Grado en Ingeniería Informática

Trabajo Final de Grado

Desarrollo de una aplicación web para la monitorización de la predicción del tiempo

---

Autor:
José Vicente GAS VICIANO

Supervisor:
Javier MUÑOZ FERRARA

Tutor académico:
Luis Amable GARCÍA FERNÁNDEZ

Fecha de lectura: 14 de Septiembre de 2017
Curso académico 2016-2017

Resumen

Este documento contiene la memoria técnica del trabajo final del Grado en Ingeniería Informática, en el que se detalla el desarrollo de una aplicación web para la monitorización de predicciones del tiempo. El desarrollo del proyecto ha tenido lugar en la empresa ADC Infraestructuras y Sistemas SL, perteneciente al Grupo Gimeno.

La aplicación obtiene información sobre previsiones meteorológicas, avisos de emergencia meteorológica, lluvias recientes y el histórico de lluvias de distintas poblaciones de interés para el grupo de diferentes servicios web y almacena dicha información en una base de datos propia. Los usuarios pueden consultar la información disponible de forma monitorizada para cada población, así como realizar suscripciones a las predicciones de las diferentes poblaciones.

Para el desarrollo de dicha aplicación, se ha realizado un análisis de los requisitos exigidos por el cliente, así como un diseño de la base de datos que utilizará la aplicación. Finalmente se ha diseñado e implementado la aplicación y, con la finalidad de asegurar el cumplimiento de los requisitos, se han realizado pruebas para comprobar que el funcionamiento de la aplicación es el esperado.

Palabras clave

Aplicación web, monitorización, predicciones meteorológicas, Java

Keywords

Web app, monitoring, weather predictions, Java

Índice de contenido

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El objetivo de este capítulo es realizar una descripción inicial del proyecto. En primer lugar, se describe la empresa en la cual se ha desarrollado el proyecto. A continuación, se detallan las causas que han motivado la realización del proyecto. Seguidamente, se realiza una descripción detallada del proyecto y de los objetivos que se espera cumplir con su realización. Para concluir este capítulo, se realiza una explicación de la estructura de esta memoria para indicar los contenidos de cada uno de sus capítulos.

1.1 Contexto del proyecto

El proyecto que se documenta en esta memoria ha sido desarrollado en la empresa *ADC Infraestructuras y Sistemas*. Esta empresa forma parte de la división de servicios del *Grupo Gimeno*, grupo empresarial fundado en Castellón hace más de 140 años con la constitución de *FACSA*, empresa dedicada a la gestión del ciclo integral del agua.

Figura 1. Organigrama de las empresas del grupo.

Actualmente, el grupo está formado por empresas de múltiples sectores económicos, como el ciclo integral del agua, la restauración, la construcción o la

gestión medioambiental, entre otros. En la *figura 1* puede observarse el organigrama de las diferentes empresas del grupo según su sector económico.

ADC Infraestructuras y Sistemas está especializada en el desarrollo de los sistemas de telecontrol del ciclo integral del agua. ADC crea sus propias aplicaciones informáticas para la gestión de abastecimientos y el control de estaciones depuradoras.

Dentro de las áreas de actividad de la empresa se encuentra el Servicio de Informática y Comunicaciones, que se divide en tres áreas: software, *datacenter* y comunicaciones y microinformática. El proyecto que nos ocupa se incluye en el área de software de la empresa.

El área software principalmente desarrolla aplicaciones a medida para la gestión de las empresas del grupo, aunque también realiza un servicio de consultoría de modelización de datos y ofrece asesoramiento y mantenimiento en la adquisición de aplicaciones de terceros.

En esta área se incluye *IoTSens*, empresa propia de ADC que proporciona soluciones verticales basadas en el Internet de las Cosas con la finalidad de recolectar y analizar datos que conecten el mundo físico con los sistemas basados en ordenadores, con el resultado de una mejora de su eficiencia, exactitud y beneficio económico. La estancia en prácticas en la que se ha desarrollado el proyecto ha sido llevada a cabo en el equipo de desarrollo de *IoTSens*, del que el supervisor de la estancia en prácticas es responsable.

1.2 Motivación y descripción del proyecto

Como se ha comentado en el apartado 1.1 de esta memoria, el Grupo Gimeno consta de empresas pertenecientes a diversos sectores económicos. Entre ellas, FACSA y SITRA se encargan de la gestión del ciclo integral del agua.

El conocimiento lo más preciso posible de las condiciones meteorológicas resulta de especial importancia en la gestión del ciclo integral del agua, sobre todo, en actividades relativas al abastecimiento, captación, alcantarillado o gestión de aguas residuales; por lo que resulta especialmente importante disponer de predicciones meteorológicas precisas con la finalidad de lograr anticiparse a episodios de precipitaciones extremas o para optimizar los servicios relacionados.

El proyecto consiste en desarrollar una aplicación web para la monitorización de la predicción del tiempo de diversas poblaciones relacionadas con las actividades de las empresas del Grupo Gimeno. Los datos a monitorizar en la aplicación se obtienen de diferentes fuentes.

En referencia a las predicciones, la monitorización que se quiere realizar incluye la previsión meteorológica a tres días vista, los avisos meteorológicos, información sobre las lluvias recientes e información sobre las lluvias acumuladas en los meses

previos. La aplicación será de uso interno del Grupo Gimeno, y estará disponible para los trabajadores del grupo en la intranet propia del grupo.

Además de poder visualizar las predicciones meteorológicas, los usuarios podrán suscribirse a las diferentes poblaciones para las que se generarán las predicciones con el fin de recibir diariamente avisos referentes a las precipitaciones a través de mensajes de correo electrónico que generará automáticamente la aplicación.

En referencia al origen de los datos de las predicciones, la previsión meteorológica se obtiene de un servicio web de pago ofrecido por ElTiempo24, contratado por la empresa y que permite hacer un número limitado de peticiones a dicho servicio de forma mensual.

Por otra parte, los avisos meteorológicos diarios se obtienen de la página web de la Agencia Estatal de Meteorología (AEMET), mientras que la información de las lluvias recientes se obtiene del Sistema Automático de Información Hidrológica (S.A.I.H.) de la Confederación Hidrográfica del Júcar y la información del histórico de las lluvias acumuladas de un servicio interno propio del grupo que almacena los datos de sus propios pluviómetros distribuidos por las poblaciones que se quieren monitorizar, principalmente localizados en sus estaciones depuradoras de aguas residuales.

Debido a que las previsiones meteorológicas se obtienen de un servicio de pago y con el objetivo de minimizar las llamadas a dicho servició y no sobrepasar el límite contratado, resulta necesario el diseño de una base de datos donde se almacenará toda la información que posteriormente el sistema monitorizará. Una vez obtenida toda la información, esta deberá guardarse en la base de datos propia de la aplicación; de forma que cuando los usuarios accedan a la aplicación esta obtendrá los datos solicitados por los usuarios mediante consultas a la base de datos creada.

Los datos (previsiones, avisos, lluvias recientes y lluvias acumuladas) se mostrarán integrados en una misma pantalla a modo de *dashboard* y el usuario podrá, a través de un menú desplegable, elegir la población de la que desee visualizar la predicción completa.

El sistema debe permitir a los usuarios:

- Consultar la predicción meteorológica. Esta predicción incluye previsiones, avisos de emergencias meteorológicas, lluvias recientes y lluvias acumuladas.

- Navegar entre las diferentes provincias y poblaciones de interés para la empresa de las que se monitorizan predicciones para consultar la predicción de cada una de estas.

- Suscribirse a las predicciones de las poblaciones que desee, recibiendo un mensaje de correo electrónico de aviso cuando se esperen precipitaciones en las localidades a las que el usuario está suscrito.

- Cancelar las suscripciones a las predicciones de las poblaciones a las que esté suscrito, dejando de recibir la información diaria vía correo electrónico.

Además, el sistema ha de gestionar:

- La obtención y almacenamiento diario en la base de datos de las previsiones meteorológicas de las poblaciones monitorizadas a través del servicio de elTiempo24.

- La obtención periódica y almacenamiento en la base de datos de los avisos o alertas meteorológicas de las poblaciones monitorizadas a través de la página web de AEMET.

- La obtención periódica y tratamiento de la información sobre las lluvias recientes en las poblaciones monitorizadas a través del S.A.I.H. de la Confederación Hidrográfica del Júcar y su almacenamiento en la base de datos.

