ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

jGALiWeather

Xeración automática de prognósticos meteorolóxicos a curto prazo con técnicas avanzadas de xeración de linguaxe natural.

Autor:
Diego Iglesias Freire

Directores:
Alberto Bugarín Diz
Alejandro Ramos Soto

Grao en Enxeñaría Informática

Julio 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. Alberto Bugarín Diz, Catedrático da Universidade de Santiago de Compostela, e **D. Alejandro Ramos Soto**, Investigador do Centro Singular de Investigación en Tecnoloxías da Información(CiTIUS) da Universidade de Santiago de Compostela,

INFORMAN:

Que a presente memoria, titulada *jGALiWeather: Xeración automática de prognósticos meteorolóxicos a curto prazo con técnicas avanzadas de xeración de linguaxe natural.*, presentada por **D. Diego Iglesias Freire** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo a nosa dirección no Centro Singular de Investigación en Tecnoloxías da Información(CiTIUS) 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 xullo de 2016:

O director, O codirector, O alumno,

Alberto Bugarín Diz Alejandro Ramos Soto Diego Iglesias Freire

Agradecementos

A Andrea, por estar aí todos estes anos, polo seu apoio continuo e, sobre todo, pola súa paciencia.

Á miña familia, por seguir estando aí a pesar dos meus cambios de humor.

A Alberto e Alejandro, por ser os mellores titores que un alumno poidera desexar e por estar sempre a miña disposición.

Aos meus compañeiros de carreira, que converteron estes 5 anos nunha experiencia fabulosa.

Aos meus amigos, por perdoar as miñas continuas ausencias.

Índice xeral

Índice de figuras

Índice de cadros

Capítulo 1

Introdución

A ciencia de datos (*Data Science*) sempre intentou extraer coñecemento de grandes volumes de datos mediante a súa interpretación utilizando métodos analíticos e técnicas de visualización. Sen embargo este coñecemento extraído no proceso de análise presentase aos usuarios dunha maneira que non sempre é facil de interpretar e, de feito, moitas veces é necesaria unha formación previa para chegar ao seu entendemento.

Non solemos pensar na linguaxe coma unha interface de usuario, pero se nos paramos a pensalo dámonos de conta de que realmente o é. Todos utilizamos a linguaxe para plasmar os nosos pensamentos e despois envialos a través das nosas voces, teclados, pantallas táctiles, etc. O obxectivo final é sempre o mesmo, que o receptor entenda os nosos pensamentos da mesma maneira que nos o facemos.

Polo tanto, por que non vamos utilizar a linguaxe natural para comunicar toda esa información xerada por un analista de datos? A disciplina de xeración de linguaxe natural (NLG) busca facer isto posible xerando texto en linguaxe natural, que sexa coherente, para satisfacer un ou máis obxectivos comunicativos [3, 5].

Sen embargo, a NLG non se aplica só no ámbito aquí descrito senón que se utiliza nunha cantidade inxente de sistemas dispares. Por exemplo, unha das tecnoloxías nas que se están facendo máis avances hoxe en dia son os asistentes persoais. Sistemas coma Apple Siri o Google Now permítennos falar cos nosos dispositivos, que son capaces de entender as preguntas que lles facemos e xerar unha resposta útil en linguaxe natural. Todo este proceso nútrese das diferentes técnicas de NLG.

Entre as diversas aproximacións existentes dentro da NLG, destaca especialmente a rama especializada na xeración de textos a partir de conxuntos de datos

numéricos coñecida coma "data-to-text" (D2T) [4], que nos últimos anos está experimentando un importante auxe científico e comercial debido á cada vez maior cantidade de datos que os expertos deben manexar e interpretar nos seus respectivos dominios. Os sistemas D2T axudan aos expertos e usuarios a aforrar esforzo e tempo na xeración de textos. Nos últimos anos hanse desenvolvido múltiples sistemas de xeración automática de novas en medios de comunicación, e-saúde, supervisión e alertas en procesos industriais e incluso, sistemas de información medioambiental e meteorolóxica. A este ultimo grupo pertence a aplicación desenvolvida neste proxecto, **jGALiWeather**.

jGALiWeather é unha aplicación D2T que é capaz de xerar predicións meteorolóxicas textuais para varias variables meteorolóxicas coma o estado do ceo ou a temperatura. Esta aplicación esta baseada na aplicación GALiWeather desenvolvida no CiTIUS para cubrir unha necesidade real do servizo galego de meteoroloxía (MeteoGalicia). MeteoGalicia necesitaba unha forma de xerar predicións en linguaxe natural para os 314 concellos galegos, debido á imposibilidade, por razóns de esforzo e tempo, de elaboralas por parte dos meteorólogos.

Figura 1.1: GALiWeather en uso na web de MeteoGalicia).

GALiWeather emprega técnicas de soft computing e xeración de linguaxe natural a través do uso de modelos de texto nos dous idiomas actualmente operativos (galego e español). jGALiWeather busca mellorar diversos aspectos da xeración de textos realizada por GALiWeather utilizando técnicas NLG máis sofisticadas.

Intégranse melloras metodolóxicas e tecnolóxicas que permitirán superar as limitacións da actual GALiWeather nos aspectos relativos á xeración de linguaxe natural, tanto desde o punto de vista tecnolóxico, de mantemento e funcional (especificamente no soporte a novos idiomas).

1.1. Obxectivos

O obxectivo xeral de jGALiWeather é o deseño, implementación e proba dunha nova aplicación na que se van aplicar técnicas de xeración de linguaxe natural máis sofisticadas cas orixinalmente utilizadas en GALiWeather, coa utilización de recursos de xeración de linguaxe natural (NLG) coma a libraría Java SimpleNLG. Este obxectivo xeral pode dividirse nos seguintes obxectivos específicos:

- Deseñar a nova aplicación jGALiWeather na linguaxe Java.
- Producir un novo módulo de xeración de linguaxe natural en inglés utilizando a libraría Simple NLG.
- Integrar os diferentes módulos de xeración de linguaxe natural en jGALiWeather.
- Crear un servizo web, baseado en REST, para permitir o acceso ás predicións textuais xeradas.
- Crear unha interface gráfica para visualizar a información xerada por jGALiWeather.

1.2. Organización do documento

O obxectivo deste documento é presentar o traballo realizado para resolver correctamente os obxectivos definidos, explicando as tarefas realizadas e a estrutura e funcionamento das diferentes aplicacións implicadas no proxecto.

Esta memoria consta de 7 capítulos, un apéndice e a bibliografía correspondente:

- **Capítulo 1:** *Introdución.* Neste capítulo realizase unha pequena introdución ao conceptos xerais que se van a desenvolver ao longo da memoria. Conceptos coma ciencia de datos, xeración de linguaxe natural ou data-to-text son explicados neste capitulo. Ademais tamén faise mención aos obxectivos do proxecto e a estrutura deste documento.

- **Capítulo 2:** *Xestión do proxecto*. Neste capítulo descríbense as xestiones do alcance, da configuración, do tempo e de riscos seguidas durante este proxecto. Ademais indicase e explicase a metodoloxía seguida e faise unha hipotética estimación de custos.

- **Capítulo 3:** *Especificación de requisitos*. Neste capítulo faise un análise dos diferentes requisitos identificados, utilizando historias de usuario para representalos.

- **Capítulo 4:** *Arquitectura e ferramentas*. Neste capítulo mostrase a arquitectura do sistema a alto nivel. Por outra parte descríbense as diferentes ferramentas e tecnoloxías utilizadas ao longo do proxecto.

- **Capítulo 5:** *Deseño e Implementación*. Neste capítulo explicase o deseño a baixo nivel das diferentes aplicacións integradas no sistema, así como detallarase o seu funcionamento e como se implementaron.

- **Capítulo 6:** *Probas*. Neste capítulo se detallan as diferentes probas feitas para validar e verificar a aplicación.

- **Capítulo 7:** *Conclusión*. Neste capítulo establécense as diferentes conclusións derivadas da realización do proxecto e indícanse as posibles modificacións ou ampliacións a facer nun futuro.

- **Apéndice A:** *Manual de usuario*. Neste apéndice se describen tanto os pasos para instalar a aplicación como para manexar a mesma.

- **Bibliografía:** Referencias ao material utilizado para a realización desta memoria.

Capítulo 2

Xestión do proxecto

A xestión de proxectos software é unha parte esencial da enxeñaría de software levada a cabo ao longo de todo o proxecto, que busca lograr que o produto final se axuste aos obxectivos, necesidades e restricións impostas. Unha boa xestión non asegura o éxito dun proxecto pero unha mala xestión asegura o fracaso do mesmo.

Neste capítulo se explican de forma detallada a xestión do alcance, da configuración, dos tempos e dos riscos do proxecto, así coma a metodoloxía de traballo escollida neste caso. Adicionalmente inclúese unha análise dos posibles custos deste proxecto.

2.1. Xestión do alcance

A xestión do alcance inclúe tódolos procesos necesarios que permiten asegurar que o proxecto inclúe todo o traballo requirido, e só o traballo requirido, para completar o proxecto satisfactoriamente. Normalmente relaciónase principalmente coa definición e o control das tarefas a considerar ou omitir no proxecto [7].

2.1.1. Descrición do alcance do proxecto

jGALiWeather estará dividida en dous servizos: un servizo de xeración de predicións meteorolóxicas textuais a curto prazo e un servizo de xeración de predicións textuais para a calidade do aire a curto prazo. Para xerar esta predicións utilizará as seguintes variables meteorolóxicas de interese a curto prazo a catro días vista:

- **Estado do ceo:** Datos sobre cobertura nubosa e precipitacións para a mañá, a tarde e a noite. Estes datos preséntanse como códigos numéricos asociados a un determinado estado do ceo (21 distintos en total).

- **Vento:** Similar ao anterior, utilízanse códigos numéricos (34 en total) para representar a dirección e a intensidade do vento nos tres momentos do día.
- **Temperatura (máxima e mínima):** Proporcionase a temperatura máxima e mínima esperada para cada día en grados Celsius.
- **Calidade do aire:** Proporciona un código numérico para o nivel de calidade do aire de forma que pode ser bo, admisible, malo ou moi malo. Esta variable, a diferencia das demais, proporcionase só a tres días vista.

Os diferentes módulos da aplicación serán independentes entre eles e as predicións textuais terán que estar escritas en inglés.

Ademais deberase desenvolver unha interface web que mostre os datos anteriores xunto coas predicións en linguaxe textual para os 314 municipios galegos.

2.1.2. Obxectivos

O obxectivo xeral de jGALiWeather é o deseño, implementación e proba dunha nova aplicación na que se van aplicar técnicas de xeración de linguaxe natural máis sofisticadas cas orixinalmente utilizadas en GALiWeather, coa utilización de recursos de Xeración de Linguaxe natural (NLG) coma a libraría Java SimpleNLG [6]. Este obxectivo xeral pode dividirse nos seguintes obxectivos específicos:

- Deseñar unha nova aplicación (jGaliWeather) na linguaxe de programación Java a partir da aplicación GALiWeather.
- Producir un novo módulo de xeración de predicións meteorolóxicas textuais en inglés utilizando a libraría Simple NLG.
- Integrar os diferentes módulos de xeración de linguaxe natural en jGALiWeather.
- Crear unha interface gráfica para visualizar a información xerada por jGALiWeather.

2.1.3. Criterios de aceptación

O proxecto será aceptado se a aplicación jGALiWeather é capaz de realizar previsións meteorolóxicas en inglés de forma precisa e se proporciona unha interface web para a visualización dos diferentes datos meteorolóxicos.

2.1.4. Restricións do proxecto

- O proxecto debe de constar, como mínimo, de **412,5 horas** de traballo.
- O proxecto debe ser entregado antes de **08/07/16**.

2.1.5. Entregables do proxecto

Os produtos a entregar ao finalizar o proxecto son:

- Código fonte do software desenvolvido.
- Executables.
- Manual de instalación.
- Memoria do proxecto.
- Base de datos cos datos meteorolóxicos.

2.2. Metodoloxía de desenvolvemento

Nun proxecto de desenvolvemento de software é crucial a correcta elección da metodoloxía a utilizar. De non ser así, o proxecto pode sufrir retrasos, sobrecustos ou incluso pode fracasar na súa totalidade. Por esta razón é necesario analizar tódalas características, necesidades e limitacións do proxecto á hora de determinar a metodoloxía a utilizar.

Podemos diferenciar dous grandes grupos de metodoloxías: as metodoloxías tradicionais e as metodoloxías áxiles. O primeiro grupo caracterízase por levar a cabo unha documentación exhaustiva de todo o proxecto e por centrarse en cumprir un plan de proxecto definido ao inicio do mesmo. Isto implica uns altos custos á hora de realizar un cambio e unha falta de flexibilidade notable. As chamadas metodoloxías áxiles baséanse na adaptabilidade dos diferentes procesos do proxecto solucionando os problemas comentados anteriormente das metodoloxías tradicionais [9].

Debido a que o equipo de desenvolvemento deste proxecto está formado por unha soa persoa que carece da experiencia necesaria para realizar unha planificación inicial correcta, óptase pola utilización dunha metodoloxía áxil. Ademais disto, existen outras razóns para esta decisión, tales coma que o proceso está menos controlado ou se utilizan menos roles (ao tratarse dun equipo unipersoal non é necesario un control exhaustivo ou moitos roles).

2.2.1. Metodoloxía áxil - Scrum

Scrum é un proceso áxil que se pode usar para xestionar e controlar desenvolvementos complexos de software e produtos utilizando prácticas iterativas e incrementais [9].

En 1986, Ikujiro Nonaka e Hirotaka Takeuchi identificaron e definiron un enfoque integral que incrementaba a velocidade e flexibilidade do desenvolvemento de produtos comerciais. Compararon este novo enfoque de traballo en grupo ao avance en formación de *scrum* dos xogadores de rugby.

Aínda que existiron varias referencias por persoas de importancia dentro do sector a o uso do enfoque de Nonaka e Takeuchi no desenvolvemento de software ao longo dos anos, non foi ata 1995 cando Jeff Sutherland e Ken Schwaber presentaron un artículo de forma conxunta dando forma ao Scrum que coñecemos hoxe en dia.

Scrum non é unha metodoloxía, é un marco de traballo. Isto quere dicir que Scrum non define exactamente o que debe facerse senón que proporciona unha serie de boas prácticas a levar a cabo para asegurar o éxito de proxecto [10].

Scrum confía en que un equipo debe estar auto-xestionado e ser multi-disciplinar. Este equipo apoiase en dous roles específicos: o **Scrum Master** e o **Product Owner**. O primeiro pode entenderse coma un adestrador para o equipo, que axuda aos membros do equipo a usar Scrum para obter o máximo rendemento. O Product Owner representa aos clientes ou os usuarios e guía ao equipo a construír o produto correcto.

Durante cada iteración ou *sprint*, o equipo crea un novo incremento do software. Ao principio de cada sprint celébrase unha reunión entre o equipo e o Product Owner na que se deciden as características que se van implementar nese sprint. Esas características proveñen da pila do produto que contén tódolos requisitos priorizados que deben de ser implementados. A pila do produto pode modificarse nas reunións de inicio de cada sprint pero durante o mesmo debe manterse inalterada. Os items escóllense mediante consenso entre o Product Owner, que determina o alcance e a prioridade dos mesmos, e o equipo, que estima a cantidade de traballo á que pode comprometerse. O resultado de cada sprint debe ser unha nova versión operativa do software.

Este proceso xunto cunha duración de sprint relativamente curta (normalmente de 2 a 4 semanas) fan que a probabilidade de éxito sexa moi alta. Isto é debido a

que existen versións operativas do software durante todo o proxecto, facendo posible que o Product Owner logre entender mellor as súas necesidades, permitindo facer os cambios pertinentes. Scrum recoñece que é moi posible que os clientes poidan cambiar os seus pensamentos sobre o que queren e o que necesitan durante o proxecto. Por esa razón, centrase en maximizar as habilidades do equipo para facer entregas rápidas e responder aos cambios emerxentes.

2.2.2. Aplicación de Scrum no proxecto

Como Scrum está moi orientado a maximizar o rendemento do equipo de traballo, é moi difícil aplicar moitas das súas practicas debido a que neste proxecto dispoñemos dun equipo unipersoal. Sen embargo, si se utilizaran técnicas coma a organización do traballo en sprints, as reunións ou o uso de gráficos Burn-Down.

Neste proxecto tivemos sprints de 3 semanas de duración e ao principio de cada un fixose unha reunión cos Product Owner (titores) na que se mostrou o traballo feito no sprint anterior e se planificou o seguinte. Durante o sprint o traballo controlouse con gráficos Burn-Down para determinar se existe algún perigo para o éxito do sprint.

2.3. Xestión da configuración

A xestión da configuración é un proceso cuxo propósito é establecer e manter a integridade dos produtos de traballo identificando os elementos ou produtos que van ser controlados, definindo un procedemento para o control deses produtos e o rexistro do estado dos produtos [8].

Neste proxecto os elementos de traballo son os ficheiros de código fonte e tódolos arquivos que forman a documentación do proxecto. Á hora de crear un plan de xestión de configuración parece máis acertado tratar estes dous tipos de elementos de traballo de forma independente, xa que as necesidades de control e mantemento da integridade son diferentes.

2.3.1. Xestión do código fonte

O código fonte almacénase nun repositorio Git aloxado en *Github*. Git é unha ferramenta moi potente que permite o control eficiente das versións dunha aplicación cando esta ten un gran número de arquivos de código fonte. Para aumentar a seguridade aloxouse o repositorio na plataforma *Github*, o que eliminou a dependencia da integridade dun único ordenador de traballo.

Ao remate de cada nova funcionalidade faise un *Commit* local para gardar os cambios realizados. Cada *Commit* leva asociado un comentario no que se debe explicar os cambios realizados nesa instantánea. Cando remata a xornada de traballo executase un Push para subir todos os cambios ao repositorio remoto e así estar cubertos fronte a posibles fallos ou perda de información do ordenador de traballo.

2.3.2. Xestión da documentación

Para manter a documentación a salvo de posibles fallos no ordenador de traballo utilizamos a ferramenta *Google Drive*, que permite ter todos os nosos documentos aloxados na nube. Ademais deste incremento da seguridade, o manter os documentos na nube permite aos titores do traballo ter en todo momento acceso a documentación para consultala cando lles sexa necesario. Para manter os documentos ordenados e poder acceder a eles o máis rápido posible emprégase a estrutura de carpetas que se mostra na figura 2.1.

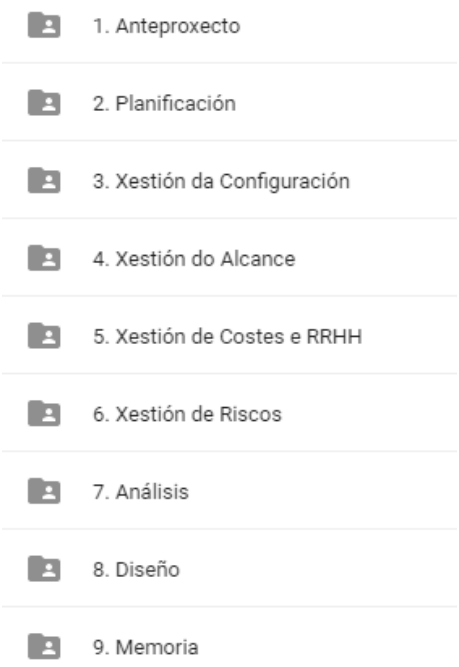


Figura 2.1: Estructura de carpetas.

Os documentos manterán a nomenclatura:

<Nome>_<Versión>.<Extensión>

A versión de cada documento increméntase cando hai suficientes cambios importantes, sendo criterio do dono do proxecto decidir isto. Para manter un control de cambios sobre os documentos utilizamos as ferramentas que proporcionan a suite *Microsoft Office* e o editor LaTeX *Texmaker*. Para poder aumentar a versión dun documento é condición necesaria que tódolos cambios realizados na versión anterior sexan aceptados polo dono do proxecto.

2.4. Xestión do tempo

A xestión do tempo inclúe os procesos necesarios para lograr a conclusión do proxecto a tempo [7]. Eses procesos son os seguintes:

- Definición das actividades: identifica as actividades específicas que deben ser realizadas para producir os diferentes produtos entregables do proxecto.
- Establecemento da secuencia das actividades: identifica e documenta as dependencias entre as actividades.
- Estimación dos recursos: estima o tipo e cantidade de recursos para realizar cada actividade.
- Desenvolvemento do cronograma: analiza as secuencias das actividades, a duración das actividades, os requisitos de recursos e as restricións para crear o cronograma do proxecto.
- Control do cronograma: controla os cambios no cronograma do proxecto.

Para representar o traballo executado durante o proxecto utilizamos a EDT da figura 2.2, especificado previamente no anteproxecto. Unha EDT (estrutura de descomposición do traballo) é unha descomposición xerárquica, orientada ao produto entregable, do traballo que será executado polo equipo do proxecto en porcións de traballo máis pequenas e fáciles de manexar, onde cada nivel descendente da EDT representa unha definición cada vez máis detallada do traballo do proxecto.

