar cada un dos requisitos funcionais establecidos no proxecto.

Cadro 6.1: Proba unitaria: Mostrar as KPIs do sistema

ID	PRU-001
Descrición	O usuario entra na interface web, diríxese a ventá de administración de KPIs.
Requisitos asociados	RF-001
Resultado esperado	O sistema procura as KPIs gardadas en base de datos e mostras nun listado para que o usuario as poida ver.
Resultado	Éxito

Cadro 6.2: Proba unitaria: Creación dunha KPI

ID	PRU-002
Descrición	O usuario entra na interface web, diríxese a ventá de administración de KPIs, a continuación selecciona a creación dunha nova KPI. Introduce os datos básicos correctamente, selecciona unha plantilla, cumpriméntaa correctamente. Elixe unha representación e acepta a creación da KPI
Requisitos asociados	RF-001, RF-002, RF-003, RF-004, RF-005
Resultado esperado	A KPI aparece creada correctamente na lista de KPIs e os resultados son gardados na base de datos.
Resultado	Éxito

Cadro 6.3: Proba unitaria: Dato no campo de refresco da KPI incorrecto

ID	PRU-003
Descrición	O usuario introduce dentro dos datos básicos no campo de refresco da KPI un valor non numérico.
Requisitos asociados	RF-003
Resultado esperado	A interface web avisa do erro que sucedeu para que o usuario poida corrixilo
Resultado	Éxito

Cadro 6.4: Proba unitaria: Algún dos campos do datos básicos da KPI esta valeiro

ID	PRU-004
Descrición	O usuario deixa sen cubrir algún dos campos dos datos básicos.
Requisitos asociados	RF-003
Resultado esperado	A interface informa de que ditos campos non poden ser valeiros e que tipo de dato ten que ir nel.
Resultado	Éxito

Cadro 6.5: Proba unitaria: O Workflow introducido na plantilla de workflows non existe

ID	PRU-005
Descrición	O usuario introduce un workflow que non existe dentro da plantilla de xeración de código para workflows.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web debe informar do erro para que o usuario poida corrixilo.
Resultado	Éxito

Cadro 6.6: Proba unitaria: Xerar código empregando a plantilla de workflows sen introducir ningún workflow

ID	PRU-006
Descrición	O usuario intenta xerar o código empregando a plantilla de xeración automática para workflows sen engadir ningún workflow como condición.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web non deixa continuar e permanece na mesma ventá posto que non engadiu ningún workflow, mostrando o erro sobre o campo de introdución do workflow.
Resultado	Éxito

Cadro 6.7: Proba unitaria: O Workflow introducido na plantilla de tarefas non existe

ID	PRU-007
Descrición	O usuario introduce un workflow que non existe dentro da plantilla de xeración de código para tarefas.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web debe informar do erro para que o usuario poida corrixilo.
Resultado	Éxito

Cadro 6.8: Proba unitaria: Xerar código empregando a plantilla de tarefas sen introducir ningunha tarefa

ID	PRU-008
Descrición	O usuario intenta xerar o código empregando a plantilla de xeración automática para tarefas sen engadir ningunha tarefa como condición.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web non deixa continuar e permanece na mesma ventá posto que non engadiu ningunha tarefa, mostrando o erro sobre o campo de introdución da tarefa.
Resultado	Éxito

Cadro 6.9: Proba unitaria: A tarefa introducida na plantilla de tarefas non existe

ID	PRU-009
Descrición	O usuario introduce unha tarefa que non existe dentro da plantilla de xeración de código para tarefas.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web debe informar do erro para que o usuario poida corrixilo.
Resultado	Éxito

Cadro 6.10: Proba unitaria: Xerar código empregando a plantilla de tarefas sen propiedades sen introducir ningunha propiedade

ID	PRU-010
Descrición	O usuario intenta xerar o código empregando a plantilla de xeración automática para propiedades sen engadir ningunha propiedade como condición.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web non deixa continuar e permanece na mesma ventá posto que non engadiu ningunha propiedade, mostrando o erro sobre o campo de introdución da propiedade.
Resultado	Éxito

Cadro 6.11: Proba unitaria: A propiedade introducido como condición na plantilla de propiedades non existe

ID	PRU-011
Descrición	O usuario introduce unha propiedade que non existe dentro da plantilla de xeración de código para propiedades como condición.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web debe informar do erro para que o usuario poida corrixilo.
Resultado	Éxito

Cadro 6.12: Proba unitaria: A propiedade introducido como propiedade de redución na plantilla de propiedades non existe

ID	PRU-012
Descrición	O usuario introduce unha propiedade que non existe dentro da plantilla de xeración de código para propiedades como propiedade de redución.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web debe informar do erro para que o usuario poida corrixilo.
Resultado	Éxito

Cadro 6.13: Proba unitaria: Non se cubre algún dos select box dentro da plantilla de xeración de código para propiedades

ID	PRU-013
Descrición	O usuario non cumprimenta algún dos select box dentro da plantilla de xeración de código para propiedades.
Requisitos asociados	RF-004, RF-012
Resultado esperado	A interface web debe informar do select box que falta por seleccionar para que o usuario poida corrixilo.
Resultado	Éxito

Cadro 6.14: Proba unitaria: O usuario engade unha representación

ID	PRU-014
Descrición	O usuario crea unha KPI até o punto de engadir unha representación. O usuario selecciona unha das representación e cubre os datos, para a continuación finalizar o proceso.
Requisitos asociados	RF-005
Resultado esperado	A representación aparece engadida correctamente.
Resultado	Éxito

Cadro 6.15: Proba unitaria: O usuario intenta engadir unha representación sen introducir as variables

ID	PRU-015
Descrición	O usuario intenta engadir unha representación sen introducir as variables correspondentes dependendo da representación.
Requisitos asociados	RF-005
Resultado esperado	O sistema informa do erro sobre o campo correspondente.
Resultado	Éxito

Cadro 6.16: Proba unitaria: O usuario engade unha representación tendo variables de saída na formulación

ID	PRU-016
Descrición	O usuario engade unha formulación con variables e a continuación intenta engadir unha representación.
Requisitos asociados	RF-004, RF-005
Resultado esperado	O sistema mostrara as correspondentes variables cando se vaia engadir unha representación.
Resultado	Éxito

Cadro 6.17: Proba unitaria: O usuario engade unha representación sen ter variables na formulación