- La obtención de los datos referentes al histórico de precipitaciones a través de un servicio web interno a la empresa.

- El envío de correos electrónicos a los usuarios suscritos a las predicciones de las poblaciones seleccionadas por el propio usuario.

- La notificación automática al responsable del mantenimiento de la aplicación cuando no se hayan podido obtener los datos referentes a las predicciones para alguna de las poblaciones monitorizadas dentro del plazo temporal establecido.

Por otra parte, cabe remarcar que el sistema será de uso interno al Grupo Gimeno y solo accesible a través de la intranet de la empresa, quedando fuera del alcance del proyecto el registro de usuarios externos a la empresa. Además, también queda fuera del alcance del proyecto la consulta del histórico de predicciones, es decir, las predicciones generadas en fechas anteriores al día en el que se han generado.

1.3 Objetivos del proyecto

El principal objetivo del proyecto es el diseño y la implementación de una aplicación web que monitorice las predicciones meteorológicas obtenidas de distintas fuentes, con el fin de ofrecer a los empleados del grupo una información meteorológica centralizada y accesible desde la intranet del grupo para aquellas áreas de negocio que precisan de un conocimiento a corto plazo de las precipitaciones previstas.

El objetivo del proyecto puede desglosarse en los siguientes subobjetivos:

- Permitir a los usuarios, mediante una interfaz web, la visualización de información referente a predicciones meteorológicas, en concreto

previsiones, avisos, precipitaciones recientes y precipitaciones acumuladas; para diferentes poblaciones de interés para el grupo.

- Permitir a los usuarios realizar suscripciones a las diferentes poblaciones sobre las que se monitorizan las predicciones con la finalidad de recibir mensajes de correo electrónico de aviso cuando la previsión meteorológica indique que se pueden producir precipitaciones en un plazo de tres días.

1.4 Estructura de la memoria

En este apartado se explica brevemente la estructura de esta memoria, con el objetivo de facilitar a los lectores el seguimiento de la misma.

La memoria está formada por seis capítulos y dos anexos. El primer capítulo, *Introducción*, ofrece una visión general de la empresa donde se ha realizado el proyecto, así como una breve descripción del proyecto, sus motivaciones y objetivos.

En el segundo capítulo, *Planificación del proyecto*; describe la metodología de desarrollo utilizada así como la planificación temporal del proyecto y la estimación de recursos y costes asociada al proyecto. A continuación, el tercer capítulo, *Tecnologías utilizadas en el proyecto*, se describen todas las tecnologías que han sido utilizadas durante la realización del proyecto, incluyendo las herramientas y estándares que se han tenido en consideración para el diseño e implementación del sistema.

En el cuarto capítulo, *Análisis y diseño del sistema*, se documenta la fase de análisis del proyecto y se muestra el diagrama de casos de uso del proyecto, así como la descripción de requisitos y los prototipos realizados del diseño de la interfaz de la aplicación. Además, también expone el diseño de la base de datos creada para el proyecto así como la arquitectura del sistema.

El quinto capítulo, *Implementación y pruebas*, muestra la arquitectura de la aplicación y sus funcionalidades, así como el proceso de verificación y validación que permite comprobar que el proyecto cumple con los requisitos definidos durante la planificación del proyecto.

Finalmente, el sexto capítulo muestra las conclusiones obtenidas después de la realización del proyecto, además de sugerir posibles extensiones al proyecto desarrollado.

Capítulo 2

Planificación del proyecto

Este capítulo se dedica a la planificación previa del proyecto, describiendo la metodología utilizada durante el desarrollo del proyecto, así como su desglose de tareas y su seguimiento. Finalmente, se muestran las estimaciones realizadas con el objetivo de calcular a priori los recursos que serán necesarios para la realización del proyecto.

2.1 Metodología

Para el desarrollo del proyecto, y siguiendo con la metodología de desarrollo de proyectos software utilizada en la empresa, se ha seguido una metodología ágil SCRUM[1]. Esta metodología aporta flexibilidad y eficiencia cuando se trabaja en proyectos que se llevan a cabo en entornos complejos, con requisitos cambiantes o poco definidos; cosa que sucede con frecuencia.

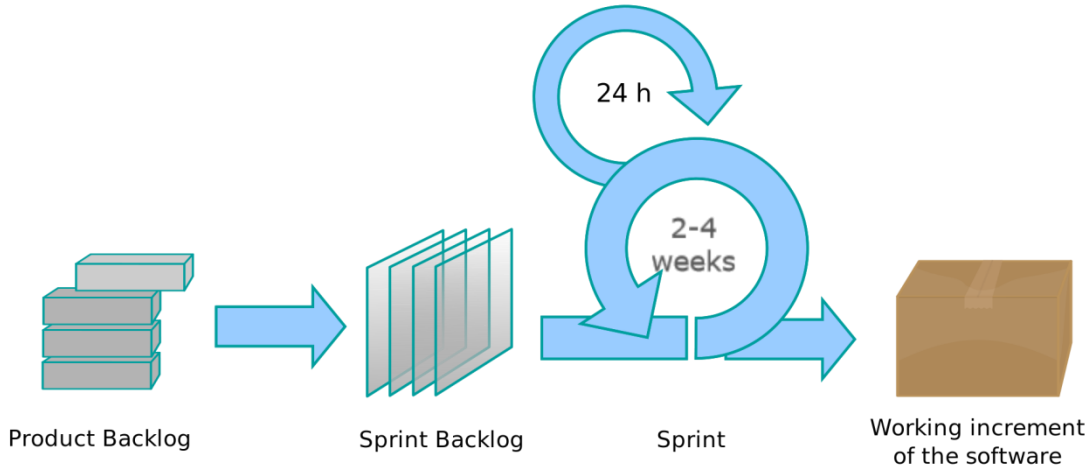

Figura 2. Ciclo de vida SCRUM

En la *figura 2* se muestra un diagrama con el ciclo de vida SCRUM. Como se puede observar en dicha figura, esta metodología está compuesta de diferentes fases, que se explican a continuación:

- El ***product backlog*** es el conjunto de tareas que deben desarrollarse para que el sistema cumpla con todos los requisitos que debe implementar. Al

[1] Fuente: *The Scrum Guide* [5].

tratarse SCRUM de una metodología ágil, estas tareas están en constante evolución y pueden modificarse a lo largo del desarrollo del proyecto.

- El ***sprint backlog*** es la pila de tareas que se van a desarrollar en el sprint actual. Para decidir cuáles de las tareas del *product backlog* se transfieren al *sprint backlog* suele asignarse un nivel de prioridad a cada tarea. Esta prioridad puede estar determinada tanto por la complejidad de las tareas como por las necesidades del usuario. La carga de trabajo de cada *sprint backlog* está planificada de forma acorde a la duración de cada sprint.

- El ***sprint*** es el periodo en el cual se lleva a cabo el desarrollo de las tareas incluidas en el *sprint backlog*. Es una fase iterativa y su duración puede variar entre dos y cuatro semanas. En el caso del proyecto desarrollado, cada sprint ha tenido una duración de tres semanas. El resultado de cada una de estas iteraciones es un producto funcional que puede ejecutarse correctamente y que tiene un valor añadido respecto al producto obtenido en la iteración anterior.

Por otra parte, el equipo de desarrollo de SCRUM está formado por los roles que se describen a continuación:

- El ***Product Owner*** decide la funcionalidad del producto, es el encargado de priorizar las tareas y decidir las tareas que se incluyen en cada *sprint backlog*.

- El ***Scrum Manager*** es el responsable de motivar y coordinar al equipo y de detectar los problemas que puedan surgir. Es el encargado de comprobar que las reglas de SCRUM se cumplen.

- El ***Team Scrum*** es el equipo de trabajo multidisciplinar encargado de desarrollar el producto

Finalmente, cabe añadir que de forma diaria en cada sprint, el equipo de trabajo se reúne en las conocidas como *Daily Scrum*, dónde los miembros del equipo ponen en común su trabajo durante el último día. En ellas, cada participante responde a tres preguntas:

- ¿Qué he hecho desde la última *Daily Scrum*?
- ¿Qué tengo planeado hacer hoy?
- ¿Qué impedimentos he tenido o voy a tener?

Estas reuniones facilitan la colaboración entre los miembros del grupo, permitiendo que se comparta información del trabajo que cada miembro del equipo ha realizado y facilitando la resolución de problemas de coordinación al conocer cada miembro del grupo el trabajo realizado por los demás.