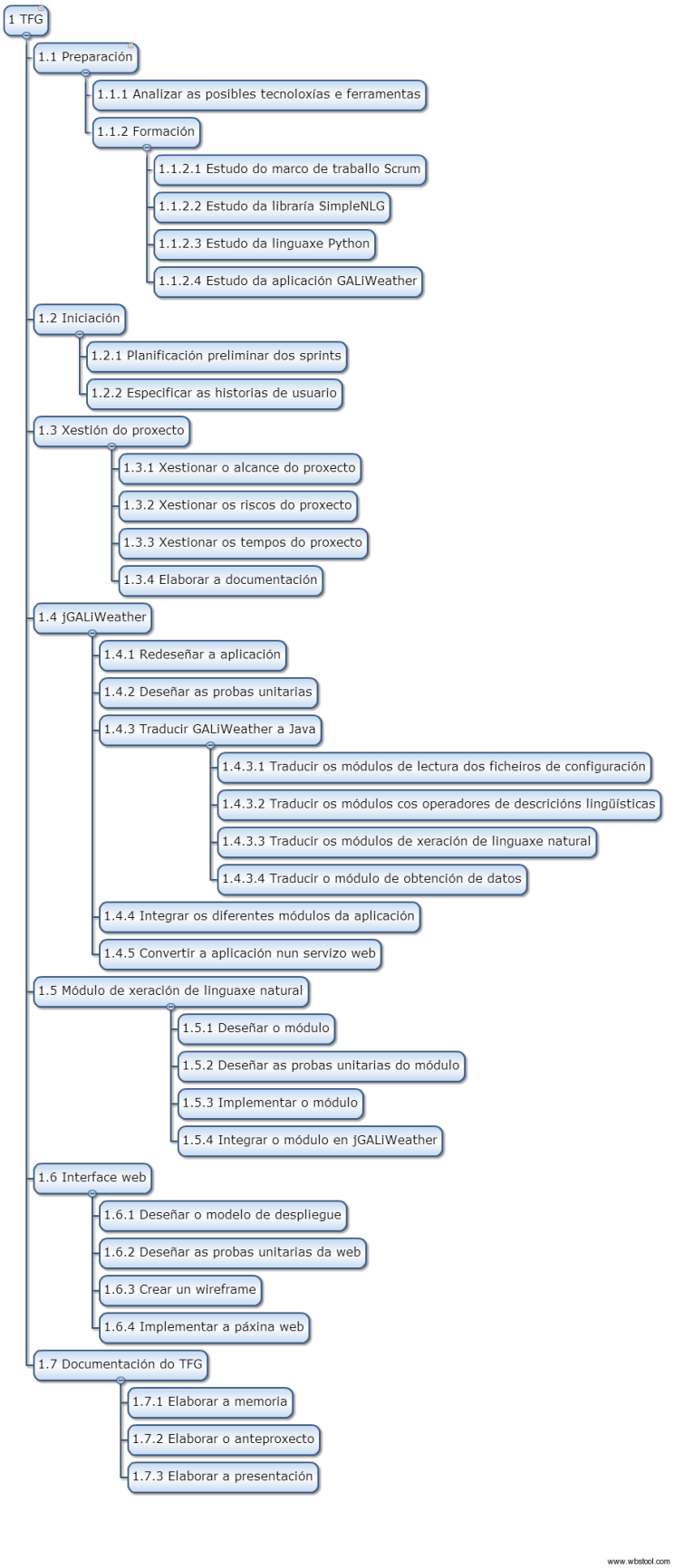


Figura 2.2: Estrutura de descomposición do traballo (EDT).

A continuación descríbense as diferentes etapas identificadas dunha forma máis detallada. Indicarase os recursos que farán falta para a súa realización ademais da súa duración estimada ao principio do proxecto. As estimacións son moi inexactas por tratarse de estimación iniciais e non das estimación feitas no inicio de cada sprint. A dedicación será de 21 horas/semana que polas 20 semanas de duración do proxecto da unha aproximación de 412,5 horas de traballo totais.

Existen dous roles diferenciados ao longo do proxecto: o Product Owner interpretado polos titores do proxecto e o desarrollador interpretado polo estudante.

Etapa de preparación

Esta fase abarca tódalas actividades de estudo previas ao inicio do proxecto. Estudiáronse as tecnoloxías utilizadas ao longo do proxecto, coma a linguaxe Python ou a libraría SimpleNLG, o marco de traballo Scrum e, o máis importante, a aplicación GALiWeather. Foi necesario estudar a estrutura e as funcionalidades da aplicación GALiWeather dado que gran parte do proxecto baseouse nela. Todo este estudio e formación realizouse mediante a documentación oficial e as diferentes lecturas e tutoriais que se poden encontrar na web. Todas estas actividades son realizadas exclusivamente polo desarrollador.

Tempo estimado
1 semana

Etapa de iniciación

Nesta etapa os Product Owner e o desarrollador traballan conxuntamente para determinar a súas necesidades con respecto á aplicación e crear as diferentes historias de usuario. Por outra parte, se fai unha estimación inicial da duración dos sprints para realizar unha planificación inicial. Esta planificación axuda dunha maneira gráfica a determinar a organización das tarefas e determinar que é posible realizar durante o proxecto tendo en conta a súa duración. Como produtos xerados por esta fase temos a pila de produto coas historias de usuario e un diagrama de Gantt inicial.

Tempo estimado
1 semana

Etapa de xestión do proxecto

O obxectivo desta fase é xestionar o alcance, os riscos e os tempos do proxecto para asegurar o éxito do mesmo. Esta etapa prolongase ao longo de todo o proxecto xa que todas estas tarefas implican un certo control sobre o mesmo. A estimación de dedicación diaria a esta tarefa é dun 10 %. A realización das actividades desta fase é responsabilidade do desarrollador e dos Product Owner e obtemos o plan de xestión do proxecto como produto xerado.

Tempo estimado
2 semanas

Etapa de desenvolvemento de jGALiWeather

Esta etapa abarca todas as tarefas necesarias para o desenvolvemento da aplicación xeradora de prognósticos en linguaxe natural **jGALiWeather**. Isto implica tanto o deseño e a codificación dos diferentes módulos da aplicación coma a súa integración e probas. Esta fase foi realizada en exclusiva polo desenvolvedor e xerou a documentación e o código fonte de **jGALiWeather**.

Tempo estimado
6 semanas

Etapa de desenvolvemento do módulo de xeración de linguaxe natural

Esta etapa abarca todas as tarefas necesarias para o desenvolvemento dun novo módulo de xeración de linguaxe natural. Na fase anterior realizouse un módulo de xeración mediante modelos similar ao de GALiWeather. O obxectivo desta fase é substituír ese módulo por un novo utilizando técnicas NLG máis sofisticadas grazas a libraría SimpleNLG. Esta fase foi realizada polo desarrollador e xerou a documentación e o código fonte da aplicación co novo módulo.

Tempo estimado
4 semanas

Etapa de desenvolvemento da interface web

Esta etapa abarca todas as tarefas necesarias para o desenvolvemento da interface de visualización do datos meteorolóxicos. Esta fase foi realizada en exclusiva polo desarrollador e xerou a documentación da aplicación web.

Tempo estimado
3 semanas

Etapa de documentación do TFG

Fase na que se redactan todos os documentos necesarios relacionados co procedemento do traballo de fin de grao. Esta etapa xera tres documentos: o anteproxecto do TFG, a memoria do TFG e a presentación final. Esta fase foi realizada polo desarrollador baixo a supervision dos titores.

Tempo estimado
3 semanas

A continuación, na figura 2.3, podemos ver o cronograma da planificación dos distintos fases e sprints do proxecto.

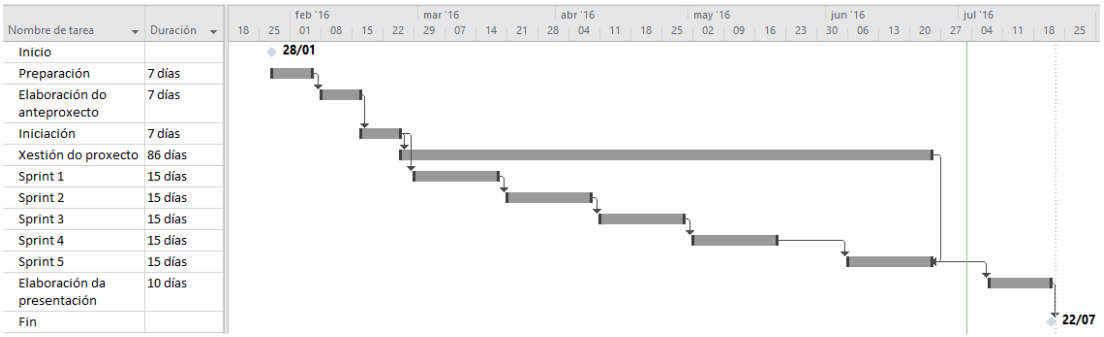


Figura 2.3: Cronograma da planificación (Gantt).

O diagrama anterior non é moi representativo do traballo realizado polo que utilizamos gráficos *Burn-Down* para mostralo. Un gráfico *Burn-Down* é un diagrama de dous eixes que sirve para determinar canto tempo falta para terminar as historias comprometidas nun sprint. No eixe X representase o tempo en horas ideais de duración do sprint, mentres que no eixe Y temos a cantidade de traballo, en horas ideais (o concepto de hora ideal explícase no capitulo 3), comprometida co Product Owner durante o sprint. Nela se trazan dúas líneas: unha

que representa o avance do traballo ideal e outra que mostra o avance do traballo real.

Todos os sprints estaba planificados para ter unha duración de 3 semanas (5 días de traballo por semana) pero debido a diversas razóns modificouse a duración dalgúns deles.

Na figura 2.4 móstrase o primeiro sprint no que se realizaron todas as tarefas de tradución da aplicación GALiWeather a Java. Tívose que aumentarse o prazo do sprint a 19 días (+- 4 semanas) para poder ter un versión executable da aplicación.

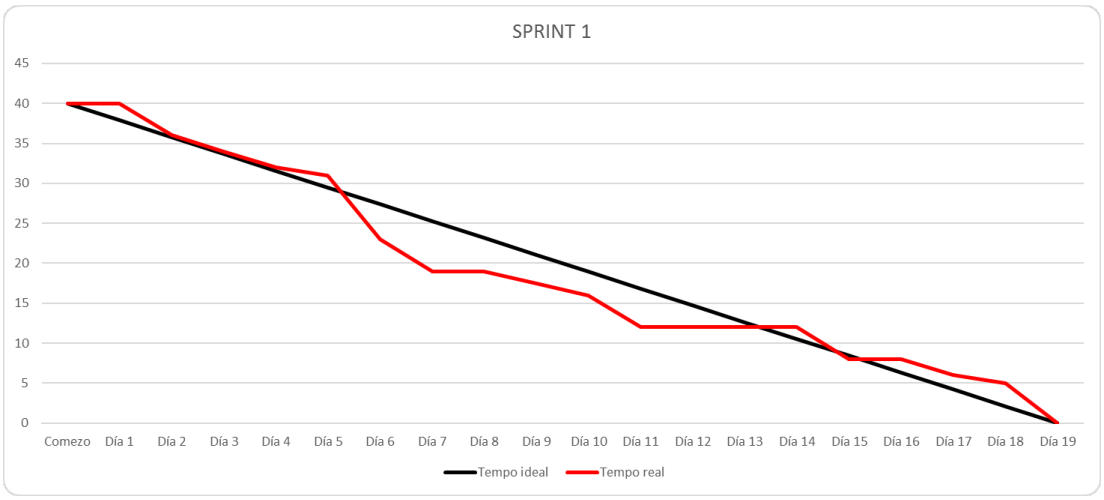


Figura 2.4: Gráfico Burn-Down do Sprint 1.

O segundo sprint se mostra na figura 2.5 e centrouse na creación do novo módulo de xeración en linguaxe natural. Neste gráfico podemos ver como durante este sprint o traballo foi moi irregular chegando a traballar máis de 8 horas diarias para despois ter un parón de algo máis dunha semana. Este parón debeuse á necesidade de tempo para afrontar o traballo doutras materias.

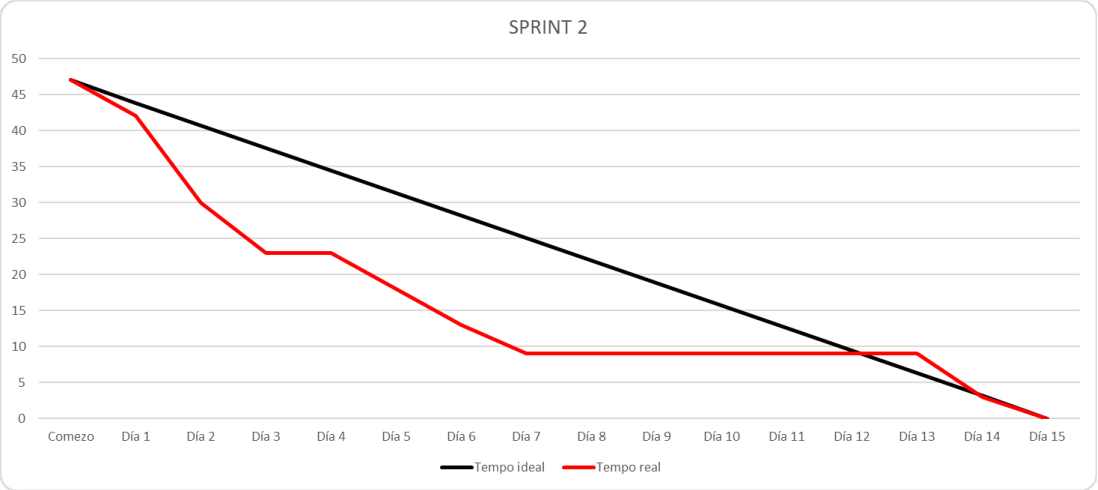


Figura 2.5: Gráfico Burn-Down do Sprint 2.

O terceiro sprint (figura 2.6) foi o máis heteroxéneo de todos. Nel creouse toda a interface web, completouse a aplicación xeradora de textos, xa que anteriormente non se tiñan en conta os datos históricos da temperatura, e fixéronse as 25 probas proporcionadas polo meteorólogo profesional para validar a parte encargada do ICA.

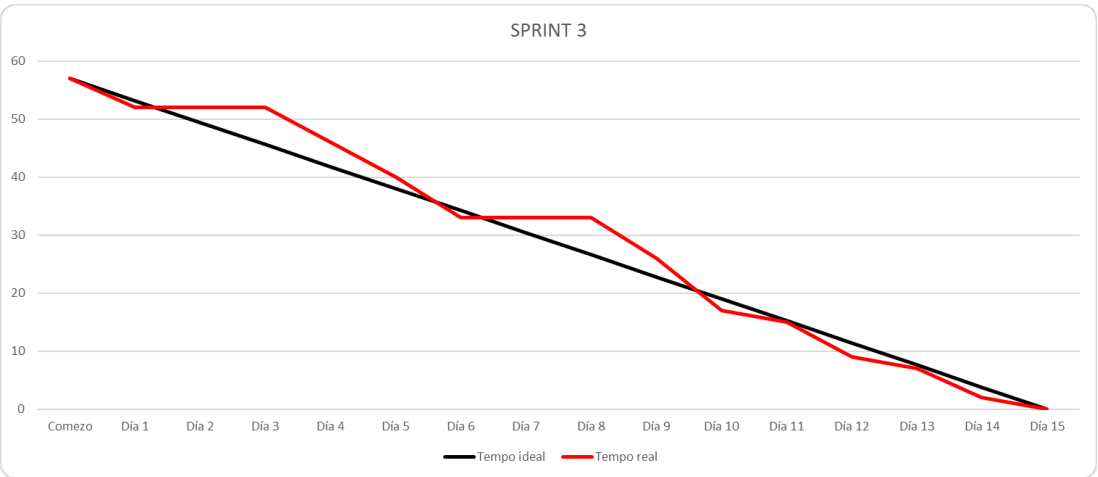


Figura 2.6: Gráfico Burn-Down do Sprint 3.

O cuarto sprint, mostrado na figura 2.7, foi un sprint moi corto de tan solo 10 días debido a un parón no proxecto debido aos exames de maio e unha incompatibilidade horaria cos titores (risco **R05**). Para cadrar a planificación do proxecto decidiuse facer un sprint moi corto con moitas horas de traballo diarias para recuperar o tempo perdido. Neste sprint implementaronse as 50 probas necesarias

para a validación do historia de usuario **HU-01**, deseñouse a interface *responsive* e se engadiu a xeolocalización á aplicación web.

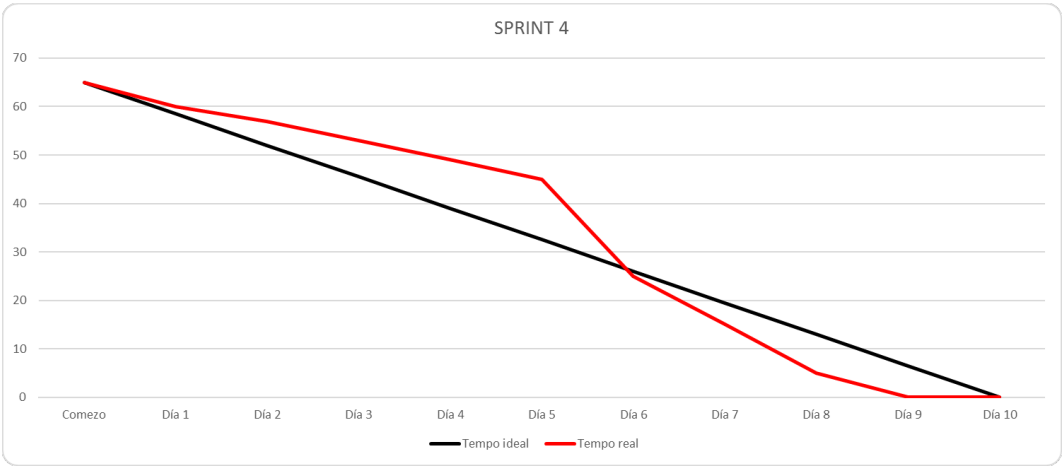


Figura 2.7: Gráfico Burn-Down do Sprint 4.

No ultimo sprint (figura 2.8) volveuse á normalidade de 3 semanas de duración e se centrou na realización da memoria do TFG e en diversas correccións e melloras no software. Cabe destacar que debido á ocorrencia do risco **R04** tivo que encontrarse unha maneira alternativa de acceso a datos meteorolóxicos. Ao final, decidiuse facer un *parser* da páxina web de Meteogalicia que cada día recuperara os datos meteorolóxicos de todos os municipios galegos para almacenalos nunha base de datos en local.

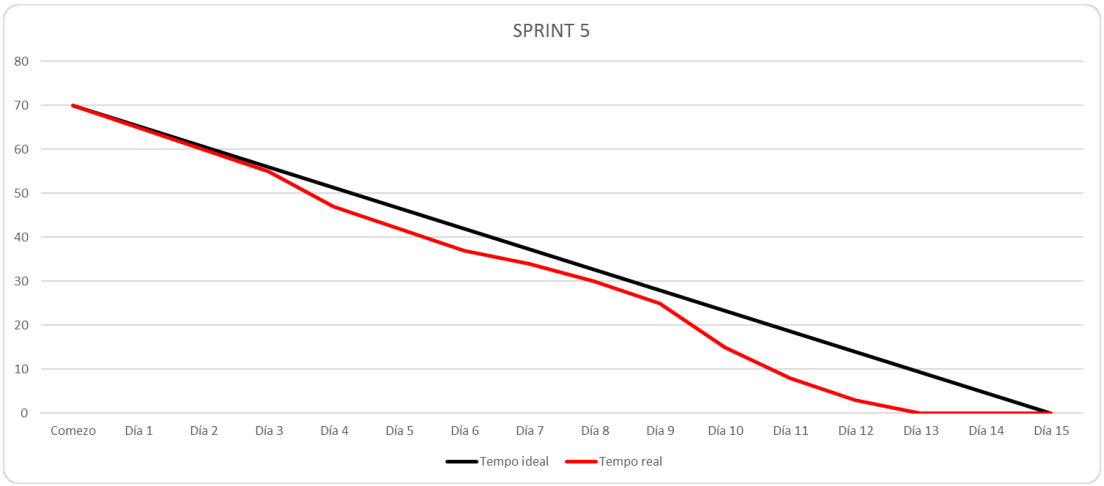


Figura 2.8: Gráfico Burn-Down do Sprint 5.

2.5. Xestión de riscos

O propósito da xestión de riscos é identificar os problemas e oportunidades potenciais antes de que ocorran para poder planificar actividades de tratamento necesarias para mitigar ou eliminar as posibles adversidades á hora de alcanzar os obxectivos do proxecto ou tomar vantaxe das posibles oportunidades [11].

A xestión de riscos é un proceso continuo que debe considerar tanto fontes internas como externas de riscos. É moi importante a detección temprana e agresiva do risco xa que, normalmente, é máis fácil, menos custoso e menos prexudicial facer os cambios pertinentes durante as fases máis tempranas do proxecto.