ID	PRU-017
Descrición	O usuario engade unha formulación sen variables e a continuación intenta engadir unha representación.
Requisitos asociados	RF-004, RF-005
Resultado esperado	O sistema permitira engadir as variables de mapeo da representación manualmente.
Resultado	Éxito

Cadro 6.18: Proba unitaria: O usuario edita unha KPI correctamente

ID	PRU-18
Descrición	O usuario accede ao listado de KPI existentes e accede o panel de edición dunha KPI onde edita algún dos seus datos básicos ou a súa formulación correctamente.
Requisitos asociados	RF-001, RF-006
Resultado esperado	O sistema mostrará a KPI actualizada e enviara de novo o traballo para ser procesado.
Resultado	Éxito

Cadro 6.19: Proba unitaria: O usuario edita unha KPI incorrectamente

ID	PRU-19
Descrición	O usuario accede ao listado de KPI existentes e accede o panel edición dunha KPI onde edita algún dos seus datos básicos ou a súa formulación de forma incorrecta.
Requisitos asociados	RF-001, RF-006
Resultado esperado	O sistema amosara os mesmos erros que nas probas asociadas a creación.
Resultado	Éxito

Cadro 6.20: Proba unitaria: O usuario elimina unha KPI da lista de KPIs

ID	PRU-20
Descrición	O usuario accede a lista de KPIs dispoñibles dentro do sistema e pulsa sobre a icona de eliminar dita KPI.
Requisitos asociados	RF-001, RF-007
Resultado esperado	A lista actualizase desaparecendo a KPI incluso refrescando o sitio web.
Resultado	Éxito

Cadro 6.21: Proba unitaria: O usuario accede a lista de representación dunha KPI

ID	PRU-21
Descrición	O usuario accede a lista de KPIs e accede as representación engadidas para esa KPI.
Requisitos asociados	RF-001, RF-011
Resultado esperado	A interface mostra todas as representacións engadidas para esa KPI.
Resultado	Éxito

Cadro 6.22: Proba unitaria: O usuario edita a representación dunha KPI

ID	PRU-22
Descrición	O usuario accede a lista de representacións dunha KPI e escolle unha representación para editar os seus datos.
Requisitos asociados	RF-011, RF-008
Resultado esperado	O sistema actualiza a información da representación escollida.
Resultado	Éxito

Cadro 6.23: Proba unitaria: O usuario elimina unha representación dunha KPI

ID	PRU-23
Descrición	O usuario accede a lista de representacións dunha KPI e escolle unha representación para eliminar.
Requisitos asociados	RF-011, RF-009
Resultado esperado	O sistema elimina da lista a representación.
Resultado	Éxito

Cadro 6.24: Proba unitaria: O usuario realiza un filtrado sobre a lista das KPIs

ID	PRU-24
Descrición	O usuario accede a lista de KPIs onde dispón dun campo de texto para realizar o filtrado. O usuario escribe un patrón nel.
Requisitos asociados	RF-001, RF-010
Resultado esperado	A interface mostra soamente as KPIs que coincidan co patrón.
Resultado	Éxito

Cadro 6.25: Proba unitaria: Engadir unha representación dunha KPI no dashboard

ID	PRU-25
Descrición	O usuario accede a sección do dashboard e preme sobre o botón de engadir unha KPI a representar. O usuario selecciona a KPI e selecciona a representación que desexa mostrar.
Requisitos asociados	RF-13
Resultado esperado	A interface mostra a representación de dita KPI no dashboard do usuario.
Resultado	Éxito

Cadro 6.26: Proba unitaria: O usuario elimina unha representación do dashboard

ID	PRU-26
Descrición	O usuario accede a sección do dashboard e preme sobre o botón de eliminar dunha representación dispoñible no dashboard.
Requisitos asociados	RF-14
Resultado esperado	A interface eliminara esa representación do dashboard.
Resultado	Éxito

Cadro 6.27: Proba unitaria: O usuario move unha representación existente do dashboard

ID	PRU-27
Descrición	O usuario accede a sección do dashboard e move unha das representacións dispoñible no dashboard.
Requisitos asociados	RF-15
Resultado esperado	A interface mostrara esa representación na nova ubicación.
Resultado	Éxito

Cadro 6.28: Proba unitaria: O usuario redimensiona unha das representacións existentes no dashboard

ID	PRU-28
Descrición	O usuario accede a sección do dashboard e redimensiona unha as representacións dispoñible no dashboard.
Requisitos asociados	RF-016
Resultado esperado	A interface mostrara esa representación redimensionada.
Resultado	Éxito

Cadro 6.29: Proba unitaria: O usuario garda os cambios do dashboard

ID	PRU-29
Descrición	O usuario preme en gardar os cambios dentro do dashboard.
Requisitos asociados	RF-017
Resultado esperado	O sistema almacena o novo dashboard do usuario e o actualizar a páxina e entrar de novo no dashboard todo continua no estado que o deixou.
Resultado	Éxito

Cadro 6.30: Proba unitaria: O usuario inicia sesión con datos válidos

ID	PRU-30
Descrición	O usuario inicia sesión dende a interface web con datos válidos
Requisitos asociados	RF-018
Resultado esperado	O sistema permite o acceso ao sistema.
Resultado	Éxito

Cadro 6.31: Proba unitaria: O usuario inicia sesión con datos inválidos

ID	PRU-31
Descrición	O usuario inicia sesión dende a interface web con datos inválidos
Requisitos asociados	RF-018
Resultado esperado	O sistema denega o acceso ao sistema e pide o usuario que introduza as credencias de acceso correctas.
Resultado	Éxito

Cadro 6.32: Matriz de trazabilidade: Requisitos funcionais - Probas

	Requisitos funcionais																	
	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018
001	X																	
002	X	X	X	X	X													
003			X															
004			X															
005				X								X						
006				X								X						
007				X								X						
008				X								X						
009				X								X						
010				X								X						
011				X								X						
012				X								X						
013				X								X						
014					X													
015					X													
016				X	X													
017				X	X													
018	X					X												
019	X					X												
020	X						X											
021	X									X								
022	X							X										
023									X		X							
024	X									X								
025													X					
026														X				
027															X			
028																X		
029																	X	
030																		X
031																		X

6.2. Probas de integración

Coas probas de integración búscase demostrar que os diferentes módulos da aplicación se unen correctamente, producen os resultados esperados e non existen problemas de interacción de uns con outros. Nos cadros 6.33 a 6.35 atópase a especificación de cada unha das probas de integración realizada. Como se pode observar, todas as probas de integración foron superadas de forma satisfactoria.