Una vez detallada la metodología utilizada durante el transcurso de la estancia en prácticas en la que se ha llevado a cabo el desarrollo del proyecto, la *tabla 1* desglosa los diferentes *sprints* de los que ha constado el proyecto, así como las tareas que se han realizado en cada uno de ellos.

Sprint	Fecha de inicio	Fecha de fin	Tareas realizadas
Sprint 1	12-12-2016	30-12-2016	Mostrar predicción meteorológica diaria de una población. Mostrar avisos meteorológicos de una población. Mostrar lluvias recientes de una población.
Sprint 2	2-1-2017	20-1-2017	Mostrar lluvias acumuladas de una población. Seleccionar datos por provincia y población. Actualizar información de precipitaciones de SAIH para cada población.
Sprint 3	23-1-2017	10-2-2017	Actualizar avisos AEMET para cada población. Actualizar predicciones de ElTiempo24h para cada población.
Sprint 4	13-2-2017	3-3-2017	Lectura del histórico de precipitaciones de servicio SiccaEdar. Gestionar usuarios de la intranet en la aplicación. Mostrar las suscripciones a predicciones de poblaciones.
Sprint 5	6-3-2017	31-3-2017	Gestionar las suscripciones a poblaciones. Generar *iframe* de resumen de la predicción por población. Envío por email de las predicciones a usuarios suscritos.

Tabla 1. Detalle de los *sprints* realizados durante el desarrollo del proyecto.

2.2 Planificación

En el apartado 2.1 se explica que se ha seguido una metodología ágil SCRUM para el desarrollo del proyecto. No obstante, para facilitar la documentación de la planificación y el análisis del proyecto, tanto en este apartado como en los siguientes del capítulo 2 (*Planificación del proyecto*) y 4 (*Análisis del sistema*) se han realizado de acuerdo con las metodologías propias de un proceso de ingeniería clásico, incluyendo las fases de análisis de requisitos, diseño e implementación, y verificación y validación.

En la fase de análisis de requisitos se estudian las necesidades del cliente y qué espera este que el sistema ofrezca. Los resultados obtenidos tras esta fase son un diagrama de casos de uso, el catálogo de requisitos del sistema y un conjunto de prototipos sencillos. A continuación, en la fase de diseño e implementación se realiza el diseño de la arquitectura. Además, se explican los patrones de diseño y estándares de desarrollo utilizados tanto en la empresa donde se ha realizado la estancia en prácticas como en el proyecto realizado. Finalmente, a partir de los resultados obtenidos en la fase de análisis, se lleva a cabo la implementación del sistema.

Una vez implementado el sistema, en la fase de verificación y validación se prueba el sistema en diferentes escenarios con el fin de corroborar su correcto funcionamiento mediante pruebas unitarias, donde se prueba el correcto funcionamiento de la lógica interna de cada componente; y pruebas de integración, donde se comprueba la correcta coordinación entre los diferentes módulos de la aplicación.

Finalmente, en la fase de implantación se incluyen las tareas necesarias para realizar un despliegue de la aplicación en un entorno de producción. No obstante, en el proyecto no se incluye la fase de implantación.

2.3.1 Descomposición del trabajo

Una vez indicadas las fases planificadas del proyecto, el siguiente paso consiste en realizar un desglose de todas las tareas que se incluyen en el proyecto. A las tareas de implementación propiamente dichas se han añadido las tareas de formación y de redacción de la propuesta técnica del proyecto, ya que también han formado parte del proyecto, así como las tareas relativas al análisis del sistema.

En la *figura 3* se definen todas las tareas que forman parte del proyecto mediante un esquema de descomposición de trabajo, conocido como *EDT*.

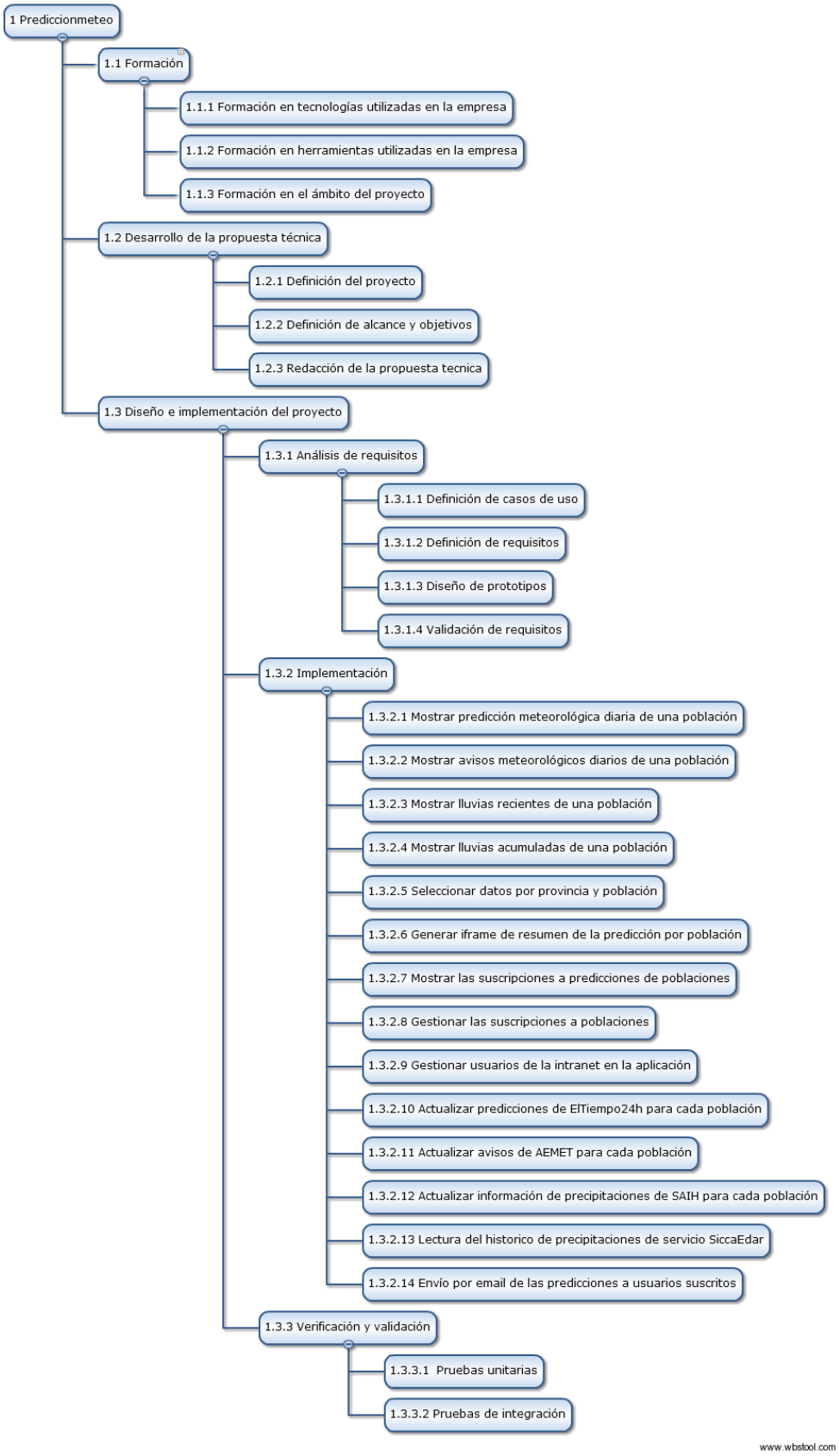

Figura 3. Diagrama EDT del proyecto

2.2.2 Planificación temporal

En este apartado se asigna un coste temporal a las tareas desglosadas en el diagrama EDT del proyecto. Para realizar una correcta asignación del tiempo del proyecto a las tareas es necesario considerar que, por motivos curriculares del Grado en Ingeniería Informática, el tiempo total del proyecto debe ser de 300 horas. Estas se han distribuido desde el mes de diciembre de 2016 hasta el mes de marzo de 2017. En la tabla 2 se muestra el detalle de las horas realizadas distribuidas por semanas.