En Scrum a identificación e planificación dos riscos fanse ao principio do proxecto deixando o control dos riscos en mans das numerosas reunións que se fan nun marco de traballo como este. Desta maneira faise unha xestión de riscos continua máis eficiente que a xestión de riscos das metodoloxías tradicionais onde a burocracia dificulta que se leve a cabo correctamente.

Existen catro procesos a realizar na xestión de riscos:

- Identificación dos riscos.
- Análise dos riscos.
- Planificación da resposta.
- Seguimento.

Primeiro identificaremos os riscos para posteriormente poder avaliar a súa probabilidade de aparición e índice de impacto sobre o proxecto. Dado que este proxecto realízase a custe cero o impacto medirase en días de traballo. En función destes dous valores determinarase o índice de exposición ao risco que dará unha medida moi adecuada para determinar a importancia dun risco fronte aos demais.

Por ultimo se planificaran as estratexias de minimización, prevención ou continxencia para cada un dos riscos identificados.

Riscos do proxecto

R01 - Análise de requisitos errónea	
Descrición	Os requisitos identificados non se axustan as necesidades do cliente.
Probabilidade	10 %
Impacto (días)	10 días
Exposición (días)	1 día
Estratexia de minimización	O desenvolvemento baseado en sprints fai que os erros na captura de requisitos se detecten desde fases moi tempranas do proxecto.

R02 - Atraso con respecto a planificación	
Descrición	Atraso con respecto a planificación inicial debido a falta de experiencia coas tecnoloxías.
Probabilidade	40 %
Impacto (días)	20 días
Exposición (días)	8 días
Estratexia de minimización	A planificación das tarefas en días ideais ten en conta os posibles atrasos que podan aparecer durante os sprints.
Estratexia de continxencia	Reaxustar a planificación inicial deixando de lado os requisitos con menor importancia.

R03 - Alcance do proxecto impreciso	
Descrición	O alcance do proxecto non cumpre as necesidades do cliente.
Probabilidade	10 %
Impacto (días)	20 días
Exposición (días)	2 días
Estratexia de prevención	Consultar cos titores o documento do alcance tan pronto como este estea feito.

Riscos de dispoñibilidade

R04 - Acceso denegado á BBDD de Meteogalicia	
Descrición	Meteogalicia non da o seu consentimento para acceder aos seus datos meteorolóxicos.
Probabilidade	20 %
Impacto (días)	10 días
Exposición (días)	2 días
Estratexia de continxencia	Crear unha BBDD propia a partir dos datos recollidos da páxina web de Meteogalicia.

R05 - Indispoñibilidade dos titores	
Descrición	Os titores non están dispoñibles para facer as reunións de planificación dos diversos sprints.
Probabilidade	5 %
Impacto (días)	5 días
Exposición (días)	0.25 días
Estratexia de continxencia	Busca dunha canle de comunicación alternativo para facer a reunión ou atraso da reunión para outra data.

Riscos do produto

R06 - Fallo no ordenador persoal	
Descrición	Un fallo ou aviría no ordenador de traballo produce a perda da información do proxecto.
Probabilidade	5 %
Impacto (días)	5 días
Exposición (días)	0.25 días
Estratexia de prevención	A utilización de solucións na nube para o almacenamento de arquivos reduce a probabilidade de que se produza algunha perda de información.

R07 - Falta de soporte por parte das tecnoloxías utilizadas	
Descrición	Algunha das tecnoloxías utilizadas presentan errores que impiden o desenvolvemento do produto.
Probabilidade	10 %
Impacto (días)	20 días
Exposición (días)	2 días
Estratexia de continxencia	Busca de novas alternativas para substituír as tecnoloxías actuais do proxecto.

R08 - Mala usabilidade da plataforma	
Descrición	A web non é suficientemente clara e sinxela de usar para os usuarios.
Probabilidade	10 %
Impacto (días)	5 días
Exposición (días)	0.5 días
Estratexia de continxencia	Ter unha reunión cos titores para identificar os problemas de usabilidade e as posibles soluciones a eses problemas.

2.6. Estimación de custos

Debido á natureza do Traballo de Fin de Grado, os custos manexados neste apartado son puramente teóricos. Os custos directos divídense en custos de persoal, de material e de software.

Para todos os produtos software e materiais establecese que a vida útil sera de 3 anos de amortización dos cales só se utilizaron durante 3 meses a tempo completo neste proxecto.

Custos de persoal

O equipo de desenvolvemento esta formado por unha persoa co rol de Analista-Programador xa que realiza tanto o análise como o desenvolvemento do software. Segundo un estudio salarial feito por *Vitae Consultores* estima o salario medio dun Analista-Programador Java na provincia da Coruña en 23.000€ brutos anuais [24]. Estimase que un terzo do salario bruto dun traballador son os custos que ten

que asumir a empresa pola súa contratación e que ó ano hai 1.800 horas laborais nunha xornada estándar de 8 horas diarias. Tendo en conta estes datos podemos concluír co custo total dun Analista-Programador é de 17,00€/hora.

Custos materiais

Para este proxecto soamente necesitase un PC de sobremesa para o único traballador do proxecto que podemos estimalo nuns 75,00€ de amortización.

Custos de software

Durante este proxectos utilizáronse as seguintes ferramentas software:

- Windows 10 Pro (23,25€)
- Netbeans IDE (0,00€)
- Microsoft Office (23,25€)
- TexMaker (0,00€)
- PyCharm IDE (0,00€)
- Librarías e SDKs (0,00€)
- Microsoft Project (64,08€)
- Apache Tomcat (0,00€)

Custos totais

A todos este custos hai que engadirlle gastos indirectos asociados a facturas de servizo básicos, alugamentos, etc. que na USC para proxectos TIC establecese nun 21 % adicional do custos directos do proxecto.

Concepto	**Cantidade**	**Custo Unitario**	**Custo Total**
Analista-Programador	412.5 horas	17,00€/hora	7012,50€
Ordenador Persoal	-	-	75,00€
Software	-	-	110,58€
Custos Indirectos	-	-	1511,60€
		Total	**8709,67€**

Cadro 2.1: Custos totais do proxecto.

Capítulo 3

Especificación de requisitos

O análise de requisitos é o proceso de estudo das necesidades dos usuarios para chegar a unha definición dos requisitos do sistema, do hardware ou do software, así coma de estudo e refinamento deses requisitos [12]. Esta tarefa é esencial para o éxito de calquera proxecto xa que dela depende que o produto creado se axuste as necesidades do cliente (O produto pode ser excepcional a nivel técnico pero non serve de nada se non lle aporta valor ao cliente).

Neste proxecto nos afastamos das técnicas de especificación de requisitos máis clásicas e pesadas, nas que os requisitos son descritos de forma moi estrita e formal, para utilizar historias de usuario, empregadas por diversas metodoloxías áxiles. Unha historia de usuario é un requisito escrito de forma breve e nunha linguaxe entendible polo cliente ou usuario. As historias de usuario deben cumprir 6 características (INVEST) [13]:

- **Independente:** pódense desenvolver en calquera orde.
- **Negociable:** non é un contrato explícito, só unha intención que pode ser modificada co tempo.
- **Valorable:** sempre lle debe dar valor ao cliente.
- **Estimable:** deben de estar suficientemente claras como para poder estimalas.
- Pequena (**Small**): as historias pequenas son máis fáciles de estimar.
- Verificable (**Testable**): é a única forma de saber se unha historia está acabada.

Mark Coehn recomenda utilizar o seguinte modelo para escribir historias de usuario:

Como <tipo de usuario>, quero <obxectivo>, xa que <razón>

Neste proxecto utilizamos solo as dúas primeiras partes do modelo omitindo a razón da historia. Esta ultima parte ten coma fin que os integrantes de proxectos de longa duración poidan recordar a importancia dunha historia cando haxan pasado varios meses de desenvolvemento. Como este é un proxecto corto e con poucos integrantes non é necesario gardar esta información. Se unha historia de usuario describe unha restrición engadiremos a palabra 'Restrición' ao final da descrición.

Ademais, a cada historia de usuario se lle engadirá un identificador, un titulo, unha estimación, uns criterios de aceptación e un apartado de notas no que se engadirá calquera outra información en caso de que sexa necesario.

As estimacións faranse en días ideais. Un día ideal é equivalente a un día no que non se sufre ningunha distracción durante a xornada de traballo. Un desenvolvedor de software nun día normal de traballo, ademais de traballar nas tarefas planificadas, pasa moito tempo respondendo correos, asistindo a reunións, corrixindo bugs, etc.. Os días ideais serven para estimar o tamaño dunha historia para poder comparalas entre elas, á vez que da unha primeira idea do tempo que levará desenvolvela.

As historias de usuario dispoñen dun nivel de importancia que indica o impacto que terá cada unha no valor final do produto e no éxito do proxecto. Definimos tres niveis de importancia:

- **Obrigatorio:** Indica que esta historia é crucial para o éxito do proxecto. Se non esta presente na versión final do produto o proxecto considerarase un fracaso.
- **Desexable:** O proxecto considerase un éxito aínda que a historia non esté implementada pero a súa presenza aumentara de forma significativa o valor final do produto.
- **Opcional:** A historia terá un impacto mínimo no valor final do produto.

3.1. Historias de Usuario

3.1.1. Historias de Usuario: Cliente

HU-01 - Xerar predicións meteorolóxicas	
Solicitante	Usuario
Descrición	quero poder xerar un prognóstico meteorolóxico en linguaxe natural (inglés) a partires dos datos de predición a 4 días vista.
Estimación	7 días
Importancia	Obrigatorio
C. Aceptación	O requisito considerarase cumprido cando 50 prognósticos aleatorias repartidas nos deferentes períodos do ano son similares a os xerados nos mesmos períodos por GALiWeather.
Notas	

HU-02 - Xerar predicións da calidade do aire	
Solicitante	Usuario
Descrición	quero poder xerar un prognóstico do estado da calidade do aire en linguaxe natural (inglés) a partires dos datos de predición meteorolóxicos a 3 días vista.
Estimación	3 días
Importancia	Obrigatorio
C. Aceptación	O requisito considerarase cumprido cando a aplicación sexa capaz de xerar os mesmos prognósticos que os realizados por un experto en calidade do aire tendo como entrada os mesmos datos meteorolóxicos.
Notas	

HU-03 - Xerar predicións en portugués	
Solicitante	Usuario
Descrición	quero que os prognósticos sexan escritos en portugués.
Estimación	5 días
Importancia	Opcional
C. Aceptación	A aplicación é capaz de xerar prognósticos meteorolóxicos entendibles por unha persoa da lingua portuguesa.
Notas	

HU-04 - Xerar predicións en francés	
Solicitante	Usuario
Descrición	quero que os prognósticos sexan escritos en francés.
Estimación	5 días
Importancia	Opcional
C. Aceptación	A aplicación é capaz de xerar prognósticos meteorolóxicos entendibles por unha persoa da lingua francesa.
Notas	

HU-05 - Engadir máis parámetros	
Solicitante	Usuario
Descrición	quero que nas predicións se faga referencia á presión atmosférica e á humidade.
Estimación	6 días
Importancia	Opcional
C. Aceptación	A aplicación é capaz de xerar prognósticos en linguaxe natural facendo referencia á humidade e á presión atmosférica.
Notas	

HU-06 - Facer predicións sen todos os parámetros	
Solicitante	Usuario
Descrición	quero obter prognósticos aínda que non estean dispoñibles todos os datos meteorolóxicos.
Estimación	4 días
Importancia	Opcional
C. Aceptación	A aplicación é capaz de xerar prognósticos en linguaxe natural coherentes aínda que so se proporcione un dato meteorolóxico.
Notas	

HU-07 - Mellorar as predicións	
Solicitante	Usuario
Descrición	quero que as predicións en inglés se correspondan mellor coa forma de falar dun inglés nativo.
Estimación	5 días
Importancia	Desexable
C. Aceptación	Un experto en inglés valida todas as posibles predicións xeradas pola aplicación.
Notas	

HU-08 - Visualizar datos meteorolóxicos	
Solicitante	Usuario
Descrición	quero ter unha interface de visualización que mostre os datos meteorolóxicos e as predicións para todos os municipios galegos.
Estimación	8 días
Importancia	Obrigatorio
C. Aceptación	A aplicación web mostra a temperatura máxima e mínima, o estado do ceo, a dirección e intensidade do vento, a calidade do aire e o prognostico en linguaxe natural para os 4 próximos días para un municipio galego concreto. A aplicación web permite visualizar os datos anteriores para cada municipio galego existente.
Notas	A interface de visualización debe ser unha páxina web.

HU-09 - Interface web responsive	
Solicitante	Usuario
Descrición	quero que a interface de visualización se adapte a diferentes tamaños de pantalla. Restrición
Estimación	7 días
Importancia	Desexable
C. Aceptación	A web cambia o seu formato dependendo do tamaño de pantalla do dispositivo que a visualiza.
Notas	

HU-10 - Obtención do municipio por xeolocalización	
Solicitante	Usuario
Descrición	quero que se mostren automaticamente os datos meteorolóxicos do municipio no que me atopo ao visitar o sitio web.
Estimación	2 días
Importancia	Opcional
C. Aceptación	A web mostra os datos meteorolóxicos do municipio no que se encontra físicamente o dispositivo.
Notas	

3.1.2. Historias de Usuario: Desenvolvedor de software

HU-11 - Crear módulo de lectura dos ficheiros de configuración	
Solicitante	Desenvolvedor de software
Descrición	quero obter a información de configuración aloxada en ficheiros externos.
Estimación	4 días
Importancia	Obrigatorio
C. Aceptación	A aplicación é capaz de xerar estruturas cos datos obtidos nos ficheiros de configuración externos.
Notas	

HU-12 - Crear módulo cos operadores de descricións lingüísticas	
Solicitante	Desenvolvedor de software
Descrición	quero converter os datos meteorolóxicos en descricións lingüísticas intermedias.
Estimación	5 días
Importancia	Obrigatorio
C. Aceptación	A aplicación é capaz de xerar descricións lingüísticas consistentes cos datos meteorolóxicos proporcionados.
Notas	

HU-13 - Crear módulo de xeración de linguaxe natural	
Solicitante	Desenvolvedor de software
Descrición	quero converter descricións lingüísticas en prognósticos en linguaxe natural.
Estimación	5 días
Importancia	Obrigatorio
C. Aceptación	A aplicación é capaz de xerar prognósticos en linguaxe natural consistentes coas descricións lingüísticas proporcionados.
Notas	

HU-14 - Crear módulo de obtención de datos	
Solicitante	Desenvolvedor de software
Descrición	quero obter os datos meteorolóxicos desde unha fonte externa.
Estimación	5 días
Importancia	Obrigatorio
C. Aceptación	A aplicación é capaz de obter temperatura máxima e mínima, estado do ceo xunto coa intensidade e dirección do vento para cada momento do día(maña, tarde e noite) e o índice de calidade do aire para cada municipio galego.
Notas	

HU-15 - Xeración de textos en módulos independentes	
Solicitante	Desenvolvedor de software
Descrición	quero que a xeración de textos en cada idioma sexa estruturada en módulos independentes ao resto da aplicación. Restrición.
Estimación	1 día
Importancia	Obrigatorio
C. Aceptación	Os módulos de xeración de texto para cada idioma non dependen dos demais módulos da aplicación para o seu funcionamento.
Notas	

HU-16 - Converter en servizo RestFul	
Solicitante	Desenvolvedor de software
Descrición	quero que o modulo de xeración de texto sexa un servicio Rest.
Estimación	2 días
Importancia	Obrigatorio
C. Aceptación	As funcións da aplicación poden ser chamadas a través de peticións HTTP.
Notas	

HU-17 - Converter en servizo SOAP	
Solicitante	Desenvolvedor de software
Descrición	quero que o modulo de xeración de texto sexa un servicio SOAP.
Estimación	1 día
Importancia	Opcional
C. Aceptación	Existe un ficheiro WSDL co que se poden acceder ás funcións da aplicación .
Notas	

HU-18 - Información BBDD en ficheiro externo	
Solicitante	Desenvolvedor de software
Descrición	quero que a información de acceso a base de datos poida modificarse dende un ficheiro externo. Restrición.
Estimación	0.5 días
Importancia	Obrigatorio
C. Aceptación	A información de acceso á BBDD é lida desde un ficheiro externo con éxito.
Notas	

HU-19 - Información de localización en ficheiro externo	
Solicitante	Desenvolvedor de software
Descrición	quero que a localización dos ficheiros poda indicarse nun ficheiro externo. Restrición.
Estimación	0.5 días
Importancia	Obrigatorio
C. Aceptación	A aplicación é capaz de ler os ficheiros necesarios para a execución desde a localización indicada no ficheiro externo.
Notas	

HU-20 - Saída de datos en JSON	
Solicitante	Desenvolvedor de software
Descrición	quero que os datos devoltos polos servizos REST estean en formato JSON. Restrición.
Estimación	1 día
Importancia	Desexable
C. Aceptación	Os datos de saída do servizos REST están en formato JSON.
Notas	

Capítulo 4

Arquitectura e Ferramentas

A arquitectura dun sistema software é definida polo IEEE como o conxuntos dos conceptos fundamentais e as propiedades dun sistema no seu entorno, plasmados nos seus elementos, relacións e nos principios do seu deseño e evolución. De forma máis simple podemos dicir que a arquitectura dun software define, de maneira abstracta, os compoñentes que levan a cabo as diferentes tarefas de computación, as súas interfaces e a forma en que se comunican entre eles. Esta arquitectura sempre debe deseñarse en base a uns requisitos e restricións [14, 15].

Este capitulo centrarase no deseño a alto nivel da arquitectura da aplicación e na descrición das diferentes tecnoloxías e ferramentas usadas para cumprir os obxectivos do proxecto.

4.1. Arquitectura do sistema

Unha vez analizados os requisitos da aplicación identifícanse tres partes ben diferenciadas dentro da aplicación: a interface web, a aplicación que xera os prognósticos meteorolóxicos en linguaxe natural e o provedor de datos.

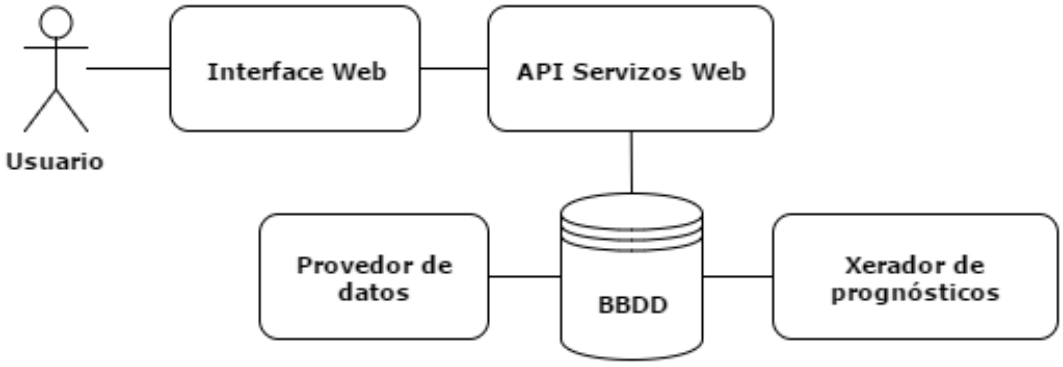


Figura 4.1: Arquitectura do sistema.

Na figura 4.1 podemos ver como o usuario interactua coa interface web onde pode visualizar os prognósticos meteorolóxicos para os vindeiros días do municipio galego que eles seleccionen. A interface comunicase cunha API de servizos web que encargase de obter os datos meteorolóxicos pertinentes da base de datos. Por ultimo, a parte mais importante da aplicación é o xerador de prognósticos que obtén os datos meteorolóxicos da base de datos para procesalos e convertelos en un texto en linguaxe natural. Ese texto é almacenado na base de datos para que así a API de servizos web poda facilitáselo á interface. Todos os datos meteorolóxicos almacenados na base de datos son obtidos diariamente por unha aplicación provedora directamente da páxina web de MeteoGalicia.

Como a frecuencia de actualización dos datos é moi baixa (1 vez cada 24 horas) non é necesario que os prognósticos se xeren individualmente a petición do usuario. Neste caso é moito máis eficiente facelo de forma asíncrona, é dicir, ter os textos xa xerados e gardados na base de datos diminuíndo o tempo de resposta do servidor ao non ter que xerar un texto en cada unha das peticións dos clientes. Este proceso é similar ao de MeteoGalicia coa condición de que eles actualizan os seus datos de predición 2 veces cada 24 horas.