Cadro 6.33: Proba de integración: Integración do módulo de xestión de datos - Interface web

ID	PRI-001
Descrición	Abrese a interface web e accedese a sección de administración e dashboard
Resultado esperado	A interface gráfica devolve os datos das KPI e do dashboard.
Resultado	Éxito

Cadro 6.34: Proba de integración: Integración do módulo de Xestión de Hadoop - Interface web

ID	PRI-002
Descrición	Ábresa a interface web e intentase crear unha KPI cos datos correctos
Resultado esperado	O traballo é enviado a Hadoop e procesado por este.
Resultado	Éxito

Cadro 6.35: Proba de integración: Integración dos servizos web HMBOntology-RestAPI - Interface web

ID	PRI-003
Descrición	Abrese a interface e intentase crear unha KPI empregando unha plantilla para Workflows, Tarefas e Propiedades.
Resultado esperado	As axudas de selección presentadas nos campos expoñen os resultados correctos.
Resultado	Éxito

Capítulo 7

Conclusións

O obxectivo deste proxecto pasou pola creación dunha arquitectura orientada a servizos que permita a creación e execución de indicadores claves sobre fluxos de traballo empregando unha interface web avanzada que imite o comportamento dunha aplicación nativa da mellor forma posible. Para o desenvolvemento desta interface web empregáronse tecnoloxías moi relacionadas co mundo web e en pleno auxe na actualidade que favoreceran a manter e adaptar a melloras futuras durante un grande período de tempo debido a que estas tecnoloxías pasarán bastantes anos en constante mellora e explotación.

Mediante esta interface web, preténdese que facilite aos profesionais a creación e estudo de KPIs baseadas en fluxos de traballo. Deste xeito esta interface permite a creación de KPIs mediante a implementación da súa formulación a través de Scalding o que permite unha gran expresividade e a posibilidade de representar calquera tipo de KPI, aínda que disminuindo a facilidade de uso, xa que se precisa de coñecementos previos sobre a libraría e técnicas de Big Data para a súa formulación. Este problema pretendeuse resolver creando algunhas plantillas de xeración automatica de código que realicen as KPIs que se usan con maior frecuencia. Por outra banda, dispón dunha ferramenta de visualización ou dashboard que permite ao usuario un total control das representacións a mostrar permitindo configurar o entorno, é dicir, a posición e tamaño de cada representación que teña representado no dashboard e gardar os cambios en base de datos para que se mostre do mesmo xeito cada vez que o visite.

A creación e uso de servizos web, que cada vez esta máis e máis presentes no desenvolvemento de aplicacións distribuídas con certo grao de complexidade, que permite construír sistemas totalmente modulares e desacoplados, constitúe un dos grandes pilares deste desenvolvemento. Grazas a eles podemos producir aplicacións, mediante o uso de Javascript, permitan diminuír a carga de traballo nos servidores pasando a parte gráfica e de mapeo de datos nas representación ao lado do cliente.

En conclusión, pódese afirmar que a realización deste proxecto é un gran paso para a realización de indicadores chaves do rendemento baseado no análise dos

fluxos de traballo a partir do tratamento dos rexistros empregando técnicas de Big Data enviados a Apache Hadoop, dun xeito simple e sinxelo.

7.1. Problemas atopados

Aínda que os resultados obtidos cumpriron cos obxectivos marcados, encontráronse algúns problemas ao longo do desenvolvemento, que se poden modificar para futuras versións do software:

- Dentro da interface web, atopámonos co problema de integrar un editor de código en Scala para a elaboración da formula da KPI. Isto foi debido a que apenas había editores en Scala que se empregaran dentro dunha plataforma web e a que o desenvolvemento dun levaría demasiadas horas adicionais sobre o total do proxecto e queríase dar unha axuda para o usuario. Para a súa resolución escolleuse usar Scalakata dentro dun iframe e comunicalos mediante paso de mensaxes. Este aspecto non estaba considerado dentro dos requisitos do sistema.
- Integración con Apache Hadoop tanto para o envío dos traballos como o almacenamento en base de datos a través da formulación en Scala, debido a que Apache Hadoop non permite mediante a súa API nativa lanzar traballos.
- Manexo do dashboard dinámico, dentro da interface web, provocou bastantes problemas a hora de sufrir bastantes modificacións e a necesidade de almacenar o último estado no que o usuario o deixou. Este non era un requisito obrigatorio, pero que si seria práctico e agradable para unha mellor experiencia do usuario.

7.2. Ampliación e posibles melloras

Aínda que se alcanzaron os obxectivos propostos na definición do proxecto, este podería ser ampliado e mellorado no futuro co fin de proporcionar maior número de funcionalidades e facilitar aínda máis o proceso de creación e monitorización das KPIs. Algunhas das posibles melloras observadas son as seguintes:

- Seria interesante dispoñer dun planificador para que recalcule os novos resultados das KPIs con unha certa frecuencia especificada polo usuario, xa que actualmente so se executa o traballo en Hadoop ao confirmar a creación ou se esta sufre unha modificación. Para isto poderíase optar por empregar algunha solución xa dispoñible no mercado, como Oozie, que permite xestionar a execucion de traballos en Hadoop de forma declarativa; ou crear unha implementación propia.

- Para mellorar o editor Scalakata, seria interesante poder, no caso de empregar as plantillas xeradoras de código, autocompletar as propiedades, tarefas ou workflows empregados para a súa xeración.
- No dashboard, permitir o autogardado sen ter que manualmente premer sobre o botón de gardar, se non que o faga cada vez que se realiza un cambio, cando se pecha a ventá ou simplemente se se cambia de sección, xa que isto facilita que o usuario non teña que lembrarse de confirmar os cambios.
- Engadir maior número de plantillas de xeración automática de código, para facilitar a creación da formulación para os usuarios, e de representacións para as KPI.

Apéndice A

Manual técnico

O manual que se recolle a continuación ten como obxectivo clarificar o proceso de instalación dos diferentes módulos desenvoltos neste proxecto. Para facelo máis claro móstrase o proceso de instalación de cada un dos módulos por separado.