Diciembre 2016	
Semana del 5-12-2016	15 horas
Semana del 12-12-2016	25 horas
Semana del 19-12-2016	25 horas
Semana del 26-12-2016	20 horas
Total horas diciembre:	85 horas
Enero 2017	
Semana del 2-1-2017:	20 horas
Semana del 9-1-2017:	20 horas
Semana del 16-1-2017:	20 horas
Semana del 23-1-2017:	20 horas
Semana del 30-1-2017:	15 horas
Total horas enero:	95 horas
Febrero 2017	
Semana 6-2-2017:	18 horas
Semana 13-2-2017:	18 horas
Semana 20-2-2017:	18 horas
Semana 27-2-2017:	14 horas
Total horas febrero:	68 horas
Marzo 2017	
Semana del 6-3-2017:	15 horas
Semana del 13-3-2017:	18 horas
Semana del 20-3-2017:	12 horas
Semana del 27-3-2017:	15 horas
Total horas marzo:	60 horas
Total horas del proyecto:	300 horas

Tabla 2. Horas presenciales semanales del proyecto

A continuación, debe distribuirse el tiempo total del proyecto entre las diferentes tareas, teniendo siempre en cuenta la complejidad de cada una de ellas. La *tabla 3* muestra todas las tareas incluidas en el esquema de descomposición de trabajo con su coste temporal asociado, indicando las dependencias entre tareas.

Nº	Tareas	Tiempo (h)	Dependencias
1	**Predicción Meteorológica 24h**	**300**	
1.1	**Formación**	**11**	
1.1.1	Formación en tecnologías utilizadas en la empresa	5	
1.1.2	Formación en herramientas utilizadas en la empresa	5	1.1.1
1.1.3	Formación en el ámbito del proyecto	1	1.2.1
1.2	**Desarrollo de la propuesta técnica**	**7**	
1.2.1	Definición del proyecto	2	1.1.3
1.2.2	Definición de alcance y objetivos	2	1.2.1
1.2.3	Redacción de la propuesta técnica	3	1.2.2
1.3	**Diseño e implementación del proyecto**	**282**	
1.3.1	**Análisis de requisitos**	**11**	
1.3.1.1	Definición de casos de uso	3	1.2.3
1.3.1.2	Definición de requisitos	3	1.3.1.1
1.3.1.3	Diseño de prototipos	4	1.3.1.2
1.3.1.4	Validación de requisitos	1	1.3.1.3
1.3.2	**Implementación**	**265**	
1.3.2.1	Mostrar predicción meteorológica diaria de una población	25	1.3.1.4
1.3.2.2	Mostrar avisos meteorológicos diarios de una población	20	1.3.2.1
1.3.2.3	Mostrar lluvias recientes de una población	20	1.3.2.2
1.3.2.4	Mostrar lluvias acumuladas de una población	10	1.3.2.3
1.3.2.5	Seleccionar datos por provincia y población	20	1.3.2.4
1.3.2.6	Generar *iframe* de resumen de la predicción por población	10	1.3.2.5
2.2.1.7	Mostrar las suscripciones a predicciones de poblaciones	22	1.3.2.6

1.3.2.8	Gestionar las suscripciones a poblaciones	25	1.3.2.7
1.3.2.9	Gestionar usuarios de la intranet en la aplicación	10	1.3.2.8
1.3.2.10	Actualizar predicciones de ElTiempo24h para cada población	30	1.3.2.9
1.3.2.11	Actualizar avisos AEMET para cada población	25	1.3.2.10
1.3.2.12	Actualizar información de precipitaciones de SAIH para cada población	20	1.3.2.11
1.3.2.13	Lectura del histórico de precipitaciones de servicio SiccaEdar	8	1.3.2.12
1.3.2.14	Envío por email de las predicciones a usuarios suscritos	20	1.3.2.13
1.3.3	**Verificación y validación**	**6**	
1.3.3.1	Test unitarios	4	1.3.2.14
1.3.3.2	Test de integración	2	1.3.3.1

Tabla 3. Definición de tareas y costes temporales del proyecto.

Por otra parte, también resulta necesario justificar las 150 horas que faltan para llegar a las 450 horas correspondientes a los 18 créditos de la estancia en prácticas y proyecto final de grado. Estas horas son no presenciales y, como se observa en la *tabla 4*, se han dedicado mayoritariamente a la redacción de la memoria.

	Tareas no presenciales	Tiempo (horas)	
1.4	**Redacción y presentación del Trabajo Final de Grado**	**150**	
1.4.1	Redacción de informes quincenales	20	
1.4.2	Redacción de la memoria	119	1.4.1
1.4.3	Entrega de la memoria	0	1.4.2
1.4.4	Preparación de la presentación	10	1.4.3
1.4.5	Presentación del Trabajo Final de Grado	1	1.4.4

Tabla 4. Definición de tareas no presenciales y su tiempo asignado.

Una vez asignado el coste temporal de cada tarea, en todo proyecto de desarrollo de software surge la necesidad de detectar las tareas que pueden provocar retrasos con respecto a la planificación del proyecto. Para llevar a cabo esto, se ha creado

un diagrama de Gantt, que se muestra en la *figura 4*. No obstante, aunque cuando se trata de proyectos llevados a cabo por un equipo de trabajo este diagrama es útil para hallar el camino crítico, que incluye todas las tareas cuyo retraso provoca el retraso global del proyecto; en el caso de nuestro proyecto, realizado por una única persona, al no poderse realizar dos tareas a la vez, la planificación de las tareas es completamente secuencial, como se puede observar en el diagrama de Gantt.

Nombre de tarea	Duración	Comienzo	Fin	Pred
Predicción Meteorológica 24h	**302 horas**	**lun 05/12/16**	**vie 31/03/17**	
Formación	**11 horas**	**lun 05/12/16**	**vie 09/12/16**	
Formación en tecnologías utilizadas en la empresa	5 horas	lun 05/12/16	lun 05/12/16	
Formación en herramientas utilizadas en la empresa	5 horas	mié 07/12/16	mié 07/12/16	3
Formación en el ámbito del proyecto	1 hora	vie 09/12/16	vie 09/12/16	4
Desarrollo de la propuesta técnica	**7 horas**	**vie 09/12/16**	**lun 12/12/16**	**5**
Definición del proyecto	2 horas	vie 09/12/16	vie 09/12/16	5
Definición de alcance y objetivos	2 horas	vie 09/12/16	vie 09/12/16	7
Redacción de la propuesta técnica	3 horas	lun 12/12/16	lun 12/12/16	8
Diseño e implementación del proyecto	**284 horas**	**lun 12/12/16**	**vie 31/03/17**	**9**
Análisis de requisitos	**11 horas**	**lun 12/12/16**	**mié 14/12/16**	**9**
Definición de casos de uso	3 horas	lun 12/12/16	mar 13/12/16	9
Definición de requisitos	3 horas	mar 13/12/16	mar 13/12/16	12
Diseño de prototipos	4 horas	mar 13/12/16	mié 14/12/16	13
Validación de requisitos	1 hora	mié 14/12/16	mié 14/12/16	14
Implementación	**265 horas**	**mié 14/12/16**	**mar 28/03/17**	
Mostrar predicción meteorológica diaria de una población	25 horas	mié 14/12/16	mié 21/12/16	15
Mostrar avisos meteorológicos diarios de una población	20 horas	mié 21/12/16	mié 28/12/16	17
Mostrar lluvias recientes de una población	20 horas	mié 28/12/16	mar 03/01/17	18
Mostrar lluvias acumuladas de una población	10 horas	mar 03/01/17	jue 05/01/17	19
Seleccionar datos por provincia y población	20 horas	jue 05/01/17	jue 12/01/17	20
Generar iframe de resumen de la predicción por población	10 horas	jue 12/01/17	lun 16/01/17	21
Mostrar las suscripciones a predicciones de poblaciones	22 horas	mar 17/01/17	mar 24/01/17	22
Gestionar las suscripciones a poblaciones	25 horas	mar 24/01/17	jue 02/02/17	23
Gestionar usuarios de la intranet en la aplicación	10 horas	vie 03/02/17	mar 07/02/17	24
Actualizar predicciones de ElTiempo24h para cada población	30 horas	mar 07/02/17	mar 21/02/17	25
Actualizar avisos AEMET para cada población	25 horas	mar 21/02/17	lun 06/03/17	26
Actualizar información de precipitaciones de SAIH para cada población	20 horas	lun 06/03/17	mar 14/03/17	27
Lectura del histórico de precipitaciones de servicio SiccaEdar	8 horas	mar 14/03/17	vie 17/03/17	28
Envío por email de las predicciones a usuarios suscritos	20 horas	vie 17/03/17	mar 28/03/17	29
Verificación y validación	**8 horas**	**jue 30/03/17**	**vie 31/03/17**	**30**
Tests unitarios	4 horas	jue 30/03/17	jue 30/03/17	30
Test de integración	2 horas	vie 31/03/17	vie 31/03/17	32