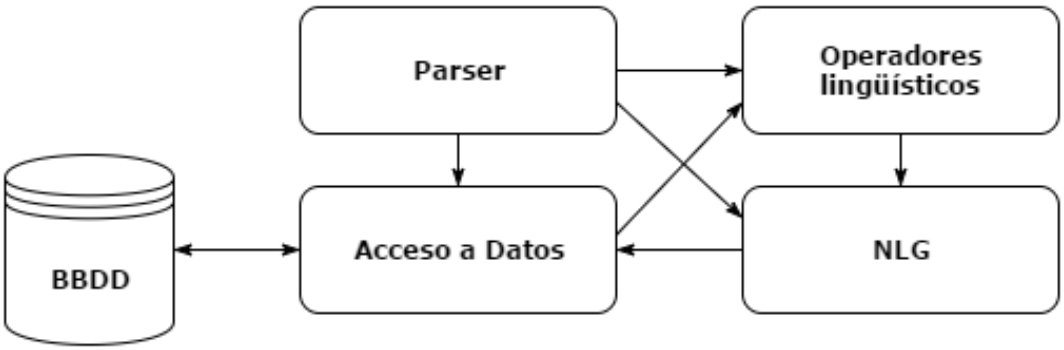


Figura 4.2: Arquitectura da aplicación xeradora de prognósticos.

A aplicación xeradora divídese en 4 módulos, como se mostra na figura 4.2:

- Un modulo de acceso a datos, encargado de ler e escribir os datos meteorolóxicos e predicións da base de datos correspondente.
- Un módulo *parser* encargado de ler todos os datos relativos á configuración da aplicación e implementar as tarefas de logging.
- Un módulo que converte os datos de entrada en descricións lingüísticas intermedias.
- Un módulo que converte esas descricións lingüísticas en textos en linguaxe natural.

O funcionamento da aplicación detallarase en máis profundidade no capitulo 5 desta memoria.

4.2. Patróns de deseño

Os patróns de deseño [16] son o esqueleto das solucións a problemas comúns no desenvolvemento de software. Noutras palabras, brindan unha solución xa probada e documentada a problemas de desenvolvemento de software que están suxeitos a contextos similares. Existen tres categorías de patróns de deseño:

- **Patróns de creación:** inicialización e configuración de obxectos.
- **Patróns estruturais:** separan a interface da implementación.
- **Patróns de comportamento:** describen a comunicación entre obxectos ou clases.

Neste proxecto utilízanse os seguintes patróns de deseño:

- **Patrón Singleton:** Restrinxe a instanciación dunha clase ou valor dun tipo a un solo obxecto. Permite garantir que unha clase só teña unha instancia e proporcionar un punto de acceso global a ela. Isto conséguese creando na nosa clase un método que cree unha instancia do obxecto só se aínda non existe algunha. Para asegurar que a clase non pode ser instanciada novamente se regula o alcance do construtor facéndoo privado.
- **Patrón MVC:** O patrón MVC (Modelo-Vista-Controlador) plántea a separación do problema en tres capas: o modelo, que representa a realidade; o controlador, que controlar os eventos da aplicación; e a vista, que é utilizada para interaccionar co usuario.
- **Patrón DAO:** O patrón DAO (Data Access Object) utilizase para abstraer e encapsular todos os accesos á base de datos. O DAO manexa a conexión coa base para obter e almacenar os datos.

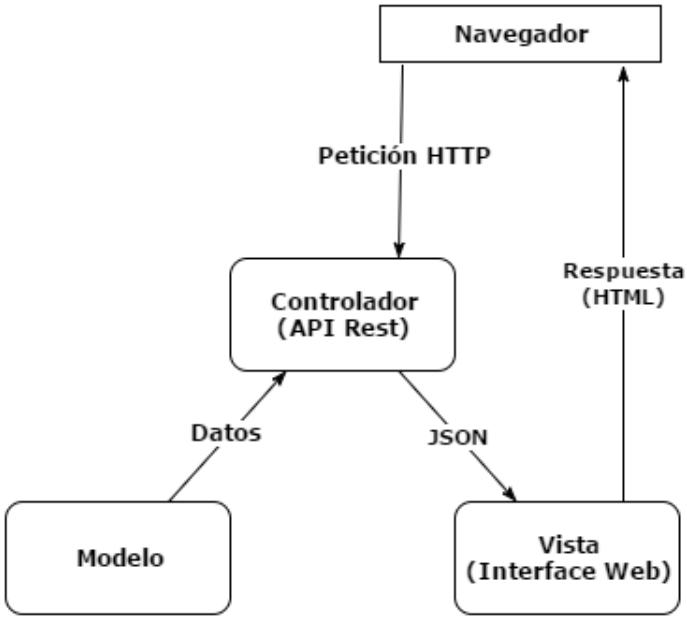


Figura 4.3: Patrón MVC.

4.3. Ferramentas

4.3.1. Librarías

SimpleNLG: Xeración de Texto en Linguaxe Natural

Para a xeración de texto en linguaxe natural utilizarase a libraría SimpleNLG [6]. SimpleNLG é unha libraría escrita en Java que permite a un programa xerar frases ortográfica, sintática e gramaticalmente correctas en inglés. Á hora de crear unha estructura sintactica e convertila en texto, estos son os pasos que a libraria segue:

1. Inicialización dos compoñentes básicos necesarios; cos obxectos léxicos necesarios.

2. Uso das operacións proporcionadas pola API para configurar as características (*features*) dos compoñentes.

3. Combinación dos compoñentes para formar estruturas máis grandes e complexas.

4. Proporcionar a estrutura resultante ao linealizador, aplicando as inflexións correctas e ordenación linear dependendo das *features*, antes de devolver o texto resultante.

A libraría foi desenvolvida polo grupo de lingüística computacion (CLAN) da Universidade de Aberdeen e inclúe un extenso tutorial ademais da documentación da API e un foro de consultas.

SimpleNLG permítenos traballar coas estruturas lingüísticas máis comúns (oracións nominais, verbais, enumeracións, preguntas, etc.) e combinándoas entre elas podemos formar textos realmente complexos. Ademais existen varias adaptacións da libraría para dar soporte a idiomas coma o Francés ou o Portugués.

Fuzzy4j: Lóxica Difusa en Java

Nos operadores lingüísticos facemos uso da lóxica difusa ou borrosa (fuzzy logic) [18] para transformar os datos do estado do ceo en descricións lingüísticas o máis precisas posible.

A lóxica difusa permítenos introducir valores intermedios entre a afirmación completa ou a negación absoluta de igual maneira que poida definirse termos e expresións que non son totalmente certas nin totalmente falsas. Por exemplo, podemos considerar intuitivamente que unha persoa é alta se mide máis de 1.8 metros, pero de igual maneira consideramos a unha persoa como alta se mide 1.79 metros. Sen embargo, se o concepto de persoa alta defínese seguindo unha aproximación booleana, unha persoa que mida 1.79 metros podería ser considerada como baixa, tal e como se ilustra nas funcións de pertenza da Fig. 4.4 (esquerda), mentres que empregando definicións borrosas a altura pode definirse coma un concepto gradual (Fig. 4.4 dereita).

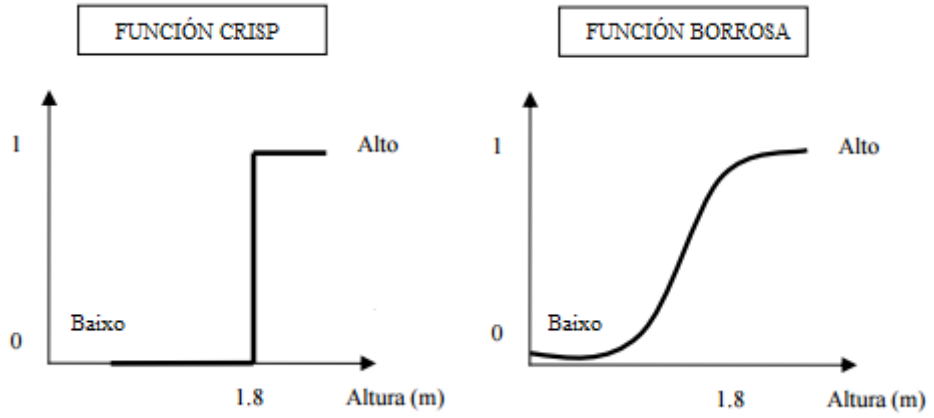


Figura 4.4: Funcións de pertenza do conxunto borroso .ªlto".

Fuzzy4j [17] é unha libraría Java que implementa as funcións mais comúns dentro da lóxica difusa nas areas dos conxuntos difusos, agregación difusa e controladores difusos. Neste caso solo utilizaremos a función trapezoidal para obter unha descrición da evolución do estado do ceo.

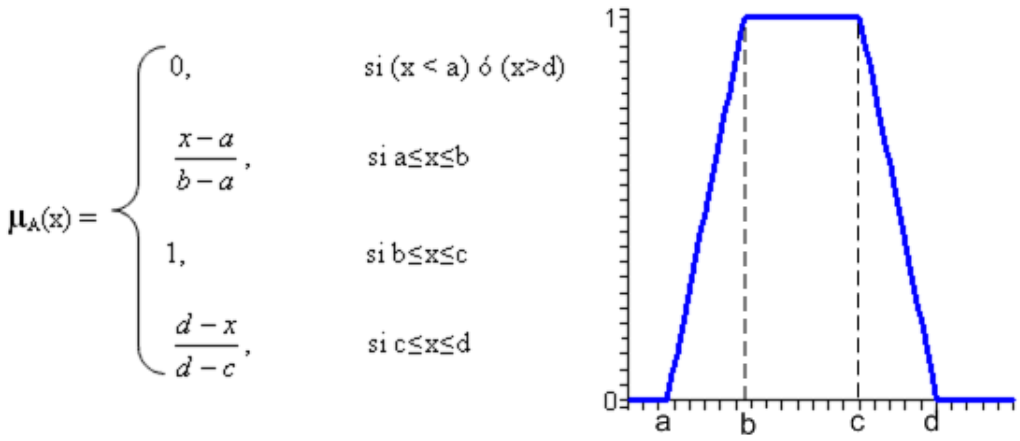


Figura 4.5: Función de pertenza trapezoidal.

Jersey: Servizos RESTFul en Java

Os datos almacenados na base de datos serán accesibles pola interface web grazas aos diferentes servizos REST implementados. A Transferencia de Estado Representacion (Representational State Transfer) ou REST [19] é un estilo de arquitectura software para sistemas hipermedia distribuídos como a World Wide Web. O término orixinouse no ano 2000, nunha tese doutoral sobre a web escrita por Roy Fielding, un dos principais autores da especificación do protocolo HTTP e pasou a ser amplamente utilizado pola comunidade de desenvolvemento.

Rest se asenta sobre o protocolo HTTP que permite a comunicación sen estado entre o cliente e o servidor de maneira que cada mensaxe HTTP contén toda a información necesaria para comprender a petición. A súa vez proporciona un conxunto de operacións ben definidas (POST, GET, PUT e DELETE) e unha sintaxe universal, polo que cada recurso é direccionable unicamente a través da súa URI.

Neste proxecto utilizouse a libraría Jersey [21] que facilita o desarrollo de servizos RESTful grazas a que proporciona unha API estándar e portable. As peticións na interface realizaranse en JavaScript mediante a tecnoloxía AJAX.

Outras librarías

Outras librarías utilizadas pero con menos peso sobre o desempeño da aplicación son:

- **JDBC de SQLite:** Proporciona as funcións para ler e almacenar os datos da base de datos.
- **Libraría Lang3 de Apache Commons:** Proporciona funcións para o tratamento de arrays.
- **JavaTuples:** Proporciona estruturas de datos para traballar con tuplas.
- **jQuery:** Biblioteca JavaScript que permite simplificar a maneira de interactuar cos documentos HTML, manipular a arbore DOM, manexar eventos, desenvolver animacións e engadir interacción coa técnica AJAX [22].

4.3.2. Outras Ferramentas

Desenvolvemento

Co obxectivo de maximizar a produtividade e reducir o tempo de aprendizaxe e adaptación as deferentes ferramentas, o entorno de traballo seleccionado foi o seguinte:

- **Sistema Operativo:** Windows 10 Pro.
- **IDE:** Netbeans IDE para a codificación de jGALiWeather e PyCharm IDE para revisar o código fonte da aplicación orixinal en Python, GALiWeather.
- **Servidor de aplicacións:** Utilizamos o servidor Apache Tomcat por ter un 60 % da cota de mercado actual.
- **Base de Datos:** Utilizamos unha base de datos SQLite xa que foi a que se proporcionou dende o CiTIUS cos datos meteorolóxicos proporcionados por MeteoGalicia nos últimos anos.

Documentación

Para a realización da documentación do proxecto se utilizaron a seguintes ferramentas:

- **Microsoft Office:** Suite ofimática para a realización da documentación durante a transcurso do proxecto.

- **TexMaker:** IDE de LaTeX utilizado na realización desta memoria.
- **Microsoft Project:** Utilizado para a creación de diagramas de Gantt para xustificar os prazos do proxecto.
- **Draw.io:** Aplicación web para a creación de diagramas en xeral.

Capítulo 5

Deseño e Implementación

O deseño é o proceso que traduce os requisitos nunha representación do software de forma que poida coñecerse a arquitectura, funcionalidade e incluso, a calidade do mesmo antes de comezar coa codificación. Ademais, realizar un deseño previo da aplicación é unha forma de resolver problemas nas fases máis tempéranas do proxecto, aforrando costes. En Scrum o deseño é continuo e vaise actualizando e completando en cada sprint.

Neste capitulo definirase o deseño de jGALiWeather e como se realizou a codificación a partir dese deseño. Debido as grandes diferencias entre una aplicación web (interface de visualización) e unha aplicación clásica (xerador de prognósticos e parser), os deseños e implementacións destes dous programas abordarase en seccións separadas. A ocorrencia do risco **R04**, fixo necesaria a busca dunha maneira alternativa de obter datos meteorolóxicos actualizados. Ao final decidiuse crear un *parser* que recuperara a información necesaria directamente do código HTML da web de MeteoGalicia. Esta aplicacación tamen terá a súa propia sección.

A partir de agora lle chamaremos **jGALiWeather** á aplicación xeradora de textos en linguaxe natural, **interface web** á aplicación web de visualización dos datos e **proveedor de datos** ao programa encargado de *parsear* a web de Meteogalicia para proveer datos meteorolóxicos actualizados á base de datos.

5.1. jGALiWeather

jGALiWeather emprega datos de predición simbólico-numéricos e información experta adicional para xerar as predicións textuais finais. Esta tarefa divídese en dous procesos independentes. En primeiro lugar, os datos convértense en descricións lingüísticas (codificadas nun linguaxe intermedio). Estas descricións créanse empregando un método computacional que abstrae os datos en etiquetas lingüísticas. O proceso final converte estas descricións en textos en inglés. A figura 5.1

mostra un esquema global das tarefas que leva a cabo a aplicación [1].

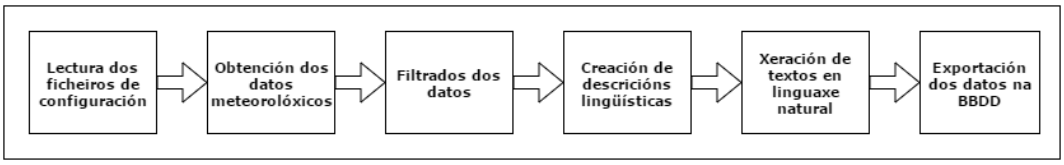


Figura 5.1: Fluxo de de tarefas da aplicación.

En primeiro lugar a aplicación lee os ficheiros de configuración recuperando información tal coma: datos de conexión a base de datos, valores válidos para os meteoros ou as particións na que se agruparan os datos meteorolóxicos á hora de crear o mellor prognóstico posible. Posteriormente obteñense os datos de predición e fíltranse para evitar erros nas fases posteriores.

Unha vez filtrados os datos créanse as descricións lingüísticas intermedias que permitirán a xerar textos en linguaxe natural o máis precisos posibles. Por último, todos estes prognósticos textuais gárdanse na base de datos.

A diferencia de GALiWeather, jGALiWeather traduce ao linguaxe de programación Java á maior parte dos seus módulos para despois substituír a parte de xeración de textos por outro módulo implementado utilizando técnicas máis sofisticadas de NLG coa libraría SimpleNLG. O módulo de acceso a datos tamén é substituído dado que non temos acceso á base de datos de MeteoGalicia e temos que crear unha base de datos local cos datos meteorolóxicos recuperados da súa paxina web.

5.1.1. Diagramas de Fluxo de Datos

A creación de diagramas de fluxo de datos (DFD) é unha técnica gráfica que utiliza un diagrama en forma de rede para representar o fluxo de información e as transformacións que se aplican aos datos ao moverse dende a entrada á saída. Os DFDs describen de forma gráfica o que fai o sistema, pero non proporcionan información nin de cando nin de en que secuencia se fai. Para describir o sistema desde este punto de vista utilizaremos tanto diagramas de fluxo de datos como especificacións de proceso para describir as primitivas de proceso.

Diagrama de Contexto

O diagrama de contexto é o DFD de nivel 0 no que o sistema esta representado por un só proceso que resume o requisito principal do sistema. Nel represéntanse todas as entidades externas coas que se relaciona o sistema.

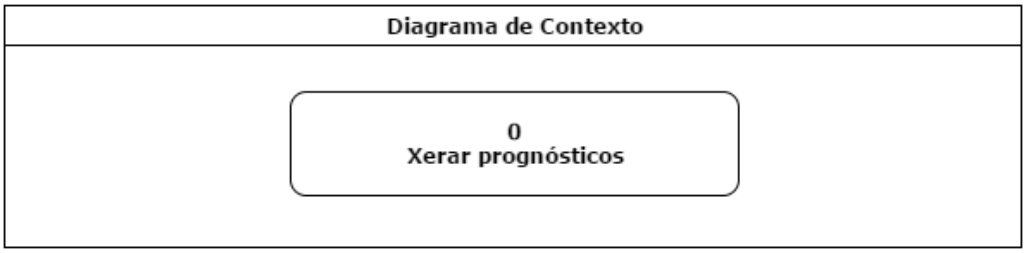


Figura 5.2: Diagrama de Contexto.

DFD de nivel 1

Na figura 5.3 podemos ver o DFD resultante da explosión do proceso 0 do diagrama de contexto da figura 5.2.

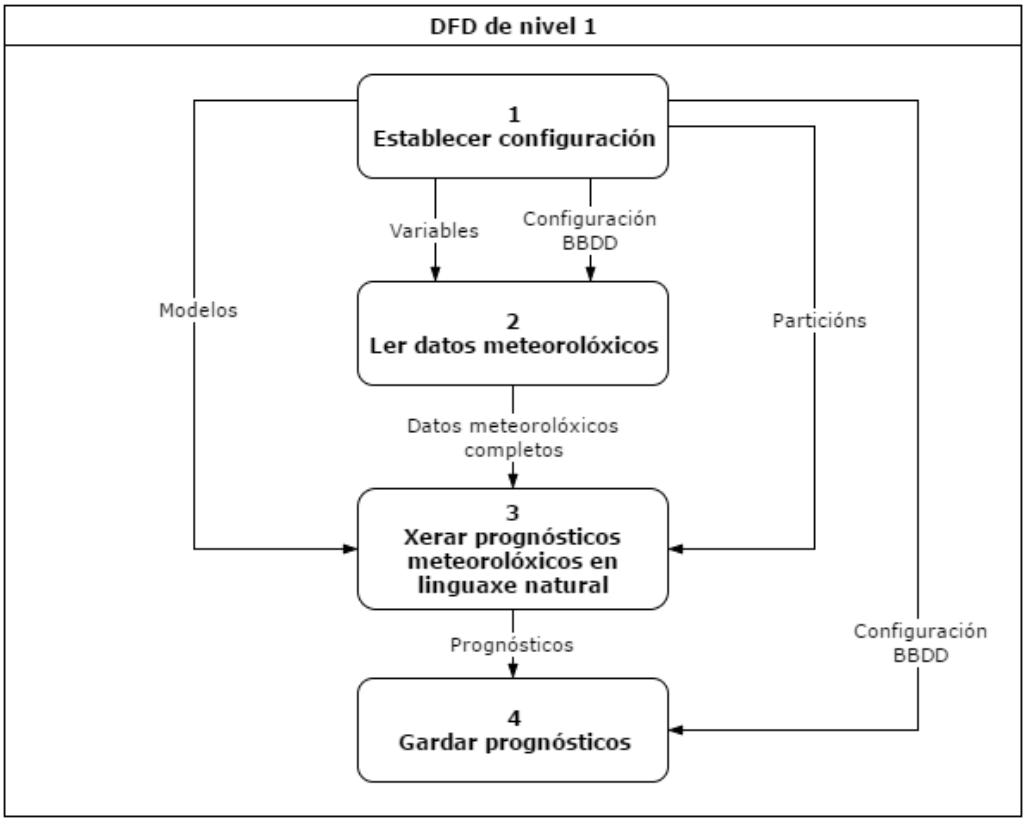


Figura 5.3: DFD de nivel 1.

DFD de nivel 2 - Establecer configuración

Na figura 5.4 podemos ver o DFD resultante da explosión do proceso 1 do diagrama de fluxo de datos de nivel 1 da figura 5.3.