A.1. Requerimentos

Para a instalación de tódolos módulos do proxecto precisarase dunha máquina cos seguintes paquetes de software instalados:

- **Java 8**, descargable desde *https://www.java.com/es/download*
- **Npm**, descargable desde *https://www.npmjs.com/package/npm*
- **Sbt**, descargable desde *http://www.scala-sbt.org/download.html*
- **MongoDB**, descargable desde *https://www.mongodb.com*
- Un contedor de servlets, por exemplo Apache Tomcat, descargable desde *http://tomcat.apache.org/*
- **Apache Hadoop 2.7.2**, descargable desde *http://hadoop.apache.org/*
- **Maven**, para compilar o módulo de xestión de datos e o módulo de xestión de Hadoop

Non se explicarán neste documento as configuracións destes aplicativos, pois dependendo do sistema onde se instalen estas poden variar. Sen embargo, na web pódese atopar todo tipo de documentación.

A.2. Instalación dos módulos do servidor

Tanto os módulos de xestión de datos (MongoDBServices) como o de xestión de Hadoop (HadoopServices) se distribúen como proxectos maven comprimidos nun ZIP (Modulos.zip). Este ZIP aparte deses dous módulos contén tres carpeta adicionais, a primeira chamada *scalding-develop* que conterá os scripts necesarios para mandar os traballos a Hadoop, unha segunda carpeta chamada *scalding2mongo* que conterá a libraría necesaria tanto para os scripts do scalding-develop para almacenar os resultados en base de datos como para correxir e autocompletar dentro do editor Scalakata e por último unha carpeta chamada *dataDB* que contén os datos das plantillas de código e un usuario inicial para a base de datos. Este ZIP pódese descomprimir onde se desexe.

A.2.1. Configuración e compilación de scalding2mongo e scalding-develop

É recomendable que ambas carpetas se atopen nun mesmo directorio, para realizar a compilación de scalding-develop, xa que esta depende da xeración do paquete .JAR de scalding2mongo. Para realizar a compilación de scalding2mongo é necesario acceder a dita carpeta e realizar os seguintes comandos:

```
$ sbt clean # Limpa o proxecto scalding2mongo
$ sbt compile # Compila o proxecto
$ sbt package # Empaqueta o proxecto nun .JAR
```

Para realizar a compilación de scalding-develop chega con entrar na carpeta scaling-develop e realizar os seguintes comandos, que poden tardar bastante en tempo en completarse:

```
$ ./sbt update #Actualiza os paquetes
$ ./sbt assembly #Realiza a compilación e empaquetamento
```

Importante: Se se desexa que ambas carpetas se atopen en directorios diferentes será necesario modificar dentro de *scalding-develop* o arquivo *build.sbt* a variable **pathScalding2Mongo** pola ruta onde se atope o paquete .JAR (scalding2mongo_2.11-0.1-SNAPSHOT.jar) obtido da compilación e empaquetamento de scalding2mongo.

A.2.2. Modulo de xestión de datos

A base de datos poderase inicializar en calquera directorio. Chega con executar o seguinte comando comando para a súa inicialización, que creara a infraestrutura automaticamente no caso de que non exista:

```
$ mongod --dbpath BD_PATH
```

Unha vez inicializada é necesario crear a base de datos e as coleccións necesarias, para iso iniciamos o cliente e executamos os seguintes comandos para entrar na base de datos e crear as coleccións:

```
$ mongo
> use kpis
> db.createCollection("kpis")
> db.createCollection("usuarios")
> db.createCollection("codeTemplates")
> db.createCollection("hadoop")
```

Para importar as plantillas de código na base de datos e un usuario inicial (nome: "root", contrasinal: "root") hai que realizar os seguintes comandos, sobre os arquivos de tipo json que se atopan dentro da carpeta *dataDB*:

```
$ mongoimport --db kpis --collection usuarios --type json
  --file PathDescompresion/usuarios.json
$ mongoimport --db kpis --collection codeTemplates --type json
  --file PathDescompresion/codeTemplates.json
```

A.2.3. Modulo de xestión de Hadoop

Neste caso será necesario configurar a ruta de acceso ao script eval-tool que se atopa dentro de scalding-develop/scripts dentro do arquivo *properties.config* que se atopa no proxecto HadoopServices, na ruta **HadoopServices/src/main/resources**. Nese arquivo debese modificar o parámetro *eval-tool*:

```
eval-tool=/home/pepe/scalding-develop/scripts/
```

Por outra banda para que funcione todo correctamente é importante ter configurado nas variables de entorno o acceso a Hadoop xa que scalding-develop fai uso del.

A.2.4. Compilación dos modulos e inicialización

Para realizar a súa compilación simplemente hai que entrar dentro da carpeta de cada un deles (MongoDBServices e HadoopServices) e executar os seguintes comandos:

```
$ mvn clean
$ mvn package
```

Estes crearán os .war de cada proxecto que constitúe a aplicación compilada. Para instalar os módulos copiaremos os arquivos .war dentro do contedor de servlet para que distribúa as aplicacións. Unha vez realizado isto as carpetas MongoDBServices e HadoopServices poden ser eliminadas.

A.3. Instalación da interface web

A aplicación web distribúese como un arquivo comprimido en ZIP (Web.zip), onde existen dúas carpetas, a primeira conten a web e a segunda conten o editor en Scala (Scalakata2). Para proceder á instalación deste módulo chega con descomprimilo en calquera lugar que se desexe.

Para realizar a inicialización da interface web deberemos por un lado inicializar o editor de Scalakata e por outro a interface web. No caso do Scalakata, primeiro debese cambiar a variable **pathScalding2Mongo** no **build.sbt** que se atopa dentro da carpeta **Scalakata2** para indicar o path absoluto onde se atopa o .JAR do scalding2mongo, para poder empregar dita libraría no editor. A continuación debese realizar a seguinte secuencia de comandos que compilará e inicializará o Scalakata:

```
$ sbt
> clean
> compile
> project webappJVM
> reStart
```

No caso da interface web, chega con modificar o parámetro **host**, dentro do arquivo config.json, que se atopa no directorio *web*, como no exemplo seguinte:

```
"host": "localhost"
```

Finalmente para instalar as dependencias e iniciar o servizo realizanse os seguintes comandos dentro da carpeta *web*:

```
$ npm run serve
```

Finalmente para acceder á interface web chega con ir o enlace no navegador web: **http://host:3000**

Apéndice B

Manual de usuario

Para simplificar o aprendizaxe do usuario, nos seguintes apartados móstrase unha serie de nocións básicas para un uso correcto da aplicación.