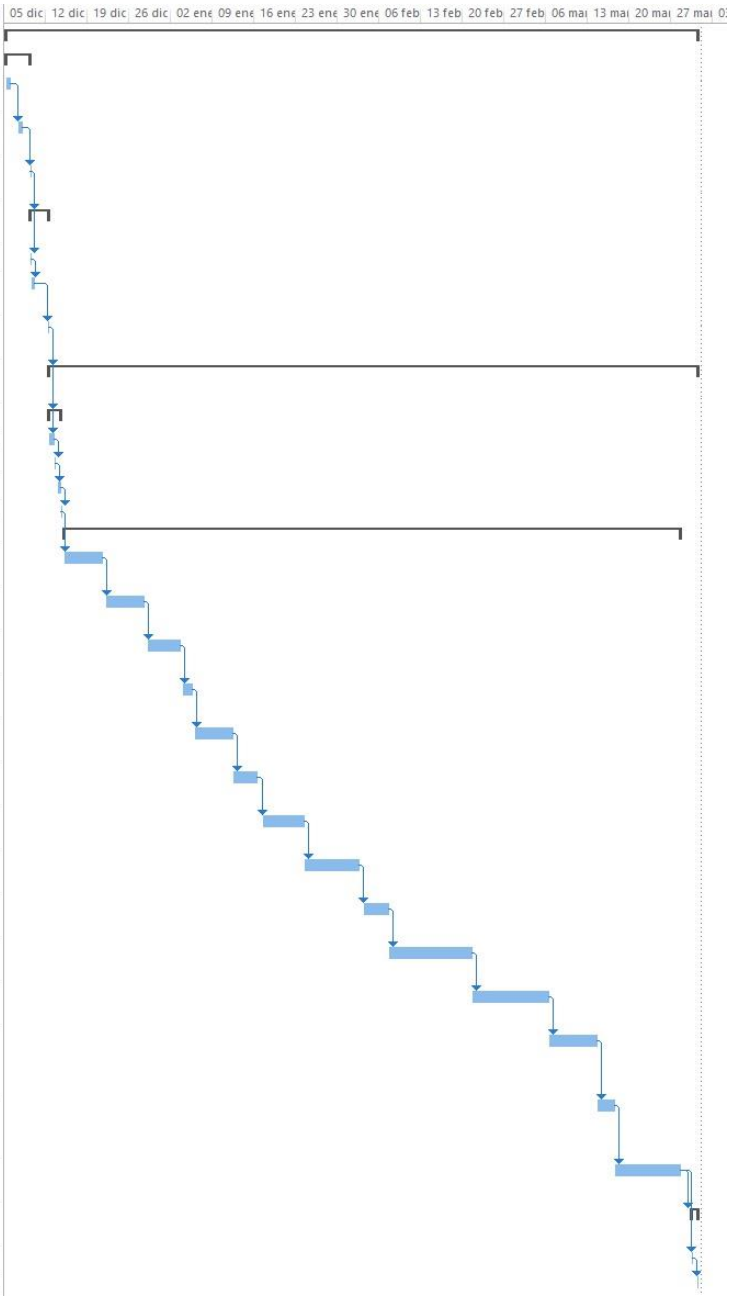

Figura 4. Diagrama de Gantt del proyecto.

Como ya se ha comentado anteriormente en el apartado 2.2, las tareas que se exponen en esta planificación son las típicas de un desarrollo con un ciclo de vida clásico, aunque el proyecto se ha desarrollado mediante una metodología ágil.

2.3 Estimación de recursos y costes del proyecto

El objetivo de este apartado es calcular los costes asociados al proyecto como si el estudiante fuese un trabajador y la empresa calculara el coste del proyecto como si fuera desarrollado por sus trabajadores en nómina.

2.3.1 Modelo de estimación LDC

El modelo de estimación LDC se encarga de estimar mediante el número total de líneas de código el tamaño del producto desarrollado. Para cada módulo del sistema, la fórmula mostrada en la *figura 5* se utiliza para calcular la cantidad esperada de líneas de código, donde O es el valor optimista, MP el valor más probable y P el valor pesimista.

$$E = \frac{O + (4 \times MP) + P}{6}$$

Figura 5. Fórmula para el cálculo de la cantidad esperada de LDC.

A continuación se muestra una descomposición en paquetes de trabajo del proyecto, de forma que las tareas especificadas en el apartado 3.2.1 se encuentran agregadas en los siguientes paquetes:

- Mostrar datos de predicciones: Incluye desde la tarea 1.3.2.1 hasta la 1.3.2.6.
- Adquirir y almacenar datos de predicciones: Está compuesto por las tareas 1.3.2.10 hasta la 1.3.2.13.
- Gestión de suscripciones: desde la tarea 1.3.2.7 hasta la 1.3.2.9.
- Envío de correos electrónicos automáticos: Representa la tarea 1.3.2.14.

Seguidamente, debe aplicarse la fórmula indicada en la figura 5 para cada uno de los paquetes anteriormente descritos. Para ello debe tenerse muy en cuenta el lenguaje de programación utilizado, puesto que el número de líneas de código puede diferir bastante al utilizar distintos lenguajes. En nuestro caso, el desarrollo *frontend* se ha realizado utilizando principalmente *TypeScript*, mientras que el desarrollo *backend* se ha implementado en Java.

El siguiente paso es obtener, a partir de datos históricos de otros proyectos similares; los valores optimista, pesimista y más probable para realizar la estimación, mostrándose dichos valores en la *tabla 5*, además del valor esperado obtenido tras realizar los cálculos pertinentes.

Finalmente, la estimación en líneas de código del proyecto es la suma de la estimación de cada uno de sus módulos, que en el caso del proyecto que nos ocupa es de 3559 LDC.

Módulo	Optimista	Más Probable	Pesimista	Esperada
Mostrar datos	900	1100	1400	1117
Adquirir datos	1200	1500	1700	1483
Gestión suscripciones	600	750	900	750
Envío correos	150	200	300	209
			Total:	3559 LDC

Tabla 5. Cálculo de la estimación de líneas de código del proyecto.

2.3.2 Estimación de costes con COCOMO

Una vez obtenida la estimación de líneas de código del proyecto, el siguiente paso es calcular el coste económico que conlleva el proyecto. Para ello se ha utilizado el modelo COCOMO[2], que permite aproximar el coste del proyecto a partir de fórmulas matemáticas en función de la estimación de LDC mostrada en el apartado 2.3.1. Para realizar los cálculos pertinentes, en primer lugar debe pasare la estimación de LDC a miles de líneas de código, obteniendo un total de 3,559 KLDC.

Seguidamente, deben seleccionarse las variables numéricas a utilizar en el modelo COCOMO, escogiendo qué tipo de proyecto se está realizando. Como nuestro proyecto no alcanza las 50 KLDC, consideramos que es de tipo orgánico. La *tabla 6* muestra el valor de las variables según el tipo de proyecto del modelo COCOMO.

Proyecto	a	b	c	d
Orgánico	2,4	1,05	2,5	0,38
Semiacoplado	3	1,12	2,5	0,35
Empotrado	3,6	1,20	2,5	0,32

Tabla 6. Valor de las variables del modelo COCOMO para cada tipo de proyecto.

[2] Fuente: *Estimating Software Based on Use Case Points* [6].

A continuación, ya pueden realizarse los cálculos del modelo:

Esfuerzo por persona y mes (E):

$$E = a \times KLDC^{b} = 2{,}4 \times 3{,}559^{1{,}05} = 9{,}101$$

Tiempo de desarrollo en meses (D):

$$D = c \times E^{d} = 2{,}5 * 9{,}101^{0{,}38} = 5{,}78\, meses$$

De este modo, hemos obtenido la duración total del proyecto en meses si un trabajador estuviera dedicado al desarrollo del mismo a jornada completa. No obstante, nuestro proyecto está limitado a cuatro meses, que ha sido la duración de la estancia en prácticas.