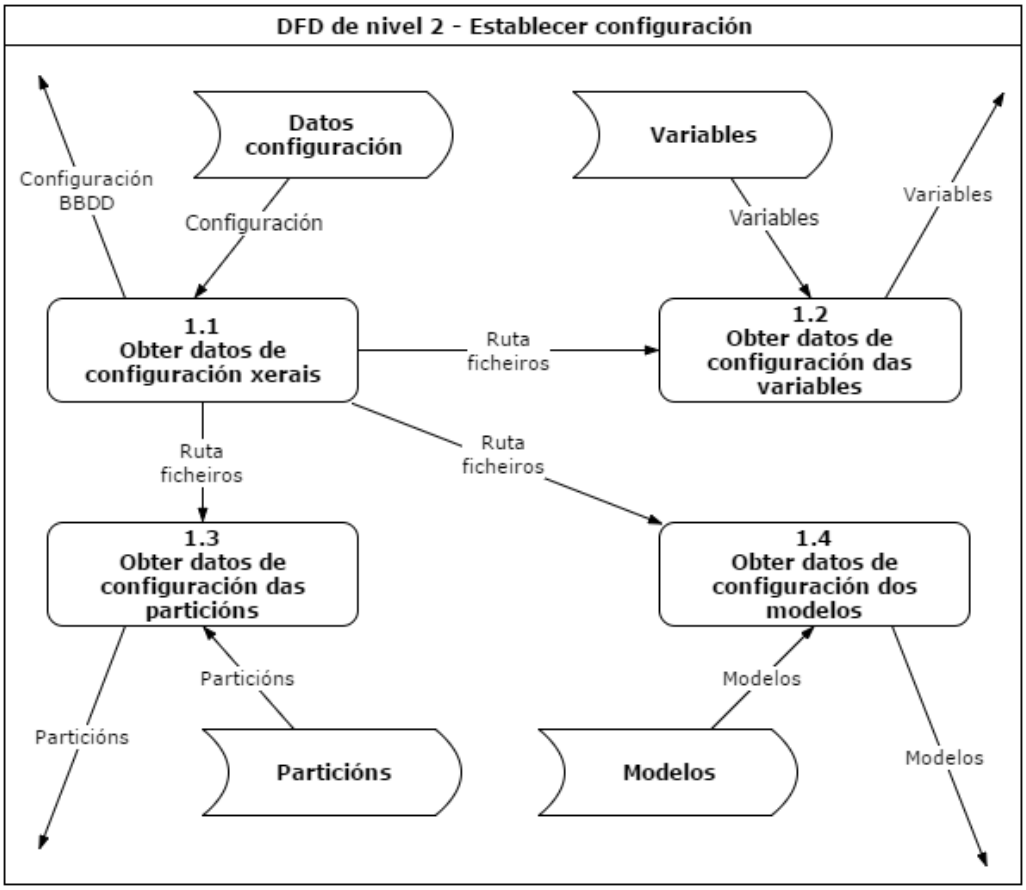


Figura 5.4: DFD de nivel 2 - Establecer configuración.

Especificacións de proceso:

- **Proceso 1.1:** Obter datos de configuración xerais.
  - Precondición: Existe un ficheiro de configuración na ruta adecuada.
  - Postcondición: Devólvense as estruturas de datos adecuadas coa configuración da aplicación.
  - Descrición: O proceso abre o ficheiro de configuración e parsea o seu contido almacenándoo nas estruturas de datos adecuadas. Os datos contidos nese ficheiro son: a rutas dos demais ficheiros de configuración e os datos de conexión á base de datos.
- **Proceso 1.2:** Obter datos de configuración das variables.
  - Precondición: Existe un ficheiro de configuración na ruta adecuada.
  - Postcondición: Devólvense as estruturas de datos adecuadas coa configuración da aplicación.

  - Descrición: O proceso abre o ficheiro de configuración e parsea o seu contido almacenándoo nas estruturas de datos adecuadas. Os datos contidos nese ficheiro son o numero de valores e valores válidos para as temperaturas, os meteoros do vento, os meteoros do estado do aire e o ICA.

- **Proceso 1.3:** Obter datos de configuración das particións.

  - Precondición: Existe un ficheiro de configuración na ruta adecuada.
  - Postcondición: Devólvense as estruturas de datos adecuadas coa configuración da aplicación.
  - Descrición: O proceso abre o ficheiro de configuración e parsea o seu contido almacenándoo nas estruturas de datos adecuadas. Neste ficheiro temos os datos necesarios para clasificar os fenómenos meteorolóxicos dependendo do seu valor.

- **Proceso 1.4:** Obter datos de configuración dos modelos.

  - Precondición: Existe un ficheiro de configuración na ruta adecuada.
  - Postcondición: Devólvense as estruturas de datos adecuadas coa configuración da aplicación.
  - Descrición: O proceso abre o ficheiro de configuración e parsea o seu contido almacenándoo nas estruturas de datos adecuadas. Os diferentos elementos textuais necesarios para crear os prognósticos en linguaxe natural están almacenados neste ficheiro.

DFD de nivel 2 - Ler datos meteorolóxicos

Na figura 5.5 podemos ver o DFD resultante da explosión do proceso 2 do diagrama de fluxo de datos de nivel 1 da figura 5.3.

Especificacións de proceso:

- **Proceso 2.1:** Obter localizacións.

  - Precondición: Os datos de conexión á base de datos son válidos.
  - Postcondición: Devólvense todos os concellos xunto cos seus identificadores.
  - Descrición: O proceso abre unha conexión coa base de datos e recupera todos os rexistros da táboa que contén todos os concellos de Galicia.

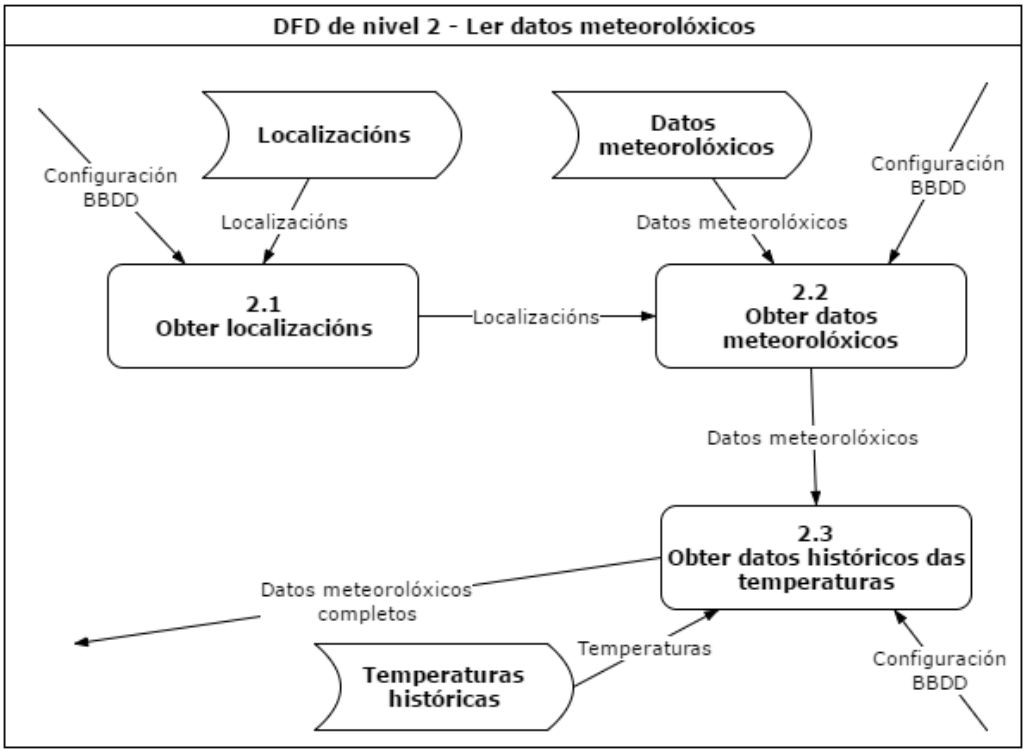


Figura 5.5: DFD de nivel 2 - Ler datos meteorolóxicos

- **Proceso 2.2:** Obter datos meteorolóxicos.
  - Precondición: Os datos de conexión á base de datos son válidos.
  - Postcondición: Devólvense todas as localización xunto coas predicións dos próximos días.
  - Descrición: O proceso obtén os datos meteorolóxicos para a data axeitada de cada localización. Eses datos engádense á estrutura de datos que representa cada concello.
- **Proceso 2.3:** Obter datos históricos das temperaturas.
  - Precondición: Os datos de conexión á base de datos son válidos.
  - Postcondición: Devólvense todas as localización xunto coas predicións dos próximos días e os valores históricos de temperatura.
  - Descrición: O proceso obtén os datos da media e da desviación típica para as temperaturas máximas e mínimas dos últimos 50 anos para cada localización. Eses datos engádense á estrutura de datos que representa cada concello.

DFD de nivel 2 - Xerar prognósticos meteorolóxicos en linguaxe natural

Na figura 5.6 podemos ver o DFD resultante da explosión do proceso 3 do diagrama de fluxo de datos de nivel 1 da figura 5.3.

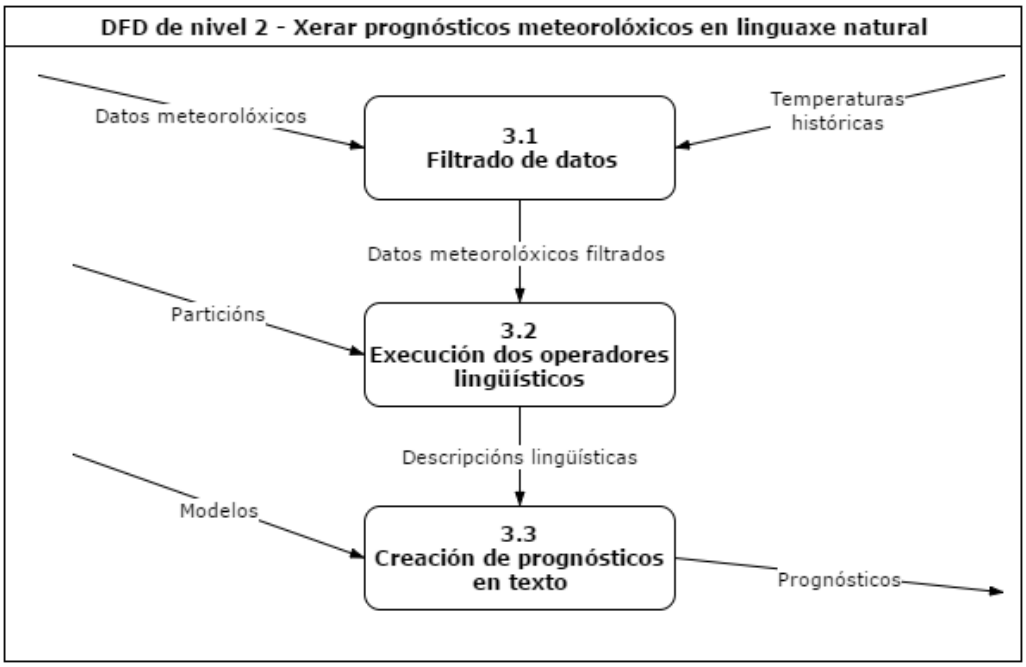


Figura 5.6: DFD de nivel 2 - Xerar prognósticos meteorolóxicos en linguaxe natural

Especificacións de proceso:

- **Proceso 3.1:** Filtrado de datos.
  - Precondición: Existen datos meteorolóxicos de cada localización.
  - Postcondición: Devólvense todos os concellos xunto cos seus datos meteorolóxicos corrixidos.
  - Descrición: En primeiro lugar o proceso determina se os datos meteorolóxicos son válidos e se o número de valores é correcto. Despois se procesan os valores das temperaturas para determinar se son uns valores válidos para esa época do ano ou non. Por ultimo corríxense os valores das funcións trapezoidais.
- **Proceso 3.2:** Execución dos operadores lingüísticos.
  - Precondición: Os datos de meteorolóxicos son válidos.
  - Postcondición: Devólvense as descricións lingüísticas intermedias para cada localización.

  - Descrición: Execútanse os diferentes operadores lingüísticos para cada concello convertendo os datos meteorolóxicos en descricións lingüísticas intermedias. Existen os seguintes operadores: dous operadores distintos para o estado do ceo, un operador para o vento, un operador para a chuvia, un operador para néboa, outro para a temperatura e un ultimo operador para o índice da calidade do aire.

- **Proceso 3.3:** Creación de prognósticos en texto.

  - Precondición: As descricións lingüísticas intermedias son válidas.
  - Postcondición: Devólvense os prognósticos textuais para cada localización.
  - Descrición: Cada unha das diferentes descricións lingüísticas convértense nun prognóstico textual utilizando técnicas NLG.

DFD de nivel 2 - Gardar prognósticos

Na figura 5.7 podemos ver o DFD resultante da explosión do proceso 4 do diagrama de fluxo de datos de nivel 1 da figura 5.3.

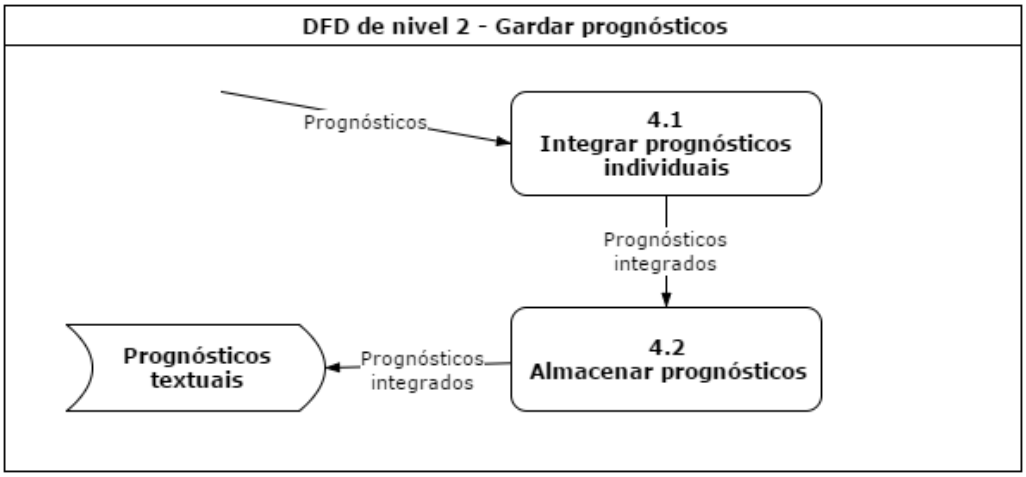


Figura 5.7: DFD de nivel 2 - Gardar prognósticos

Especificacións de proceso:

- **Proceso 4.1:** Integrar prognósticos individuais.

  - Precondición: Existen prognósticos individuais para cada tipo de dato meteorolóxico.

  - Postcondición: Devólvense todos os prognósticos integrados para cada localización.
  - Descrición: As estruturas de datos da libraría SimpleNLG utilizadas para crear prognósticos individuais para cada tipo de dato xúntanse para crear un texto máis complexo albergando todo eses prognósticos independentes.

- **Proceso 4.2:** Almacenar prognósticos.

  - Precondición: Os prognósticos textuais son válidos.
  - Postcondición: Existen novas entradas na táboa de prognósticos da base de datos.
  - Descrición: Neste proceso os prognósticos de cada localización engádense á base de datos indicando a data no que se engadiron.

5.1.2. Implementación da aplicación

jGALiWeather é unha aplicación automática de consola, polo que non dispoñe de interface gráfica e os únicos elementos que a compoñen son as estruturas de datos e os módulos que procesan eses datos.

De maneira máis detallada, a aplicación ten a seguinte estrutura de paquetes e clases:

- Clase **PredictionSummarizer**. Clase encargada de levar a cabo a xeración de prognósticos textuais a curto prazo. Para elo sérvese das funcionalidades que proporcionan os módulos do resto de paquetes.
- Paquete **algorithm**. Contén os operadores que converten os datos de entrada en descricións lingüísticas intermedias:
  - Paquete **weather_operators**. Contén os operadores lingüísticos que converten os datos de predición meteorolóxica en descricións intermedias:
    - Clase **SkyStateAOperator**. Esta clase implementa un operador que xera unha descrición lingüística da cobertura nubosa, incluíndo información do estado do ceo en diferentes subperíodos cronolóxicos.
    - Clase **SkyStateBOperator**. Esta clase implementa un operador que xera unha descrición lingüística da cobertura nubosa, incluíndo información do estado do ceo para todo o período, utilizando cuantificadores.

    - Clase **RainOperator**. Esta clase implementa un operador que xera unha descrición lingüística das precipitacións, incluíndo información sobre a intensidade das mesmas.
    - Clase **WindOperator**. Esta clase implementa un operador que xera unha descrición lingüística do vento, incluíndo información acerca dos episodios de vento e os cambios na dirección ou na intensidade.
    - Clase **FogOperator**. Esta clase implementa un operador que xera unha descrición lingüística da néboa, incluíndo información sobre os períodos de néboa.
    - Clase **TemperatureOperator**. Esta clase implementa un operador que xera unha descrición lingüística das temperaturas, incluíndo información acerca da variación na temperatura.
  - Paquete **ica_operators**. Contén un conxunto de operadores que extraen a información relevante dos datos de predición meteorolóxica e un operador que xera a descrición intermedia de calidade do aire:
    - Clase **ICASkyStateOperator**. Esta clase implementa un operador que mide o ratio de aparición de cada episodio de cobertura nubosa posible.
    - Clase **ICAWindOperator**. Esta clase implementa un operador que mide o ratio de aparición de períodos de vento relevantes.
    - Clase **ICARainOperator**. Esta clase implementa un operador que mide o ratio de aparición de períodos de precipitacións relevantes.
    - Clase **ICAConditionsOperator**. Esta clase implementa un operador que inclúe regras expertas para determinar a situación meteorolóxica máis precisa posible, tendo en conta os valores do vento, das choivas e da cobertura nubosa adquiridos dos demais operadores.
    - Clase **ICAOperator**. Esta clase implementa un operador que obtén a tendencia do estado da calidade do aire baseándose nas súas variacións.

- Paquete **database**. Contén a clase de conexión coa base de datos, que implementa consultas para a obtención e actualización da información:
  - Clase **DatabaseConnector**. Esta clase permite conectarse a unha base de datos cuxa estrutura sexa similar á que implementa MeteoGalicia, co fin de extraer os datos meteorolóxicos e de calidade do aire para a súa conversión en textos en linguaxe natural.

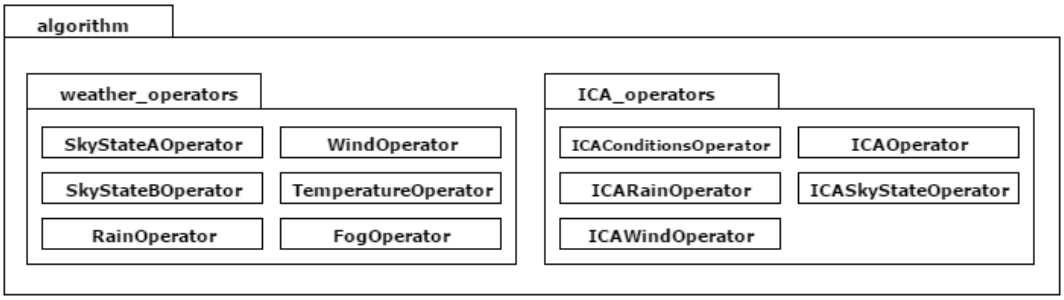


Figura 5.8: Estrutura de ficheiros do paquete **algorithm**.

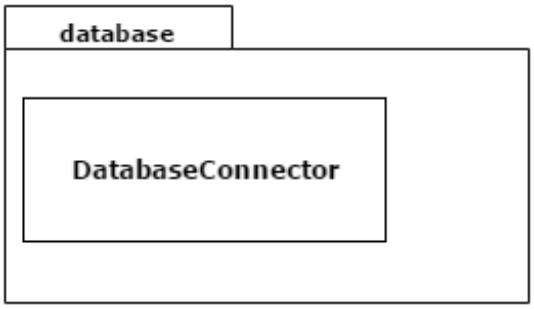


Figura 5.9: Estrutura de ficheiros do paquete **database**.

- Paquete **data**. Conten estruturas de datos necesarias para o correcto funcionamento da aplicación e unha serie de filtros para comprobar a validez dos datos:
  - Paquete **data_filters**. Contén os filtros utilizados na aplicación:
    - Clase **DataFilter**. Clase que determina se os datos de entrada son válidos e elimina os datos inválidos, no caso de que sexa posible.
    - Clase **DataLengthFilter**. Clase que determina se o número de datos de entrada é soportado pola aplicación.
    - Clase **MeteorologicFilter**. Clase que simplifica os valores numéricos que indican o estado do ceo en caso necesario.
- Paquete **configuration**. Contén os módulos encargados de ler todos os datos relativos á configuración da aplicación e un módulo que implementa as tarefas de *logging*:
  - Paquete **configuration_reader**. Contén as estruturas de datos e a clase de procesamento de datos necesarias para obter a configuración xeral da aplicación:
    - Clase **ConfigurationReader**. Clase que lee e proporciona ao resto de módulos a información almacenada no arquivo de configuración xeral da aplicación (**configuration.xml**).