B.0.1. Manual do usuario final

Aínda que a interface web se deseñou de forma que resultase simple e sinxela realizar accións sobre ela, é recomendable realizar unha breve descrición de dous caso de uso típicos que se poden dar con moita frecuencia. Neste manual realizaranse as accións de creación dunha KPI debido a que durante a edición suceden practicamente os mesmos pasos a excepción de que non se utilizan plantillas e uso do dashboard.

Creación dunha KPI

A primeira acción realizada polo usuario é a de seleccionar dentro da sección de administración da interface web o botón situado na esquina inferior dereita (simbolo +) como se pode ver na figura B.1, inicialmente no primeiro uso a lista estará vacia:

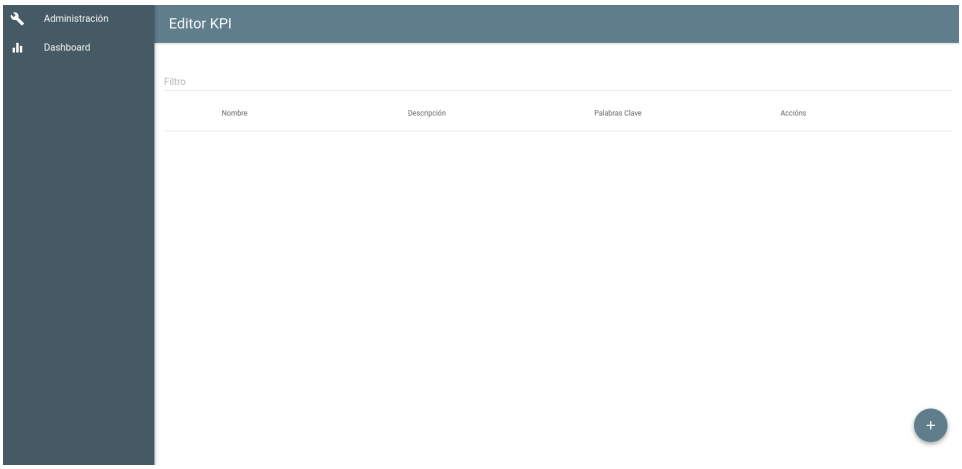

Figura B.1: Interface: Ventá de Administración sen elementos

A continuación abrirase unha ventá onde pedira os datos básicos para a KPI que queremos crear:

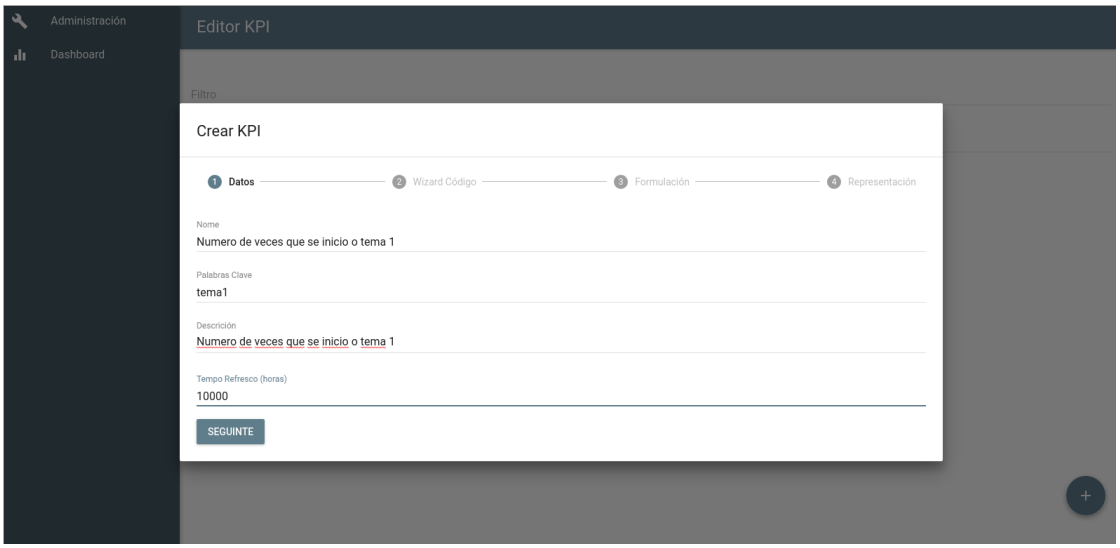

Figura B.2: Interface: Datos básicos na creación dunha KPI

Se continuamos o proceso entraremos na parte de xeración de código, este paso podemos facelo guiándonos das plantillas de código proporcionadas polo sistema ou xeralo de forma manual premendo no botón da esquina inferior esquerda desa venta (símbolo +). Para este caso empregaremos unha plantilla que actúa sobre tarefas, que se accede a ela premendo na viñeta correspondente, pero cabe destacar, que as plantillas son bastante sinxelas de cubrir xa que proporcionan atallos e recomendacións sobre os campos como se pode ver nas seguintes figuras:

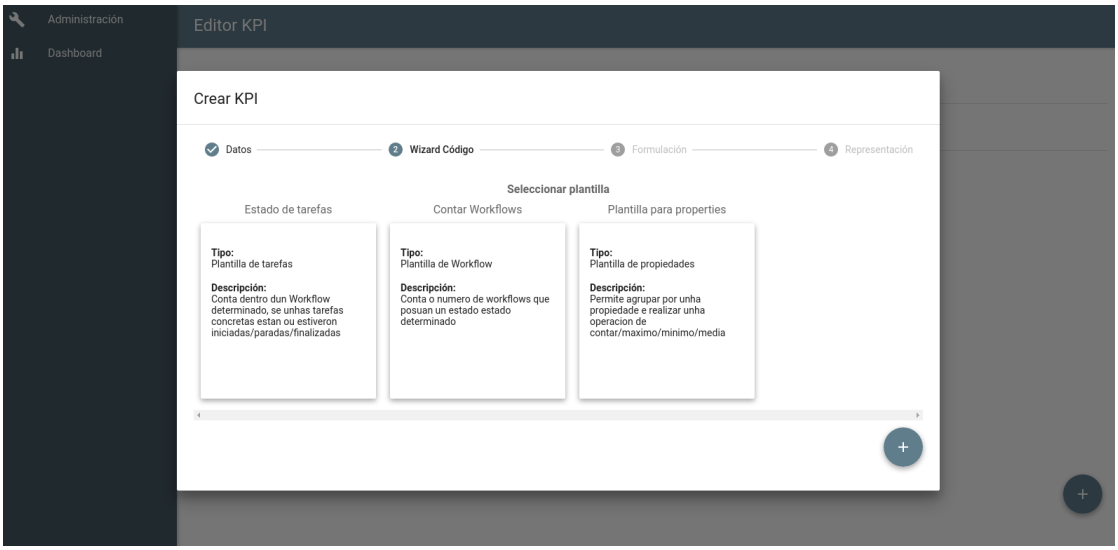

Figura B.3: Interface: Plantillas para a creación dunha KPI

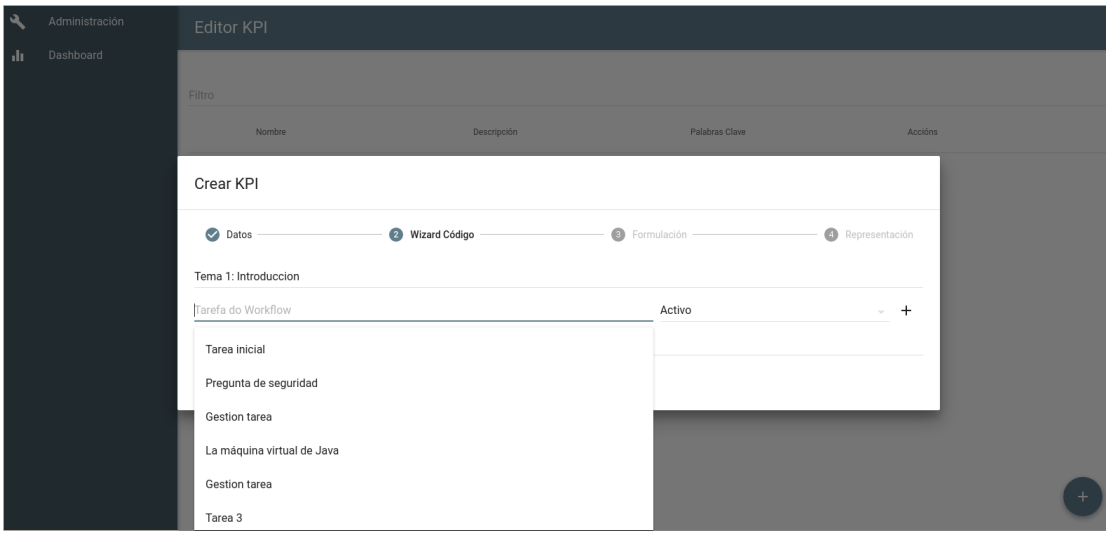

Figura B.4: Interface: Plantilla de tarefas, ofrecendo recomendacións

Unha vez cuberto os datos mostrarase o código da formulación xerado en Scalding por se se quixese realizar, algunha modificación ou acción,que non estivese contemplada na plantilla empregada:

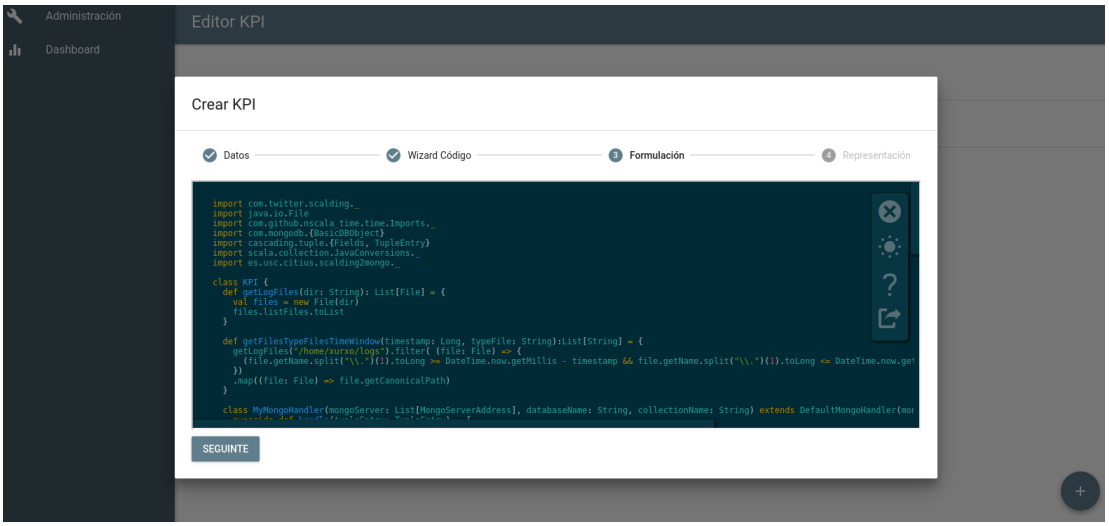

Figura B.5: Interface: Editor de código, co código autoxerado

O seguinte paso é a selección das representacións para esta KPI, por defecto durante a creación este lévanos directamente ao apartado para seleccionala, no caso da edición levaríanos un paso previo que é o listado das representacións das que xa dispón dita KPI.

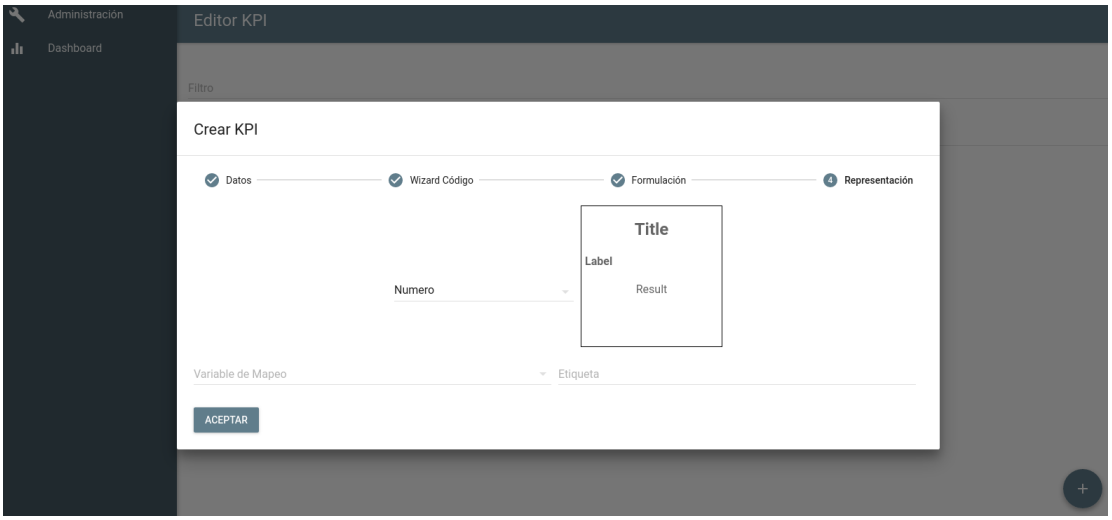

Figura B.6: Interface: Engadir Representación

Onde podemos elixir a representación que mellor se adapte a KPI que se esta creando de entre as dispoñibles no sistema. Para cada representación pediranos un mapeo de variables, que consiste en determinar que resultados se van a mostrar na representación e como se van a mostrar, co obxectivo de que a interface teña constancia destas variables e as detecte automaticamente. No caso de realizar o