Para ajustar el tiempo de desarrollo a la duración real que debe tener, es necesario asignar más recursos humanos al proyecto, de forma que la cantidad de recursos necesarios para realizar el proyecto viene dado por el cociente entre el esfuerzo por persona y mes (E) y el tiempo de desarrollo del proyecto en meses (D), que toma el valor del número de meses en el que se debe realizar el proyecto.

$$CH = \frac{E}{D} = \frac{9{,}101}{4} = 2{,}275$$

De esto se desprende, que serían necesarias 2,275 personas para la realización del proyecto en cuatro meses. Esto es un valor estimado basado en datos históricos de otros proyectos, por lo que podemos simplificar que el equipo de trabajo estará formado por dos miembros, que en este caso serán un analista y un programador junior.

Una vez conocidos los perfiles que formarán parte del equipo de trabajo, resulta sencillo, conociendo los sueldos de los perfiles analista y programador junior, calcular el coste del proyecto. El detalle de este coste se muestra en la *tabla 7*.

Perfil	Salario bruto mensual	Salario total (4 meses)
Analista	2417 €	9668 €
Programador Junior	1833 €	7332 €
	Total bruto	17000 €
	Total (+35%)	22950 €

Tabla 7. Coste económico del proyecto.

Como se puede observar, se ha añadido un 35% extra de coste sobre el total bruto. Este porcentaje se corresponde a impuestos[3] y a gastos generales para la realización del proyecto[4].

A modo de conclusión de la estimación de costes realizada, hay que tener presente en todo momento que se trata de una estimación previa a la realización del proyecto y basada en proyectos anteriores con la finalidad de poder realizar un presupuesto ajustado al proyecto antes de su inicio para que este pueda ser aceptado o rechazado por el cliente.

2.4 Seguimiento del proyecto

Para realizar un correcto seguimiento del proyecto y poder solucionar de forma rápida y eficiente los inconvenientes que puedan surgir, se ha realizado un seguimiento del proyecto junto con el supervisor, teniendo en consideración la metodología ágil utilizada para el desarrollo del mismo. Así, tras la finalización de cada sprint, se ha contrastado el avance del proyecto respecto a la planificación inicial.

Para concluir este apartado, van a exponerse brevemente las desviaciones que han ocurrido durante el desarrollo del proyecto. En primer lugar, tras la implementación de las tareas de visualización de datos, se modificó la prioridad de las tareas por parte del supervisor, por lo que el orden en el que se realizaron las tareas fue modificado, realizándose las tareas referentes a la obtención de los datos antes que las tareas referentes a las suscripciones.

Además, en la tarea 1.3.2.1, mostrar predicción meteorológica diaria de una población, tuvo una duración mayor de la planificada a causa de que el cliente propuso el análisis de la API REST gratuita de AEMET para comprobar si se podía obtener de ella la misma información sobre las previsiones en lugar de obtenerlas del servicio de pago ElTiempo24, con el fin de ahorrar costes. Una vez realizado dicho análisis, se dio el visto bueno al cambio de fuente de datos. No obstante, una vez implementado el algoritmo de obtención de los datos, el cliente cambió de opinión respecto a la fuente de datos a utilizar, por lo que la tarea tuvo que ser implementada de nuevo para obtener los datos del servicio de pago de ElTiempo24.

No obstante, en los casos mencionados anteriormente, el uso de una metodología ágil ha provocado que el doble cambio de requisitos haya sido fácil de gestionar.

[3] Un 30% corresponde a impuestos

[4] Mantenimiento de equipos, alquiler de oficinas, luz y licencias software.

Capítulo 3

Herramientas utilizadas en el proyecto

El objetivo de este capítulo es describir detalladamente las tecnologías utilizadas en el proyecto, tanto en la gestión como en la implementación de este. Un rápido aprendizaje en el uso de estas herramientas resulta de especial importancia a la hora de desarrollar el proyecto, puesto que el hecho de no tener los conocimientos suficientes de ellas puede producir importantes retrasos en el desarrollo del proyecto.

3.1 Herramientas de gestión

3.1.1 JIRA

JIRA es una aplicación web, propiedad de la empresa *Atlassian*, dedicada a la gestión de proyectos ágiles. Entre sus principales funcionalidades, permite la creación y gestión de tableros *SCRUM* y *Kanban*. En el caso del proyecto que nos ocupa, se han utilizado tableros *SCRUM*.

Los tableros ofrecen al *Scrum Manager* la posibilidad de gestionar tanto el *product* backlog como el *sprint backlog* de los proyectos, además de permitir la creación de los diferentes *sprints* que se producen durante el desarrollo del proyecto, lo que facilita el seguimiento de las tareas del proyecto por parte de los miembros del equipo de trabajo. En este tablero se incluyen las tareas del *sprint backlog* de cada iteración del proceso de desarrollo, que se pueden desplazar por los cuatro estados que muestra el tablero, los cuales se detallan a continuación:

- Pendiente: en este estado se incluyen todas las tareas del *sprint backlog* que se encuentran a la espera de ser implementadas por parte del miembro del equipo asignado a cada una de ellas.

- En progreso: incluye todas aquellas tareas en proceso de desarrollo. En teoría, cada miembro del equipo de trabajo solo debería tener una tarea en desarrollo. No obstante, en ocasiones esto no es posible, ya que puede darse el caso de que alguna tarea en progreso quede bloqueada a la espera de la finalización de otra tarea perteneciente a otro miembro del equipo.

- En pruebas: en este estado se incluyen aquellas tareas que ya han sido completamente desarrolladas y testeadas, pero que todavía deben recibir el visto bueno del *Scrum Manager*. En caso contrario, el *Scrum Manager* cambia el estado de la tarea a pendiente, además de proporcionar al desarrollador de la tarea el *feedback* necesario para mejorar el código implementado.

- **Finalizado**: incluye todas las tareas a las que el *Scrum Manager* ha dado el visto bueno, dándose éstas por completadas.

3.2 Herramientas de documentación

3.1.2. Confluence

Al igual que *JIRA*, *Confluence* es otra plataforma web propiedad de *Atlassian*. Se trata de una herramienta de colaboración en equipo[5]. *Confluence* se utiliza en la empresa para documentar gran parte del conocimiento interno de la empresa, como pueden ser estándares y patrones de diseño utilizados, librerías propias o las configuraciones necesarias para los entornos de desarrollo utilizados.

Una de las principales características que aporta valor al uso de *Confluence* es su integración con *JIRA*. Esto permite que se puedan enlazar contenidos de la wiki en tareas del *JIRA* que tenga que tener presente la información enlazada, facilitando el análisis previo de las tareas.

3.3 Herramientas de desarrollo

3.3.1. GitLab

GitLab es un servicio de control de versiones basado en *Git*[6]. En la empresa se utiliza como gestor de repositorios de código, además de permitir la creación de *merge requests* como forma de solicitar al *Scrum Manager* que valide el código de una tarea ya desarrollada.

3.3.2. IntelliJ IDEA

IntelliJ IDEA es el entorno de desarrollo integrado, *IDE* por sus siglas en inglés, utilizado por los desarrolladores de la empresa. La gran ventaja que proporciona este *IDE* específico para Java es su gran integración con otras herramientas de desarrollo, como los servicios de control de versiones, fácilmente gestionables desde la propia interfaz de *IntelliJ IDEA*.

[5] Estas herramientas colaborativas se conocen por el nombre de *wiki*.
[6] *Git* es un software de control de versiones distribuido creado por Linus Torvalds y distribuido bajo licencia *Open Source*

3.3.3. MySQL y PhpMyAdmin

Como se ha indicado en la descripción del proyecto incluida en el apartado 1.2, toda la información recopilada referente a las predicciones meteorológicas debe ser almacenada en una base de datos propia. Por ello, surge la necesidad de utilizar un Sistema de Gestión de Bases de Datos.

El SGBD que se ha utilizado en el proyecto es *MySQL* y para el acceso al mismo se ha utilizado *phpMyAdmin* cómo interfaz de administración. *PhpMyAdmin* proporciona una interfaz web de administración sencilla que permite, entre otras funcionalidades, una sencilla ejecución de scripts o gran facilidad para exportar e importar tablas y datos.

Para la conexión entre el proyecto Java y la base de datos se ha utilizado *Hibernate JPA 2.0*, en el cual, mediante el uso de anotaciones en las clases del modelo de datos, se realiza un mapeo de los objetos creados en la aplicación sobre tablas de la base de datos. Para la consulta de datos, se ha utilizado *QueryDSL*, un framework de Java que permite la realización de consultas *SQL* de forma segura.