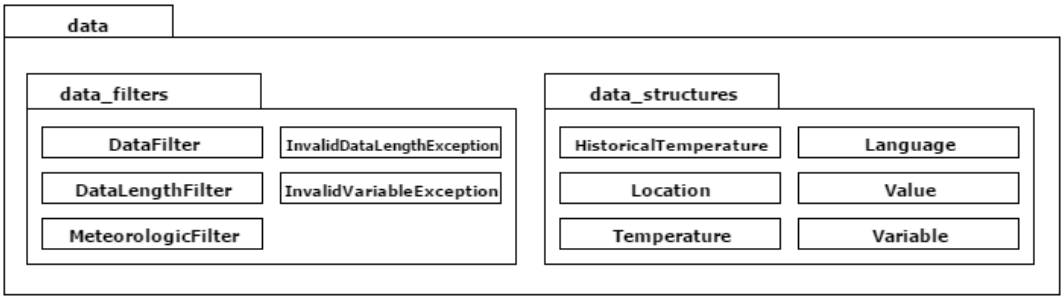


Figura 5.10: Estrutura de ficheiros do paquete **data**.

- Paquete **variable_reader**. Contén as estruturas de datos e a clase de procesamento de datos necesarias para obter a definición das variables que se utilizan para a xeración dos textos:
  - Clase **VariableReader**. Clase que lee a información das variables da aplicación (**variables.xml**) e proporciona dita información ao resto de módulos da aplicación.
- Paquete **partition_reader**. Contén unha serie de estruturas de datos e unha clase encargada de procesar eses datos para obter a información referente as particións:
  - Clase **PartitionReader**. Clase que lee a información que define as particións numéricas asociadas a etiquetas no arquivo de configuración correspondente (**partitions.xml**).
- Paquete **template_reader**. Contén unha serie de estruturas de datos e unha clase encargada de procesar eses datos para obter a información referente aos modelos:
  - Clase **TemplateReader**. Clase que lee a información que define os modelos necesarios para xeración de texto en linguaxe natural no arquivo de configuración correspondente (**partitions.xml**).
- Paquete **logger**:
  - Clase **GALiLogger**. Clase encargada de proporcionar servizos de *logging* ao resto de módulos da aplicación, có fin de informar do estado da execución ao longo do proceso.

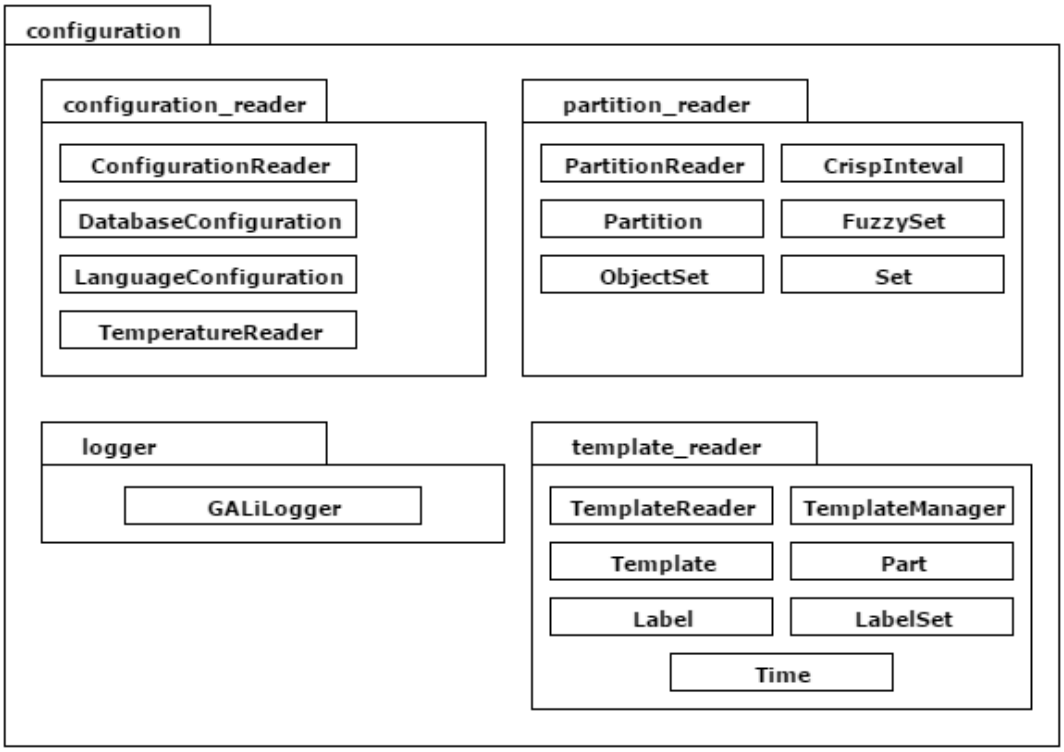


Figura 5.11: Estrutura de ficheiros do paquete **configuration**.

- Paquete **nlg**. Contén un conxunto de módulos que permiten a conversión de descricións lingüísticas en textos en linguaxe natural:
  - Clase **DescriptionAggregator**. Esta clase encargase de agregar os textos finais de cada variable meteorolóxica para proporcionar unha predición textual completa.
  - Paquete **nlg_generators**. Contén os compoñentes que, para cada variable meteorolóxica de entrada, converten unha descrición lingüística nunha predición textual. Ditos compoñentes combinan coñecemento experto con técnicas de xeración de linguaxe natural, mediante as funcións proporcionadas pola libraría SimpleNLG.
    - Clase **SkyCoverageGeneratorLevel1**. Clase encargada de xerar unha descrición cronolóxica do estado do ceo.
    - Clase **SkyCoverageGeneratorLevel2**. Clase encargada de xerar unha descrición cuantitativa do estado do ceo.
    - Clase **RainGenerator**. Clase encargada de xerar unha descrición cronolóxica dos episodios de precipitacións.
    - Clase **WindGenerator**. Clase encargada de xerar unha descrición dos episodios de vento.

- Clase **FogGenerator**. Clase encargada de xerar unha descrición dos episodios de néboa.
- Clase **TemperatureGenerator**. Clase encargada de xerar unha descrición do estado das temperaturas e a súa tendencia.
- Clase **ICAGenerator**. Clase encargada de xerar unha descrición xeral do estado do índice de calidade do aire.

– Paquete **precipitation_nlg**. Contén compoñentes e estruturas de datos utilizados para a xeración de textos en linguaxe natural para os episodios de precipitacións.

– Paquete **wind_nlg**. Da mesma maneira que o paquete anterior, contén compoñentes e estruturas de datos utilizados para xerar textos de predición do vento.

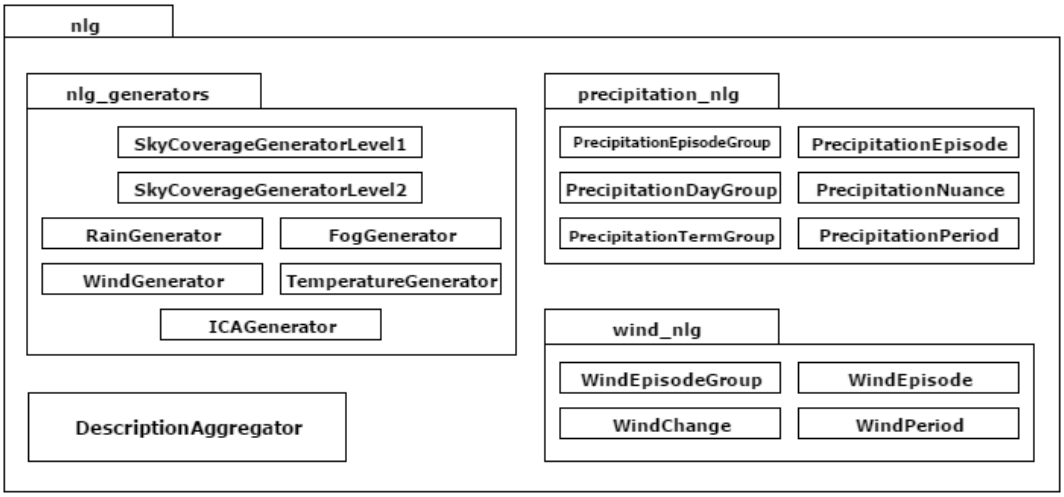


Figura 5.12: Estrutura de ficheiros do paquete **nlg**.

As clases non comentadas nesta sección trátanse de clases que teñen un papel secundario ou auxiliar de cara as clases e módulos explicados anteriormente.

5.1.3. Descrición do funcionamiento da aplicación

Lectura dos ficheiros de configuración

Para poder levar a cabo o resto de tarefas, ambos servizos precisan dun conxunto de información almacenada en arquivos de configuración. En concreto, os seguintes ficheiros son necesarios para o funcionamento da aplicación:

- Arquivo de configuración xeral (**configuration.xml**): Contén a información de conexión á base de datos e a localización do resto dos ficheiros.

- Arquivo de definición de variables (**variables.xml**): Define as variables que se utilizan na xeración dos textos de predición operativa.
- Arquivo de definición de particións de variables (**partitions.xml**): Define particións numéricas asociadas a etiquetas lingüísticas, utilizadas na xeración de descricións lingüísticas.
- Arquivo de definición de elementos textuais (**template.xml**): Contén as palabras e expresións utilizadas no proceso de xeración de texto en linguaxe natural.

O módulo **configuration_reader** é o encargado de ler os ficheiros de configuración e cargar a información que conteñen, para poñela a disposición dos demais módulos [1].

Obtención de datos de predición

Unha vez cargados todos os datos de configuración, a aplicación conectase á base de datos para obter os datos de cada municipio. O módulo **database** é o encargado de administrar esta conexión e de executar as diferentes consultas.

Filtrado de datos de predición

Os datos obtidos deben pasar por unha serie de filtros que garanten a integridade dos datos e por tanto o correcto funcionamento dos módulos de xeración de descricións lingüísticas e linguaxe natural. O módulo **data_filters** contén un conxunto de filtros que controlan a aparición de datos non válidos e o número de datos válidos dispoñibles.

Xeración de descricións lingüísticas

Para a predición a curto prazo (módulo **weather_operator**) e o estado de calidade do aire (módulo **ica_operators**), un conxunto de operadores extraen a información relevante dos datos de entrada e xeran descricións lingüísticas para cada municipio, que consisten en conxuntos de etiquetas lingüísticas xunto con referencias temporais. En concreto entre ambos módulos implementan sete operadores principais:

- **Estado do ceo 1:** Proporciona unha descrición cronolóxica do estado do ceo.
- **Estado do ceo 2:** Proporciona unha descrición cuantitativa do estado do ceo.

- **Precipitación:** Proporciona unha descrición cronolóxica dos episodios de precipitacións.
- **Vento:** Proporciona unha descrición cronolóxica dos episodios de vento intenso.
- **Néboa:** Proporciona unha descrición cronolóxica dos episodios de néboa dos datos.
- **Temperatura:** Proporciona unha descrición do estado das temperaturas e a súa tendencia.
- **Calidade do aire:** Proporciona unha descrición xeral do estado do índice de calidade do aire.

Xeración de texto en linguaxe natural

Para cada operador de xeración de descricións lingüísticas existe un xerador de texto en linguaxe natural asociado, que converte a descrición nun texto adaptado ao estilo meteorólogo e que proporciona información relevante e facilmente comprensible polos usuarios.

O paquete **nlg_simpleNLG** contén os módulos encargados desta tarefa, sendo **nlg_generators** o módulo principal, no que se han implementado todos os operadores de xeración de linguaxe natural. Estes operadores, utilizando as descricións lingüísticas e as funcionalidades da libraría **SimpleNLG**, devolven textos de predición en inglés. Os módulos auxiliares **precipitation_nlg** e **wind_nlg** utilízanse para xeración de textos de precipitacións e vento, cuxa estrutura é máis complexa. Por ultimo, o módulo **nlg_aggregator**, é o encargado de agrupar os textos das distintas variables meteorolóxicas nun único texto.

Exportación de prognósticos

Unha vez se haxan xerado os textos para todos os municipios, estes son gardados na base de datos para que sexan accesibles dende a interface web.

5.2. Interface Web

Á interface web é unha *Single Page Application*, noutras palabras, a aplicación web ten unha sola paxina ou vista e actualizase o seu contido como resposta ás accións do usuario. Grazas a este concepto podemos intercambiar datos có servidor de maneira rápida e fluída. Ademais reducimos o máximo posible a carga

do servidor, xa que non só non ten que renderizar paxinas HTML en cada petición dun cliente senón que gran parte do procesamento trasladase aos clientes. A figura 5.13 mostra o deseño da única vista da interface web.

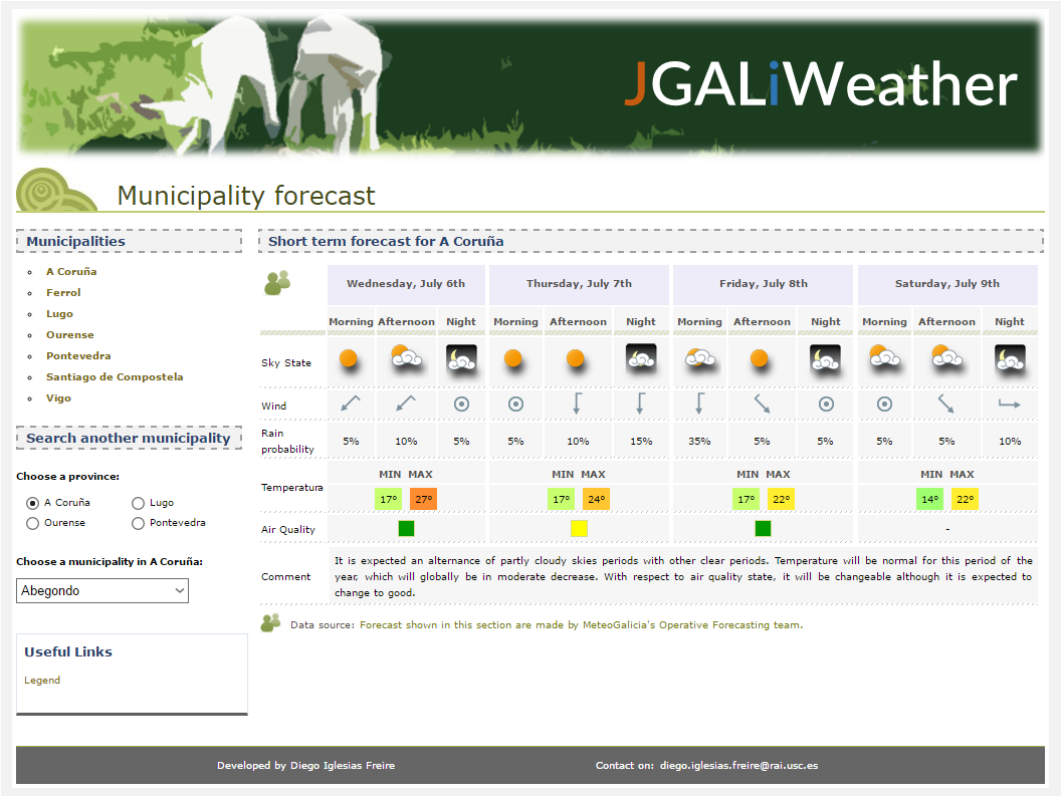


Figura 5.13: Plataforma web en funcionamento.

5.2.1. Controlador RESTFul

A arquitectura da aplicación modelase seguindo o patron de deseño MVC (Modelo-Vista-Controlador) cuxo funcionamento esta explicado no apartado 4.2. O controlador esta composto por un servizo RESTFul que accede á base de datos para recuperar os datos meteorolóxicos para que sexan accesibles dende o cliente. Este acceso realizase mediante URIs (*Uniform Resource Identifier*) e neste caso dispoñemos dunha única URI, como se mostra na táboa 5.1.

URI	Método
jgaliweather_api/v1/meteorologicalData/{id}	GET

Cadro 5.1: URI e método empregado.

Con esta URI podemos realizar unha petición ao servizo REST utilizando o metodo GET. O campo id é o identificador do municipio e a resposta estaría composta por un obxecto JSON (*JavaScript Object Notation*) que contén o nome do municipio, os datos meteorolóxicos e o prognostico textual. Un exemplo para de resposta deste servizo sería:

```
{
 name: "Fene",
 sky: [104, 101, 104, 104, 103, 103, 104, 103, 103, 101, 103, 103],
 wind: [302, 302, 299, 305, 307, 299, 305, 301, 300, 306, 307, 305],
 temp: [17, 26, 15, 24, 16, 25, 14, 26],
 ica: [1, 3, 1],
 rain_prob: ["5%", "15%", "10%", "10%", "15%", "10%", "10%", "5%", "
    5%", "5%", "5%", "5%"],
 comment: "It is expected an alternance of partly cloudy skies
    periods with other clear periods. Temperature will be normal for
     this period of the year, with minimums in moderate decrease and
     maximums without changes, despite that they will oscillate.
    With respect to air quality state, it will be changeable
    although it is expected to change to good."
}
```

O campo *name* indica o nome do municipio mentres que os demais campos son os datos meteorolóxicos para os vindeiros días. Tanto o campo *sky* como o campo *wind* teñen 12 valores numéricos, 3 para cada día, que indican o estado do ceo e a intensidade e dirección do vento respectivamente. Pola súa parte, o campo *temp* contén as temperaturas máximas e mínimas de cada día do período. Os campos *ica* e *rain_prob* conteñen os indices de calidade do aire e as probabilidades de chuvía para os próximos días respectivamente. Por último, *comment* contén o prognóstico textual xerado por jGALiWeather.

5.2.2. Xeolocalización

A aplicación fai uso da xeolocalización para determinar en que municipio se encontra o usuario e mostrarlle os datos pertinentes ao entrar no sitio web. Para implementar isto facemos uso API de xeolocalización de HTML5 xunto coa *Google Maps Geocoding API*.

A API de xeolocalización de HTML5 utiliza distintos métodos para obter as coordenadas xeográficas do dispositivo como poden ser a dirección IP, o GPS ou a triangulación de coordenadas a través de GSM entre outros. É responsabilidade dos navegadores implementar estes métodos polo que diferentes navegadores nun mesmo punto poden dar resultados dispares.

Unha vez temos as coordenadas do dispositivo, utilizamos a API *Geocoding* de Google Maps para obter a dirección correspondente a elas. No cliente mantense en todo momento un JSON que almacena todos os municipios de Galicia co seu identificador. Isto faise porque consideramos os municipios galegos coma datos estáticos que non van a variar nun período corto de tempo. Desta maneira podemos actualizar o buscador de municipios por provincia ou, coma neste caso, podemos obter o identificador dun concello a partir do seu nome sen necesidade de realizar consultas ao servidor. Grazas a isto unha vez sabemos a dirección na que se encontra o dispositivo, e polo tanto coñecer o seu municipio, xa podemos obter os datos meteorolóxicos facendo uso do servizo REST.

5.2.3. Interface Responsive

Á hora de deseñar a interface web tomáronse dúas decisións moi importantes. A primeira foi imitar, na maior medida posible, o deseño da web de MeteoGalicia e a segunda foi realizar un deseño *responsive* da aplicación. O deseño de MeteoGalicia claramente non é *responsive* polo que adaptalo para que se mostre de maneira correcta en dispositivos móbiles é unha tarefa complexa. Ao final, conseguiuse deseñar unha interface que cubría as nosas necesidades tal e como se mostra na figura 5.14.

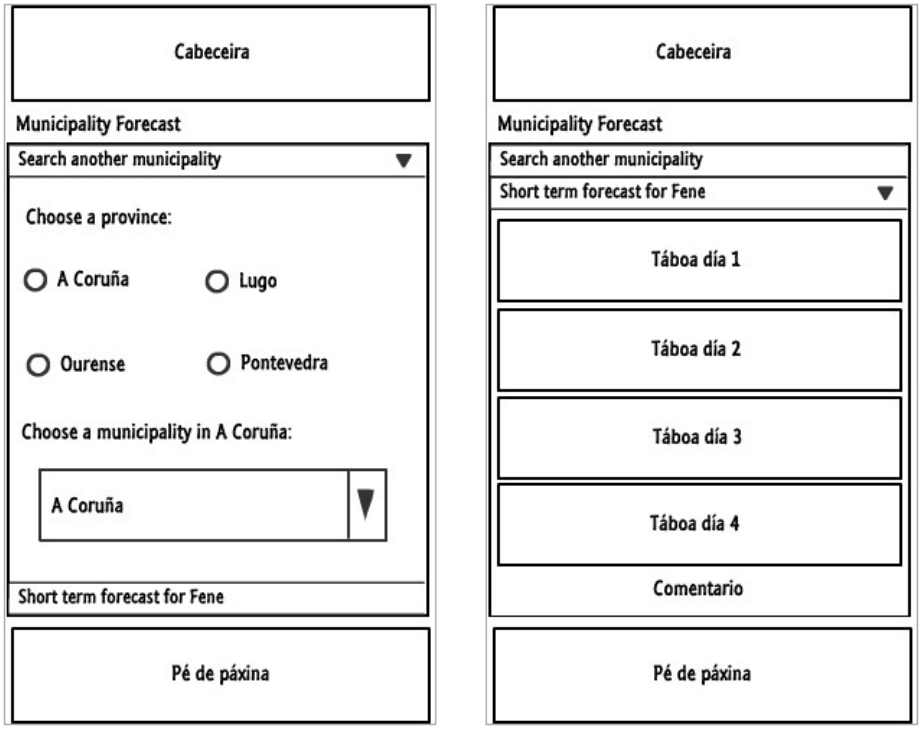


Figura 5.14: Prototipo da aplicación.