código manualmente, é recomendable renomear no código as columnas de saída que proporcionará Hadoop, engadindo antes de escribir a saída unha liña deste estilo:

```
.rename(('TaskID, 'Count) -> (('Tarefa, 'Conteo)))
```

En caso de non realizar isto, pediranos que as escribamos manualmente para poder realizar o mapeo de datos. Se confirmamos a creación da representación levaranos a lista mencionada anteriormente, onde poderemos engadir mais representación, editalas ou borrar as existentes:

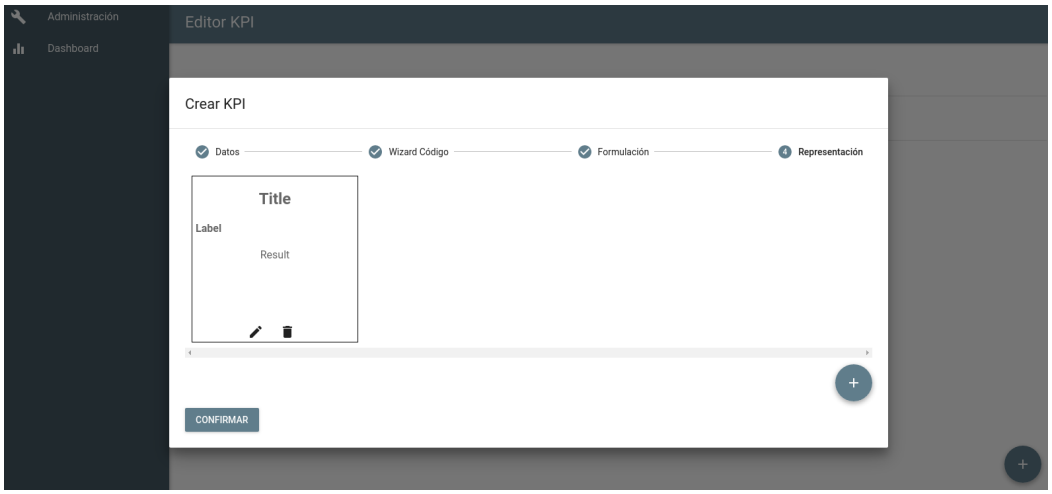

Figura B.7: Interface: Lista de representacións

Unha vez confirmada a creación da KPI enviará o traballo a Hadoop e aparecéranos na lista de KPIs dispoñibles no sistema:

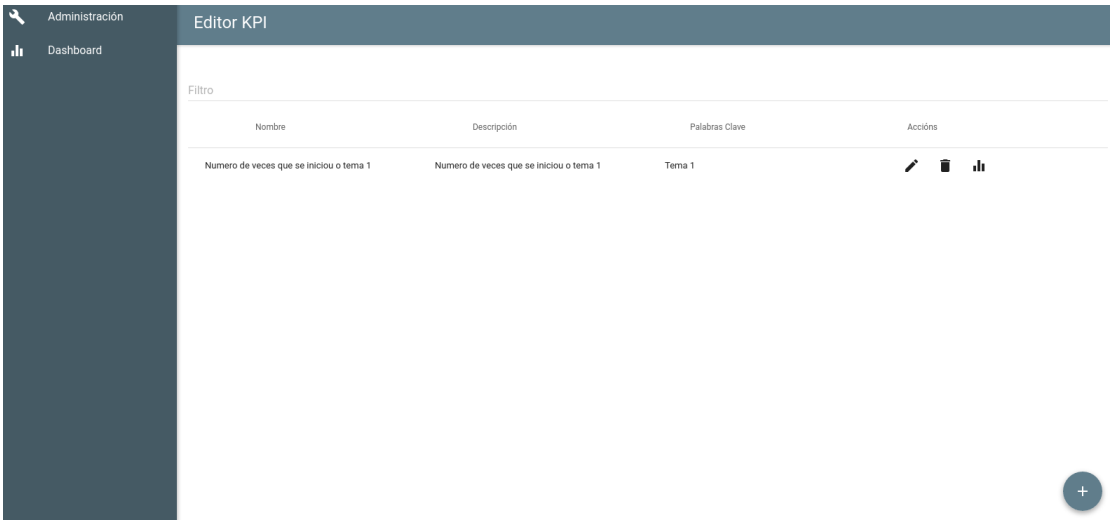

Figura B.8: Interface: Ventá de Administración coa KPI engadida

Cabe destacar que no caso de dipoñer dun maior número de KPIs se queremos buscar algunha concreta poderemos usar o campo de texto superior para introducir un patrón de filtrado, para que so nos mostre as KPIs que coincidan con dito patrón. Ademais, pódese observar que existen 3 iconas na lista da KPI: Unha para editala, outra para borrala e outra para ir directamente ao listado das súas representacións.