3.3.4 Spring Boot y Apache Maven

Spring Boot[7] es *un framework* de desarrollo de aplicaciones Java. Este surge como una versión simplificada de *Spring Framework*, para simplificar la creación de aplicaciones. Se integra con *Apache Maven*, una herramienta para la construcción de proyectos y de gestión de dependencias en Java. Utilizando *Maven* podemos indicar qué librerías necesita el proyecto y cómo se tiene que compilar, empaquetar, ejecutar o distribuir.

La configuración de un proyecto Java utilizando *Maven* se realiza en un fichero *XML* denominado *pom.xml*[8]. *Maven* lee este archivo de configuración y busca las librerías necesarias para compilar y empaquetar el proyecto; tanto las que directamente hemos indicado en el archivo *pom.xml* como las dependencias transitivas.

Además, *Spring Boot* también facilita el despliegue del proyecto desde fases tempranas, al incluir un contenedor web con soporte para *JSPs* y *servlets* con *Tomcat* embebido.

[7] Fuente: *Spring Data: Modern Data Access for Enterprise Java* [3].
[8] *POM* es el acrónimo de *Project Object Model*.

3.4 Herramientas del cliente web

En esta sección van a detallarse las diferentes tecnologías y lenguajes de programación utilizados para la implementación del cliente web de la aplicación.

3.4.1 Angular2

Angular 2 es la evolución de *AngularJS*, un *framework* de código abierto de *JavaScript*. *Angular2*, por lo tanto, es un *framework* de aplicaciones web que facilita la implementación de aplicaciones de navegador con capacidad de implementar el patrón de diseño Modelo-Vista-Controlador. Utiliza *TypeScript* como lenguaje de programación.

3.4.2 HTML5

HTML, acrónimo de *Hyper Text Markup Language*, es el estándar web utilizado mayoritariamente en el mundo. Estandarizado por el *World Wide Web Consortium* (*W3C*), es un lenguaje de etiquetas que mediante el uso de estas indica al navegador encargado de renderizar la página web información sobre su contenido.

En el proyecto ha sido utilizado para implementar las diferentes secciones y elementos contenidos que podrán visualizarse en el navegador web.

3.4.3 CSS3

Una vez se han definido los elementos, es fundamental darles un formato visual adecuado, de forma que todos estos elementos se visualicen de una forma organizada, clara y agradable para el usuario. De esto se encarga, precisamente, *CSS* (*Cascade Style Sheets*).

CSS ha sido el lenguaje utilizado para dar el aspecto visual deseado al código *HTML* del proyecto. Además, *CSS3* permite poder adaptar la visualización del contenido de la página web dependiendo de la resolución de los diferentes dispositivos que accedan a ella, permitiendo una correcta visualización independientemente del dispositivo a través del que se consulte.

3.4.4 Typescript

TypeScript es un lenguaje de programación de código abierto. Está desarrollado por Microsoft y es un superconjunto de *JavaScript*, por lo que añade nuevas funcionalidades al lenguaje *JavaScript*. Estas nuevas funcionalidades incluyen el uso estático de tipos y el uso de objetos construidos a partir de clases.

Una vez escrito código *TypeScript*, su compilador lo traduce a código *JavaScript*, que posteriormente podrá ser ejecutado en cualquier navegador web y dotará de dinamismo a la aplicación web.

3.4.5 PrimeFaces

PrimeFaces es una colección de componentes *open source* de interfaz de usuario (UI). Su versión *Prime NG* está desarrollada específicamente para Angular 2. Entre su amplia gama de componentes, ofrece diferentes componentes de *inputs*, como *checkboxes* o *dropdowns*; así como diferentes paneles, *datatables* y menús. Todos estos componentes están diseñados para que sean *responsive*, es decir, que adapten su apariencia dependiendo del dispositivo en el que se visualicen.

En el proyecto se han utilizado diferentes *inputs* así como su *grid*, con la finalidad de distribuir los diferentes componentes por la interfaz de usuario.

3.5 Estándares

En este apartado se muestran los estándares tecnológicos utilizados durante el proceso de desarrollo del proyecto.

3.5.1 REST

REST[9] es un estilo arquitectónico para sistemas *software* distribuidos basado en el concepto de recurso[10]. Un recurso es cualquier concepto con significado en la aplicación que puede ser nombrado (predicción, suscripción, ciudad, provincia, etc.) y, por lo tanto, representado. Sobre un recurso pueden realizarse una serie acotada de acciones, que son crear (operación *POST*), eliminar (operación *DELETE*), modificar (operación *PUT*) y leer (operación *GET*). También es importante destacar que los servicios *REST* no mantienen un estado entre peticiones.

De esta forma, *REST* estandariza la comunicación entre cliente y servidor mediante las operaciones sobre los recursos, de forma que lo que se comunica es el estado de cada recurso accesible mediante su representación. Habitualmente, el intercambio de datos entre cliente y servidor se realiza codificando los recursos en formato *JSON*, aunque también pueden utilizarse otros formatos como *XML*.

JSON, acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON define los datos en listas o mapas de objetos con

[9] Acrónimo de REpresentational State Transfer

[10] Fuente: wiki del Grupo Gimeno [4].

estructura clave/valor, donde la clave suele ser el nombre de una propiedad del objeto.

En el proyecto desarrollado, la comunicación entre el cliente y el servidor se ha realizado mediante el desarrollo de servicios *REST* utilizando el estándar *JAX-RS*, usando su implementación Jersey. Esta implementación proporciona una serie de anotaciones en Java que permiten definir cuáles son los recursos y subrecursos de nuestra aplicación y los métodos que deben responder a las acciones del protocolo.

3.5.2 XML

XML, *eXtensible Markup Language*, es un metalenguaje que permite definir lenguajes de marcas para representar información. Está desarrollado por el W3C y una de las características más importantes que tiene es su gran flexibilidad a la hora de representar información.

Para poder realizar una correcta lectura de los datos de la previsión meteorológica, obtenidos en formato *xml*, resulta importante conocer las características del propio documento con el fin de identificar de forma precisa la información que interese obtener para su almacenamiento en la base de datos de la aplicación.

3.5.3 Test Driven Development

El desarrollo guiado por pruebas, *Test Driven Development* por sus siglas en inglés, es una práctica en ingeniería de *software* en la que, al realizar una tarea, se prioriza la implementación de las pruebas que deberá superar el software a la implementación del propio *software*.

Este tipo de desarrollo es habitual en proyectos desarrollados con metodologías ágiles, ya que al realizar las pruebas basándose en los requisitos que debe cumplir la tarea, el hecho de que el código supere dichas pruebas significa que se cumplen los requisitos.

Capítulo 4

Análisis del sistema

En este capítulo se definen formalmente los requisitos del sistema y se exponen las características que debe implementar. Estos requisitos deben acordarse con el cliente y son fundamentales para obtener un sistema de calidad.

En las siguientes secciones se muestran las diferentes fases del análisis, además del diseño de la arquitectura del sistema y de la base de datos que utilizará la aplicación a desarrollar. Finalmente, se mostrarán los *wireframes* de la interfaz de usuario de la aplicación.

4.1 Historias de usuario

Las historias de usuario son uno de los pilares del desarrollo ágil de proyectos software. Una historia de usuario es una definición a alto nivel de un requisito software desde el punto de vista del usuario y, aunque sean menos específicas que los casos de uso, aportan la información necesaria para poder estimar el esfuerzo necesario para implementar el requisito.

La *tabla 8* contiene las historias de usuario del proyecto, junto a su código asociado:

Código	Historia de usuario
HU01	Como usuario quiero poder ver la información de la previsión meteorológica de una población.
HU02	Como usuario quiero poder ver la información de los avisos meteorológicos de una población.
HU03	Como usuario quiero poder ver la información de las lluvias recientes de una población.
HU04	Como usuario quiero poder ver la información de las lluvias acumuladas en los últimos meses de una población.
HU05	Como usuario quiero poder seleccionar la población de la que quiero consultar la predicción meteorológica.
HU06	Como usuario quiero poder suscribirme a las predicciones de lluvia de una población.
HU07	Como usuario quiero poder consultar a que poblaciones estoy suscrito.
HU08	Como usuario quiero poder cancelar suscripciones a las predicciones de lluvia de una población.
HU09	Como usuario quiero recibir una notificación vía e-mail cuándo estén previstas lluvias en una de las poblaciones a las que estoy suscrito.