Para poder realizar isto facemos uso da libraria *jQuery Mobile* que proporciona os elementos necesarios para adaptar a interface a un entorno táctil. Desta maneira podemos converter un menú de selección de municipio claramente orientado a ser utilizado cun rato, nun menú totalmente usable dende un dispositivo cunha pantalla táctil. Ademais de isto, eliminamos elementos que poderían empeorar a experiencia de usuario e dividimos a aplicación en dous bloques. O primeiro bloque corresponde ao buscador mentres que no segundo se mostran os datos meteorolóxicos, ambos bloques poden expandirse e contraerse mediante unhas pestanas. Por último engadimos unha serie de animacións que fan que sexa unha experiencia máis satisfactoria navegar pola páxina. O resultado final mostrase na figura 5.15.

Para facer a experiencia máis agradable para os usuarios de *tablets* e de *smartphones* en formato apaisado realizouse un pequeno cambio na distribución das táboas que mostran os datos meteorolóxicos. Este novo deseño pode verse nas figura 5.16 e 5.17.

Esta libraría substitúe de forma automática elementos clásicos de HTML, como a etiqueta *select*, por elementos personalizados propios dela. Coma na interface de escritorio non queremos que a súa estrutura cambie esta libraría só se carga con tamaños de pantalla inferiores aos 1080px.

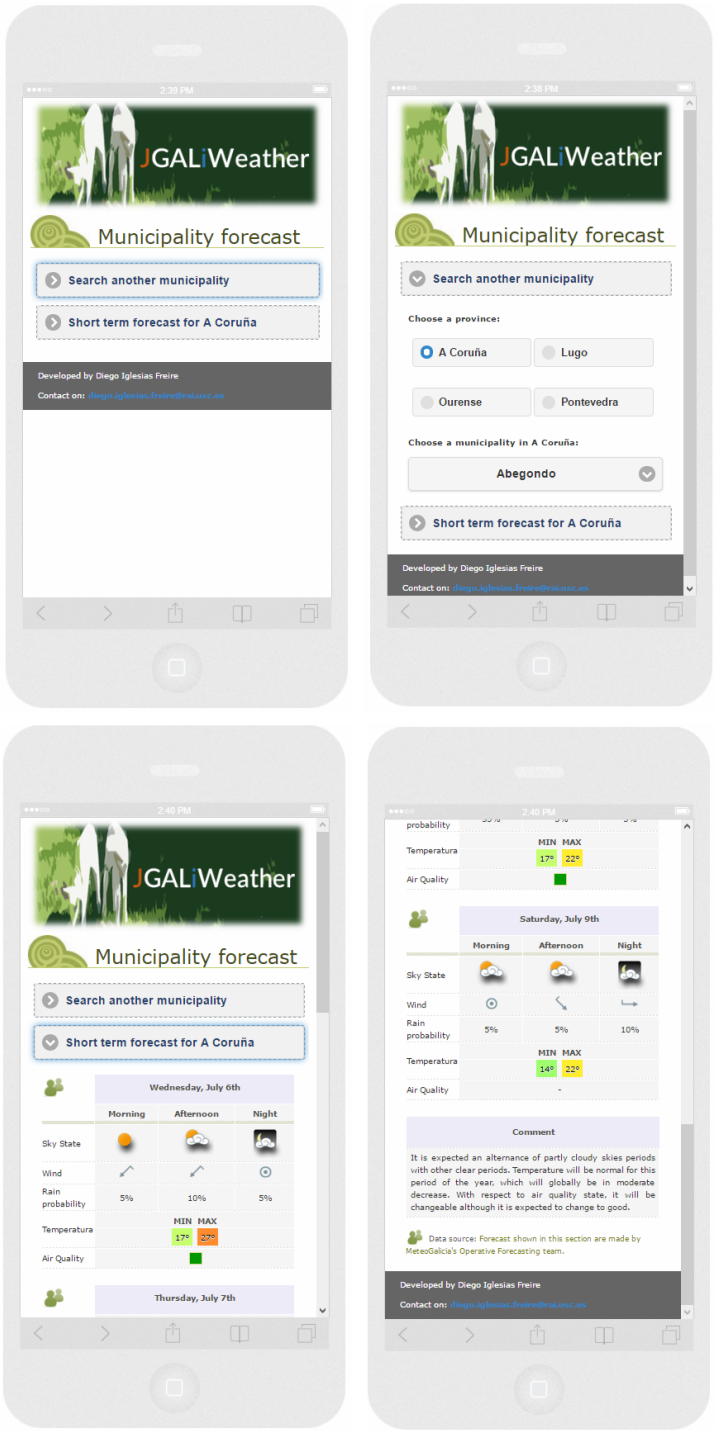


Figura 5.15: Interface *responsive* da aplicación.

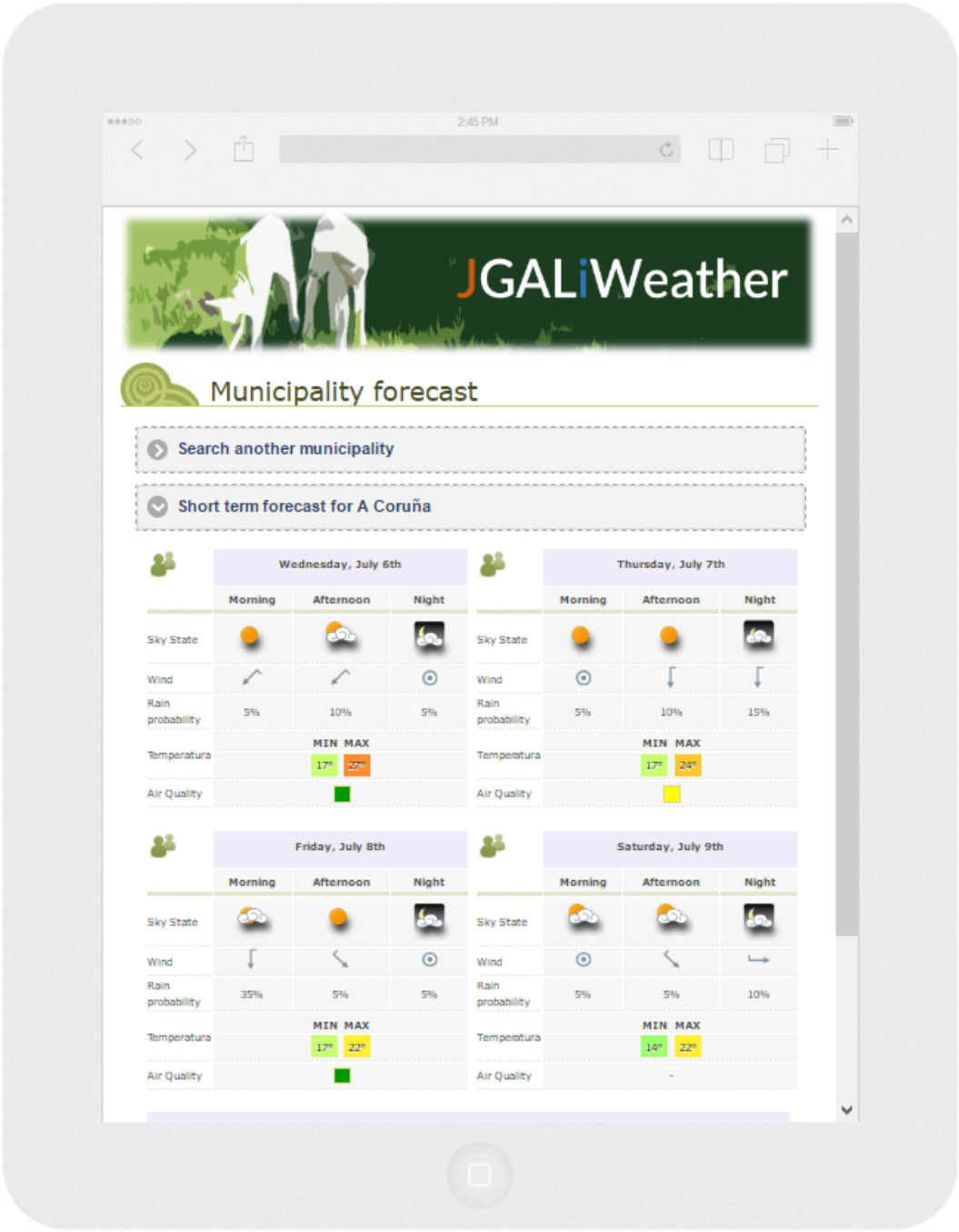


Figura 5.16: Interface *responsive* para *Tablets*.

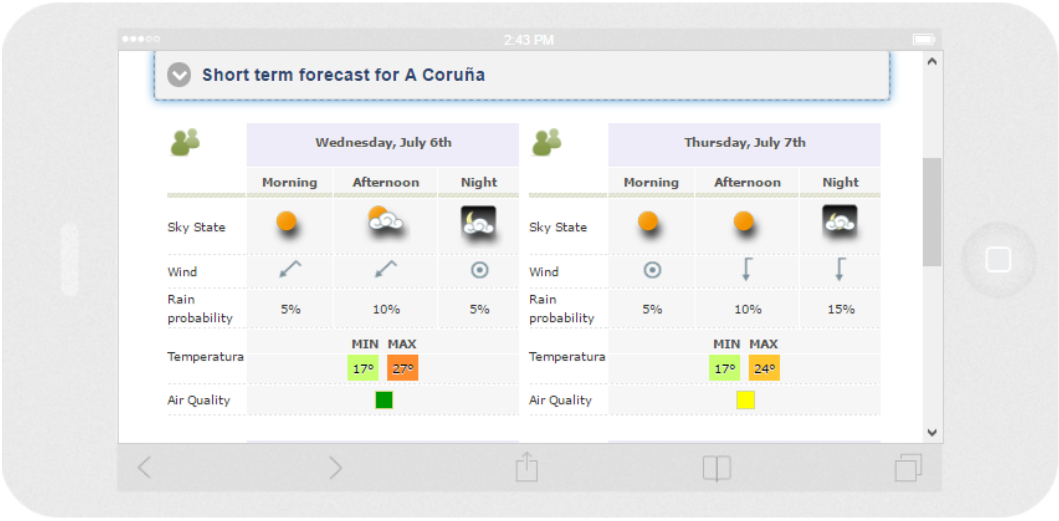


Figura 5.17: Interface *responsive* para *Smartphones* en formato apaisado.

5.3. Proveedor de datos

Debido a ocorrencia do risco **R04** nos vimos nunha situación na que careciamos de datos meteorolóxicos actualizados e o que é máis importante, non dispoñiamos de ningún dato referente ao ICA (dado que a base de datos proporcionada polo CiTIUS non tiña en conta ese tipo de datos). Por esa razón tivemos que crear unha aplicación capaz de recuperar eses datos directamente da web de MeteoGalicia.

Este *parser* é unha aplicación de consola a cal o seu único propósito e proveer á base de datos deses datos actualizados dos que antes carecía. Para elo a aplicación se divide en 3 módulos diferentes: un módulo de configuración, un de acceso a datos e o módulo encargado de analizar o código HTML.

Á hora de recuperar os datos da páxina de MeteoGalicia, en primeiro lugar o módulo de configuración recupera a información de conexión á base de datos dun ficheiro externo. A continuación se recuperan os identificadores de todos os municipios galegos da propia base de datos. MeteoGalicia proporciona os seus datos na seguinte URL:

http://www.meteogalicia.gal/web/predicion/localidades/listLocalidad.action
?idZona=**identificador**&idTipo=7&dia=-1

Esta URL prové un código HTML reducido, carente de estilos e scripts, que fai que o tempo de descarga e análise da mesma se reduza ao máximo posible.

Gracias a esta URL máis aos identificadores anteriormente recuperados, a aplicación pode *parsear* a páxina devolta dunha forma sinxela coas funcionalidades proporcionadas pola libraría **Jsoup**. Os datos recuperados son:

- Estado do ceo.
- Vento.
- Temperatura.
- Probabilidade de chuvia.
- Índice de calidade do aire.

Por último, todos estes datos engádense ou actualízanse (se xa existían datos para esa data) na base de datos para que poidan ser utilizados tanto por jGALiWeather coma pola interface web.

Capítulo 6

Probas

As probas intentan demostrar que un programa cumpre co seu propósito, así como descubrir defectos no programa antes de pasalo a produción [23]. O proceso de probas ten dúas metas distintas:

- A **validación** do software: Demostrar ao desenvolvedor de software e ao cliente que o software cumpre cos requisitos. Isto implica que exista, polo menos, unha proba de validación por cada requisito.
- A **verificación** do software: Encontrar situacións onde o comportamento do software sexa incorrecto, indesexable ou non esté de acordo coa súa especificación.

Neste capítulo, detallarase as probas realizadas para validar e verificar o software que poden ser de dous tipos:

- **Probas unitarias:** As probas unitarias son o proceso de probar compoñentes do programa, como funcións ou clases. Este tipo de probas enfocarase en comprobar a funcionalidade dos obxectos ou métodos.
- **Probas do sistema:** As probas do sistema consisten na integración dunha parte ou a totalidade dos compoñentes do sistema para probar o sistema coma un todo.

A metodoloxía Scrum utilizada neste proxecto outorga máis importancia as probas durante o desenvolvemento do software do que lle dan outras metodoloxías más clásicas. Isto é debido á restrición de entregar unha versión do software funcional ao final de cada sprint. Ao final de cada sprint se lle entrega ao cliente unha versión totalmente operativa do software co fin de obter retroalimentación da súa opinión e facer balance do cumprimento das historias de usuario (validación do software).

6.1. Verificación

Unha proba unitaria é unha forma de probar o correcto funcionamento dun compoñente dun programa. Estes compoñentes, xeralmente, son unidades encargadas de realizar unha tarefa ou funcionalidade especifica. Para verificar o software utilizaremos este tipo de probas, que neste caso probarán as distintas funcionalidades de cada un dos módulos do programa para determinas se realizan a súa tarefa correctamente devolvendo os datos requiridos.

Para implementar os test unitarios utilizamos a libraría **JUnit**. **JUnit** é un *framework* que permite realizar a execución de clases Java de maneira controlada, para poder avaliar que cada un dos métodos da clase se comporta como se espera [20]. Ademais o entorno de desenvolvemento utilizado (*NetBeans IDE*) proporciona os *plug-ins* necesarios para xerar modelos para que a creación das probas dunha clase se realice de maneira automática, reducindo o tempo de desenvolvemento de cada proba.

As probas divídense en 4 apartados (1 por cada módulo) da seguinte maneira:

- Probas do módulo de configuración: proban as funcionalidades relacionadas coa lectura dos ficheiros de configuración.
- Probas do módulo de acceso a datos: proban as funcionalidades relacionadas coa lectura dos datos almacenados na base de datos.
- Probas do módulo de operadores lingüísticos: proban as funcionalidades relacionadas coa conversión dos datos de predición en descricións lingüísticas intermedias.
- Probas do módulo NLG: proban as funcionalidades relacionadas coa xeración de textos en linguaxe natural.

Nas táboas seguintes detállanse as probas unitarias realizadas, indicando o seu identificador, unha descrición curta, a entrada proporcionada, o resultado esperado e o resultado obtido.

Probas do módulo de configuración

CP-01	
Descrición	Proba os métodos que len o ficheiro de configuración principal da aplicación.
Entrada	Ficheiro de configuración principal da aplicación.
Resultado esperado	Devolve a ruta correcta dos demais ficheiros de configuración.
Resultado obtido	Correcto

CP-02	
Descrición	Proba os métodos que len o ficheiro coas estruturas lingüísticas.
Entrada	Ficheiro de *templates* da aplicación.
Resultado esperado	Devolve a palabra **clear** ao ler o *labelset* **C** e o *label* **C**.
Resultado obtido	Correcto

CP-03	
Descrición	Proba os métodos que len o ficheiro coas variables.
Entrada	Ficheiro de *variables* da aplicación.
Resultado esperado	Devolve correctamente o rango dos valores do vento.
Resultado obtido	Correcto

CP-04	
Descrición	Proba o método **apply** da clase *ObjectSet*.
Entrada	Lista de obxectos cos valores: 1, 3, 24 e 15. Dados estos valores comprobase se o valor 24 esta entre eles.
Resultado esperado	Devolve o valor 1.
Resultado obtido	Correcto

CP-05	
Descrición	Proba os métodos que len o ficheiro coas particións.
Entrada	Ficheiro de *particións* da aplicación.
Resultado esperado	Devolve correctamente un valor de *ObjectSet*, un *CrispInterval* e un *FuzzySet*.
Resultado obtido	Correcto

Probas do módulo de operadores lingüísticos

CP-06	
Descrición	Proba que o operador lingüístico encargado da temperatura realiza correctamente a súa tarefa.
Entrada	Valores para a temperatura: 15, 16, 17, 15, 15, 16, 17 e 15.
Resultado esperado	Devolve a seguinte descrición lingüística: **VH** **WC SI** **C PV**
Resultado obtido	Correcto

CP-07	
Descrición	Proba que o operador lingüístico encargado da néboa realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o estado do ceo: 106, 115, 120, 125, 106, 105, 106, 106, 130, 106, 101 e 106.
Resultado esperado	Devolve a seguinte descrición lingüística: **[[0, 0], [2, 2], [3]]** **[[1]]** **[[3]]**
Resultado obtido	Correcto

CP-08	
Descrición	Proba que o operador lingüístico encargado do vento realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o vento: 320, 320, 320, 320, 320, 301, 332, 317, 332, 302, 317 e 332.
Resultado esperado	Devolve a seguinte descrición lingüística: **0-4 320 320 320 320 320** **6-8 332 317 332** **10-11 317 332**
Resultado obtido	Correcto

CP-09	
Descrición	Proba que o operador lingüístico encargado da chuvia realiza correctamente a súa tarefa.
Entrada	Valores numéricos para a chuvia: 110, 117, 111, 107, 118, 103, 109, 104, 113, 119, 105 e 108.
Resultado esperado	Devolve a seguinte descrición lingüística: **0-4 I I P P I** **6 SN** **8-9 ST ST** **11 P**
Resultado obtido	Correcto

CP-10	
Descrición	Proba que o operador lingüístico encargado da descrición cronolóxica do estado do ceo realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o estado do ceo: 115, 115, 115, 115, 115, 115, 115, 115, 115, 115, 115 e 115.
Resultado esperado	Devolve a seguinte descrición lingüística: **V**
Resultado obtido	Correcto

CP-11	
Descrición	Proba que o operador lingüístico encargado da descrición cuantitativa do estado do ceo realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o estado do ceo: 115, 115, 115, 115, 115, 115, 115, 115, 115, 115, 115 e 115.
Resultado esperado	Devolve a descrición lingüística axeitada.
Resultado obtido	Correcto

Probas do módulo de operadores lingüísticos

CP-12	
Descrición	Proba que o operador lingüístico encargado do vento no módulo do ICA realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o vento: 317, 320, 320, 317, 320, 320, 332, 317 e 332.
Resultado esperado	Devolve a seguinte descrición lingüística: [**3.0,6.0,3.0**]
Resultado obtido	Correcto

CP-13	
Descrición	Proba que o operador lingüístico encargado do estado do ceo no módulo ICA realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o estado do ceo: 110, 103, 111, 107, 108, 101, 109, 102 e 113.
Resultado esperado	Devolve a descrición lingüística axeitada.
Resultado obtido	Correcto

CP-14	
Descrición	Proba que o operador lingüístico encargado da chuvia no módulo ICA realiza correctamente a súa tarefa.
Entrada	Valores numéricos para a chuvia: 110, 103, 111, 107, 108, 101, 109, 102 e 113.
Resultado esperado	Devolve a seguinte descrición lingüística: [**2.0,4.0,2.0**]
Resultado obtido	Correcto

CP-15	
Descrición	Proba que o operador lingüístico encargado do ICA realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o ICA: 5, 1 e 3.
Resultado esperado	Devolve a seguinte descrición lingüística: **- + I**
Resultado obtido	Correcto

CP-16	
Descrición	Proba que o operador lingüístico encargado das condicións do ICA realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o vento: 317, 320, 320, 317, 320, 320, 332, 317 e 332. Valores numéricos para o estado do ceo: 110, 103, 111, 107, 108, 101, 109, 102 e 113. Valores numéricos para a chuvia: 110, 103, 111, 107, 108, 101, 109, 102 e 113.
Resultado esperado	Devolve a seguinte descrición lingüística: [**RW, RW, RW**]
Resultado obtido	Correcto