Utilización do dashboard

Dentro da sección do dashboard prememos, igual que na sección de administración de KPIs, sobre o botón con símbolo + para engadir unha representación no dashboard, que abrirá unha ventá, onde se introduce o nome da KPI a representar axudado por unha suxestión a medida que se vai escribindo, como se pode observar na seguinte figura:

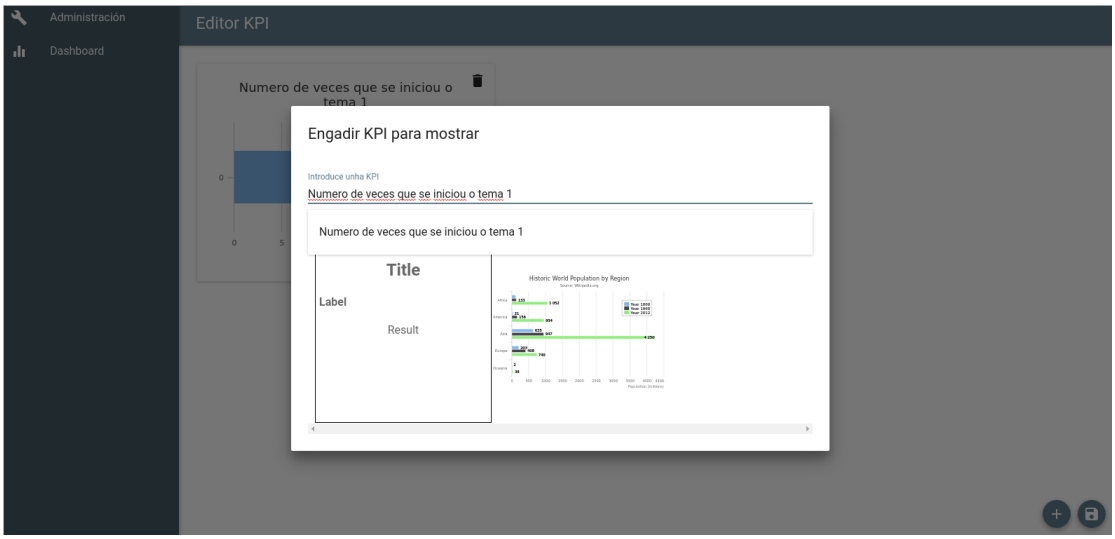

Figura B.9: Interface: Ventá para engadir unha representación ao dashboard

Chega con seleccionar unha delas para que a mostre no dashboard con datos reais.

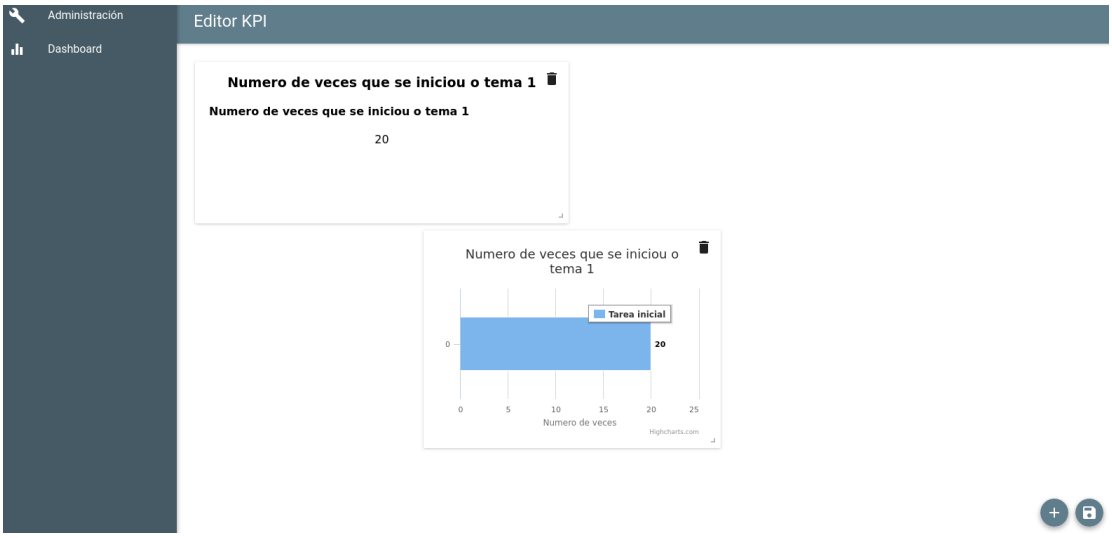

Figura B.10: Interface: Dashboard coa representación engadida

Como se pode observar na figura anterior, a representación engadida pódese mover se se preme sobre ela e se move, e redimensionar se se preme dende a esquina inferior esquerda de dita representación e se arrastra. Finalmente para almacenar o estado actual do dashboard debese premer sobre o segundo botón que aparece na esquina inferior esquerda que posúe unha icona dun disquete. Para eliminar esa representación do dashboard débese premer na papeleira que contén cada unha das representacións e posteriormente salvar os cambios.

Bibliografía

[1] Adela del-Río-Ortega, Manuel Resinas, Antonio Ruiz-Cortés, "Defining Process Performance Indicators: An Ontological Approach", *Universidad de Sevilla, Spain.*

[2] A Scala API for Cascading *https://github.com/twitter/scalding*

[3] What Is Apache Hadoop? *http://hadoop.apache.org/*

[4] Scalakata, *https://github.com/MasseGuillaume/ScalaKata2*

[5] *Guía de los Fundamentos de la Dirección de Proyectos 3ª edición*, Project Management Intitute, 2004

[6] IBM: Definición de casos de uso v11 *http://www.ibm.com/support/knowledgecenter/es/SSWSR9_11.0.0/com.ibm.pim.dev.doc/pim_tsk_arc_definingusecases.html*

[7] INTECO: Guía Práctica de la Gestión de la Configuración, *https://www.incibe.es/file/NRDmviQoTbItSH7Apl5n5Q*

[8] Flux Overview *https://facebook.github.io/flux/docs/overview.html*

[9] The Standish Group: The CHAOS Report, 2014, *https://www.projectsmart.co.uk/white-papers/chaos-report.pdf*

[10] Estudo salarial do sector TIC en Galicia 2015-2016 *http://www.vitaedigital.com/download/NTY2/Guia_Salarial_Sector_TI_Galicia_2015-2016.pdf*

[11] M. Jørgensen, K. Moløkken-Østvold: How large are software cost overruns? A review of the 1994 CHAOS report, Information and Software Technology, 48(4):297–301, 2006.

[12] Library VS. Framework? *http://www.programcreek.com/2011/09/what-is-the-difference-between-a-java-library-and-a-framework/*

[13] Why React? *https://facebook.github.io/react/docs/why-react.html*

[14] What is Virtual DOM? *https://medium.com/tony-freed-consulting/what-is-virtual-dom-c0ec6d6a925c#.ps53o7yyw*

[15] Martin Fowler. *UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.*

[16] P.W. Jordan, B. Thomas, I.L. McClelland, and B. Weerdmeester. *Usability Evaluation In Industry. Taylor & Francis, 1996.*