Probas do módulo NLG

CP-17	
Descrición	Proba que o xerador de texto en linguaxe natural encargado da descrición cuantitativa do estado do ceo realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o estado do ceo: 103, 103, 101, 101, 101, 101, 101, 101, 101, 101, 103 e 101.
Resultado esperado	Devolve o seguinte texto: **Clear skies in general for the next few days, although it will occasionally be partly cloudy.**
Resultado obtido	Correcto

CP-18	
Descrición	Proba que o xerador de texto en linguaxe natural encargado da néboa realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o estado do ceo: 104, 103, 106, 106, 101, 103, 102, 102, 102, 103, 108 e 108.
Resultado esperado	Devolve o seguinte texto: **There will be morning fog on Saturday and night fog on Friday.**
Resultado obtido	Correcto

CP-19	
Descrición	Proba que o xerador de texto en linguaxe natural encargado da temperatura realiza correctamente a súa tarefa.
Entrada	Valores numéricos para a temperatura: 15, 16, 17, 15, 15, 16, 17 e 15.
Resultado esperado	Devolve o seguinte texto: **Temperature will be very high for this period of the year, with minimums in slight increase although they will oscillate and maximums without changes.**
Resultado obtido	Correcto

CP-20	
Descrición	Proba que o xerador de texto en linguaxe natural encargado do vento realiza correctamente a súa tarefa.
Entrada	Valores numéricos para o vento: 320, 320, 320, 320, 320, 301, 332, 317, 332, 302, 317 e 332.
Resultado esperado	Devolve o seguinte texto: **Strong Southeast wind from Friday morning to Saturday afternoon, very strong Northwest from Sunday morning, changing to strong North on Sunday afternoon and to very strong Northwest on Sunday night and strong North from Monday afternoon, changing to very strong Northwest on Monday night.**
Resultado obtido	Correcto

CP-21	
Descrición	Proba que o xerador de texto en linguaxe natural encargado da chuvia realiza correctamente a súa tarefa.
Entrada	Valores numéricos para a chuvia: 104, 103, 106, 106, 101, 103, 102, 102, 102, 103, 108 e 108.
Resultado esperado	Devolve o seguinte texto: **Precipitations are expected on Monday.**
Resultado obtido	Correcto

Probas do módulo de acceso a datos

CP-22	
Descrición	Proba que o método que recupera todas as localizacións da base de datos realiza correctamente a súa tarefa.
Entrada	Datos de conexión á base de datos.
Resultado esperado	Comproba que o número total de elementos recuperados é correcto: **315**
Resultado obtido	Correcto

CP-23	
Descrición	Proba que o método que recupera os datos de predición dunha localización concreta dende base de datos realiza correctamente a súa tarefa.
Entrada	Data: 2015-07-25 Municipio: 15001
Resultado esperado	Comproba que o número total de elementos recuperados é correcto: **3**
Resultado obtido	Correcto

6.2. Validación

Como se indicou anteriormente, a validación intenta demostrar que o software desenvolvido cumpre cos requisitos especificados xunto co cliente. Para facer isto utilizamos diferentes métodos, pero entre eles o que máis destaca e a utilización de probas do sistema para validar as historias de usuario HU-01 e HU-02.

No caso da primeira historia de usuario dispoñemos de 50 casos de proba para comprobar a correcta xeración dos prognósticos textuais. Esta probas executan o sistema na súa totalidade indicando como entrada un día e unha data concreta e comparando o prognóstico textual xerado có prognóstico esperado. Os resultados esperados son os prognósticos textuais xerados pola aplicación GALiWeather pasándolle a mesma entrada nos dous programas.

Para a segunda historia de usuario o caso é similar ao anterior. Esta vez temos 25 casos proba que serven para probar que os prognósticos ICA xerados se axustan coas necesidades do cliente. Neste caso a entrada e resultados esperados son provistos por un meteorólogo profesional.

Nas táboas 6.1 e 6.2 móstranse os resultados da validación do software indicándose o identificador e o nome da historia de usuario, o resultado da validación e un pequeno comentario.

Historias de Usuario: Cliente

Ident.	Nome	Resultado	Comentario
HU-01	Xerar predicións meteorolóxicas	Feito	A aplicación pasou as 50 probas sen ningún problema.
HU-02	Xerar predicións da calidade do aire	Feito	A aplicación pasou as 25 proporcionadas polo meteorólogo.
HU-03	Xerar predicións en portugués	Non feito	Esta historia de usuario non se implementou polo seu carácter opcional e por falta de tempo.
HU-04	Xerar predicións en francés	Non feito	Esta historia de usuario non se implementou polo seu carácter opcional e por falta de tempo.
HU-05	Engadir máis parámetros	Non feito	Esta historia de usuario non se implementou polo seu carácter opcional e por falta de tempo.
HU-06	Facer predicións sen todos os parámetros	Non feito	Esta historia de usuario non se implementou polo seu carácter opcional e por falta de tempo.
HU-07	Mellorar as predicións	Feito	Unha persoa licenciada en filoloxía inglesa validou unha mostra de 25 exemplos de predición textual.
HU-08	Visualizar datos meteorolóxicos	Feito	Existe unha interface web desde onde se poden ver todos os datos.
HU-09	Interface web responsive	Feito	A interface web é capaz de adaptarse a varios tamaños de pantalla.
HU-10	Obtención do municipio por xeolocalización	Feito	O municipio que se mostra ao executar é elixido mediante xeolocalización.

Cadro 6.1: Historias de usuario do cliente.

Historias de Usuario: Desenvolvedor

Ident.	Nome	Resultado	Comentario
HU-11	Crear módulo de lectura dos ficheiros de configuración	Feito	A aplicación obtén a súa configuración de ficheiros externos.
HU-12	Crear módulo cos operadores de descricións lingüísticas	Feito	A aplicación coverte os datos de predición en descricións lingüísticas intermedias.
HU-13	Crear módulo de xeración de linguaxe natural	Feito	A aplicación coverte as descricións lingüísticas intermedias en predicións textuais.
HU-14	Crear módulo de obtención de datos	Feito	A aplicación obtén os datos de predición dunha base de datos.
HU-15	Xeración de textos en módulos independentes	Feito	As distintas funcionalidades da aplicación están implementadas en módulos independentes.
HU-16	Converter en servizo RestFul	Feito	As diferentes funcionalidades da aplicación están implementadas coma módulos independentes.
HU-17	Converter en servizo SOAP	Non feito	So tiña sentido se orientaramos a aplicación a ser unha API pública de xeración de prognósticos.
HU-18	Infomación BBDD en ficheiro externo	Feito	Os datos de conexión á base de datos encontranse no ficheiro de configuración principal.
HU-19	Información de localización en ficheiro externo	Feito	A información de localización dos demais arquivos de configuración encontrase no ficheiro de configuración principal.
HU-20	Saída de datos en JSON	Feito	A API Rest devolve os datos en formato JSON.

Cadro 6.2: Historias de usuario do desenvolvedor de software.

Capítulo 7

Conclusións

O obxectivo deste traballo foi o desenvolvemento da aplicación **jGALiWeather**, capaz de xerar predicións meteorolóxicas en linguaxe natural a partir dunha serie de datos relevantes. O traballo xustifícase no interese de crear unha aplicación que xere predicións en inglés utilizando técnicas máis sofisticadas NLG baseándose no actual servizo dispoñible para o público na sección de predición por concellos da páxina Web de MeteoGalicia.

Despois de realizar as probas de verificación e validación podemos considerar que con este proxecto se ten desenvolvido unha aplicación que cumpre cos obxectivos impostos ao principio do mesmo e, ademais engadíronse funcionalidades coas que non se contaba nos seus inicios. Creouse unha interface web para poder ver os textos xerados pola aplicación principal e que permite interaccionar con ela dun xeito case idéntico ao que experimentan os actuais usuarios de GALiWeather na web oficial de MeteoGalicia.

O deseño baseado en módulos fai que esas partes presenten un baixo acoplamento permitindo a súa reutilización por separado e proporcionando unha base sólida á hora de engadir novos datos de predición ou idiomas.

Por outro lado, a utilización do marco de traballo Scrum para o desenvolvemento do proxecto foi totalmente acertada xa que permitiu que o traballo se realizara de forma áxil cunha capacidade de resposta aos cambios e aos imprevistos moi alta. Cabe destacar que foi unha experiencia moi positiva e unha aprendizaxe constante en todas as etapas do proxecto.

Como viuse ao largo dos diferentes capítulos o proxecto desenvolveuse dunha maneira óptima, a pesar da materialización de certos riscos (coma o **R04** ou o **R05**) aos que foi fácil dar resposta. Houbo varias historias de usuario que non foron posibles de implementar debido á corta duración do proxecto. Sen embargo,

estas carecían de importancia tendo en conta os obxectivos do proxecto polo que a valoración final segue sendo positiva.

7.1. Posibles melloras

A continuación indícanse algunhas melloras coas que se podería ampliar jGALiWeather:

7.1.1. Soporte para outros idiomas

Unha das principais vantaxes de utilizar SimpleNLG para desenvolver o módulo de xeración de texto é que existen varias adaptacións da libraría que dan soporte a outros idiomas coma francés ou portugués. Estas adaptacións fan que pensar en crear novos módulos para xerar textos noutros idiomas sexa algo moi sinxelo. Ademais a isto hai que engadirlle que dispoñemos dun deseño modular que permite engadir novos módulos á aplicación de maneira relativamente sinxela. En relación ao anterior, tamén sería interesante integrar jGALiWeather no actual servizo dispoñible para o publico unha vez estivesen dispoñibles recursos similares a SimpleNLG en galego e castelán.

7.1.2. Adaptación da aplicación coma servizo web

Unha mellora interesante sería o transformar a aplicación nun servizo web que calquera podería utilizar para xerar os seus prognósticos persoais enviando os seus propios datos de entrada. Para facer isto necesitariamos modificar a maioría dos operadores lingüísticos da aplicación xa que agora mesmo están moi centrados na utilización dos datos extraídos de MeteoGalicia, os cales teñen un formato, canto menos, curioso.

7.1.3. Predicións especializadas

Outra posible mellora seria engadir módulos que, seguindo unha estratexia similar á de jGALiWeather, xerasen predicións en linguaxe natural a partir de datos numéricos dando servizos máis especializados, coma podería ser o estado do mar para mariñeiros ou surfistas, ou predicións con datos de interese para pilotos de avións comerciais. Isto precisaría partir dunha predición operativa especializada elaborada previamente.

Apéndice A

Manuais de usuario

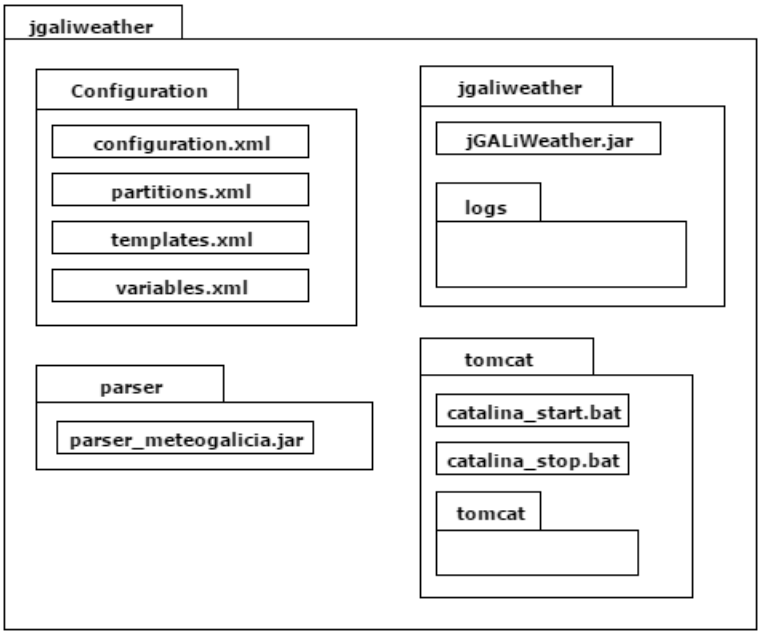
A.1. Requisitos previos

A aplicación **jGALiWeather** trae consigo todas as librarías e dependencias necesarias para a súa execución, por lo que en principio non se require ningunha libraría ou programa externo preinstalado, coa excepción da maquina virtual de Java. Os sistemas operativos compatibles abarcan desde Windows 7 hasta Windows 10 (soamente foi probada nestes sistemas, aínda que debería de ser compatible con calquera sistema operativo cunha maquina virtual de Java).

A.2. Instalación

A instalación de jGALiWeather realizase a través dun arquivo comprimido que ten no seu interior todos os executables necesarios.

O único paso necesario sería descomprimir o arquivo anterior, o que nos devolvería a seguinte estrutura de carpetas:



Na carpeta **jgaliweather** encontramonos o executable da aplicación xeradora de prognósticos en linguaxe natural xunto cunha carpeta para aloxar os ficheiros de log se non lle especificamos unha diferente.

Na carpeta **parser** encontramonos có executable da aplicación que *parsea* a web de MeteoGalicia para almacenar os datos meteorolóxicos na base de datos.

Na carpeta **tomcat** encontrase un servidor tomcat con tódolos recursos necesarios para despregar a interface web.

A carpeta **Configuration** conten tódolos ficheiros de configuración dos que dependen as aplicacións para o seu correcto funcionamento.

A base de datos esta aloxada na carpeta principal.

A.3. Configuración

Os ficheiros que se encontran no subdirectorio **Configuration** son os encargados de almacenar a información de configuración usada polas diferentes aplicacións. Este conxunto de ficheiros permiten configurar o programa en diversos

aspectos. Os arquivos *variables.xml* e *partitions.xml* conteñen información que define os datos de entrada e as operacións matemáticas da aplicación e non deben ser modificados baixo ningún concepto. Do mesmo modo, o arquivo *templates.xml* conten información necesaria para a xeración das predicións en linguaxe natural e non deben ser editados por norma xeral. O único ficheiro que pode ser editado é *configuration.xml*.

Ficheiros de configuración xeral

O ficheiro de configuración *configuration.xml* define unha serie de parámetros necesarios para a correcta execución da aplicación, que poden ser modificados para configurar as rutas de acceso ao resto de ficheiros de configuración e a conexión á base de datos.

As etiquetas <*inpath*> definen as rutas aos demais ficheiros de configuración mentres que as etiquetas <*database*> permiten definir os datos de conexión das bases de datos que fagan falta.

Por defecto, nada máis descomprimir o instalador, a aplicación é totalmente operativa polo que non se recomenda nin modificar este ficheiro nin modificar a estrutura de carpetas.

A.4. Execución

Neste apartado descríbense os detalles de execución das tres aplicacións que integran este proxecto.

jGALiWeather: Xerador de predicións

Esta aplicación admite unha serie de parámetros de entrada para a súa execución, que determinan a ubicación do ficheiro de configuración principal, o directorio de *logs*, a data da predición e se se vai xerar a predición para o ICA. Un exemplo do comando de execución de **jGALiWeather** sería:

java -jar jGALiWeather.jar -noica -date 2016-07-06 -configuration_file “ruta” -log_directory “ruta”

-noica: Bandeira, que no caso de estar presente, non se xerarían as predicións do ICA.

-date "data": Indica para que data a aplicación debe xerar as predicións. O formato utilizado é AAAA-MM-DD. Por defecto se utiliza a data actual.

-configuration_file: Determina a ruta onde se encontra o ficheiro de configuración principal.

-log_directory: Indica o directorio onde se gardaran os ficheiros de *log* (o directorio debe ser creado previamente).

-h ou *–help*: Mostra unha pequena explicación dos diferentes parámetros.

A aplicación pode executarse sen necesidade de utilizar ningún dos argumentos anteriores dado que dispón de valores por defecto para o seu correcto funcionamento nese caso.

Se todos os datos son correctos a aplicación se executará e almacenará os textos que conteñen as predicións meteorolóxicos na base de datos.

Parser MeteoGalicia

Esta aplicación é unha aplicación automática de consola cuxo único argumento (ademais do parámetro de axuda similar ao de jGALiWeather) é *–configuration_file* que indica a ruta do ficheiro de configuración principal:

java -jar parser_meteogalicia.jar -configuration_file "ruta"

No caso de omitir este parámetro a aplicación utilizaría a ruta por defecto.

Interface web

Proporcionase un servidor *Apache Tomcat* totalmente operativo que trae no seu interior a aplicación web de visualización de datos despregada. Polo tanto con só iniciar o servidor xa teremos a interface de visualización a nosa disposición.

O servidor iniciase mediante a execución do script *catalina_start.bat* aloxado na carpeta tomcat. No caso de querer deter a execución do servidor utilizariamos o script *catalina_stop.bat*.

Có servidor activo a aplicación estará dispoñible na URL:

http://localhost:8080/web_jGALiWeather/

Bibliografía

[1] A. Ramos, A. J. Bugarín, S. Barro e F. Díaz, *GALiWeather: Aplicación para la generación automática de predicciones meteorológicas en lenguaje natural*, Manual de usuario registro software (número de asiento registral 03/2014/1259), 2014.

[2] A. Ramos, A. J. Bugarín, S. Barro e J. Taboada, *Linguistic Descriptions for Automatic Generation of Textual ShortTerm Weather Forecasts on Real Prediction Data*, IEEE Transactions on fuzzy systems, 23 (1), 4457, 2015.

[3] E. Reiter and R. Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[4] E. Reiter. *An architecture for data-to-text systems*. In S. Busemann, editor, Proceedings of the 11th European Workshop on Natural Language Generation, pages 97–104, 2007.

[5] D. Woods, *Why Big Data Needs Natural Language Generation to Work* (*http://www.forbes.com/sites/danwoods/2015/07/09/why-big-data-needs-natural-language-generation-to-work*). Consultado o 6 de xullo do 2016.

[6] A. Gatt, E. Reiter. *SimpleNLG: A Realisation Engine for Practical Applications*. Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009), 9093, 2009.

[7] Project Management Institute. *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK®)*, 3ª ed., 2004. Dispoñible en *http://www.fnmt.es/documents/10179/119827/Descargar+Documentación+-+Gestión+de+Proyectos/b34b9d76-9e62-4fcb-adbd-a0e5d675b4b4*

[8] Instituto Nacional de Tecnologías de la Comunicación(INTECO). *Guía Práctica de Gestión de Configuración*. 2008.

[9] Instituto Nacional de Tecnologías de la Comunicación(INTECO). *Ingeniería del Software: Metodologías y ciclos de vida*. 2009.

[10] H. Kniberg. *Scrum and XP from the trenches*, 2ª ed., 2015. Dispoñible en *http://www.infoq.com/resource/minibooks/scrum-xp-from-the-trenches-2/en/pdf/Scrum-and-XP-from-the-Trenches-2nd-edition.pdf*

[11] CMMI® para Desarrollo. Versión 1.3, 2010. Dispoñible en *http://www.sei.cmu.edu*.

[12] Instituto de Enxeñeiros Eléctricos e Electrónicos (IEEE). *IEEE Standard 610*, 1990.

[13] B. Wake, *INVEST in Good Stories, and SMART Tasks* (*http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/*). Consultado o 25 de febreiro do 2016.

[14] Instituto de Enxeñeiros Eléctricos e Electrónicos (IEEE). *ISO/IEC/IEEE 42010*, 2013.

[15] Arquitectura de software. Artigo da wikipedia (*https://es.wikipedia.org/wiki/Arquitectura_de_software*). Consultado o 25 de xunio do 2016.

[16] ¿Qué es un Patrón de Diseño?. MSDN de Microsoft (*https://msdn.microsoft.com/es-es/library/bb972240.aspx*). Consultado o 25 de xunio del 2016.

[17] fuzzy4j - Fuzzy logic library for Java. Dispoñible en *https://github.com/sorend/fuzzy4j*. Consultado o 25 de xunio del 2016.

[18] M.I. Durán e T. Benito, Lógica Borrosa, 2009. Dispoñible en *http://www.it.uc3m.es/jvillena/irc/practicas/08-09/10.pdf*.

[19] Representational State Transfer. Artigo da wikipedia (*https://es.wikipedia.org/wiki/Representational_State_Transfer*). Consultado o 25 de xuño do 2016.

[20] JUnit. Artigo da wikipedia (*https://es.wikipedia.org/wiki/JUnit*). Consultado o 25 de xuño do 2016.

[21] Jersey - RESTful Web Services in Java. Dispoñible en *https://jersey.java.net/*. Consultado o 25 de xuño del 2016.

[22] jQuery. Dispoñible en *http://jquery.com/*. Consultado o 25 de xuño del 2016.

[23] I. Sommerville, *Ingeniería de software*, 9ª ed., 2012.

[24] Guía Salarial Sector TI Galicia 2015-2016. 2014. Dispoñible en *http://www.vitaedigital.com/download/NTY2/Guia_Salarial_Sector_TI_Galicia_2015-2016.pdf*. Consultado o 25 de xuño del 2016.