Tabla 8. Definición de historias de usuario

4.2 Diagrama de casos de uso

En el siguiente apartado se muestra el diagrama de casos de uso del sistema. Este diagrama es fundamental en la fase de análisis del sistema para definir de forma visual la relación entre los diferentes actores que interactúan con el sistema y los diferentes casos de uso. El diagrama de casos de uso del proyecto se expone en la *figura 6*.

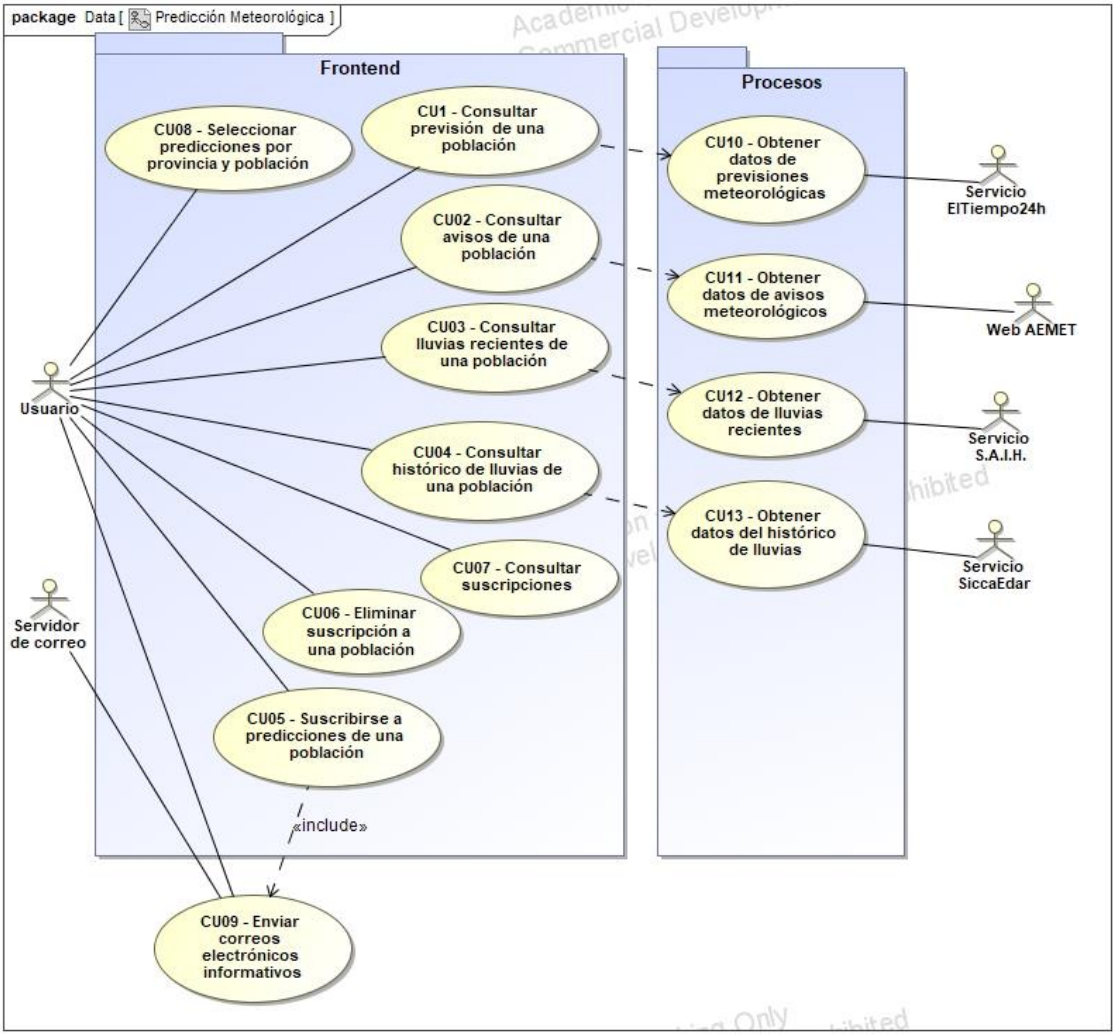

Figura 6. Diagrama de casos de uso del proyecto

4.3 Requisitos funcionales

En este apartado se muestran los requisitos funcionales del sistema, cuyo objetivo es definir el alcance del sistema. Para ello, deben definirse a nivel funcional las características que debe satisfacer el sistema. La definición de estos requisitos es fundamental para conseguir comprender el problema a resolver. En el *anexo A* se muestran las tablas donde se definen estos requisitos.

4.4 Requisitos de datos

Con la finalidad de satisfacer los requisitos funcionales expuestos en el apartado 4.3, se deben definir un conjunto de entidades y atributos necesarios para que los requisitos puedan cumplirse. En el *anexo B* se muestran las diferentes entidades junto con sus atributos y propiedades.

4.5. Diagrama de clases

Una vez obtenidos los requisitos de datos, las entidades obtenidas y las relaciones entre ellas se muestran en el diagrama de clases del proyecto de la *figura 7*.

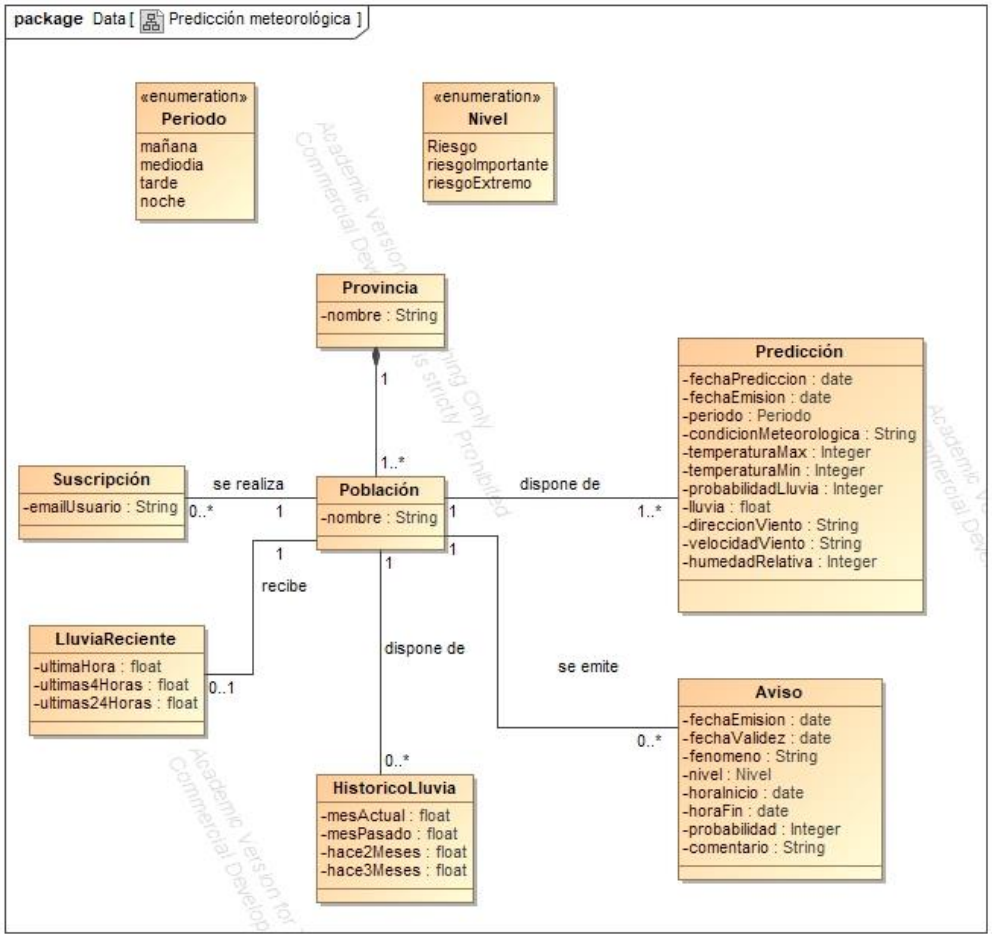

Figura 7. Diagrama de clases del proyecto

El diagrama de clases es un diagrama de estructura de UML2[11] que sirve para mostrar la estructura de la información del sistema así como su comportamiento a través de las operaciones de sus clases. El diagrama de clases se obtiene una vez

[11] UML2 (*Unified Modeling Language*) es un lenguaje de modelado de sistemas software, muy utilizado en el análisis de sistemas.

definidos los requisitos de datos del sistema y define como se organiza la información relativa al sistema.

4.6 Prototipos

En esta sección se muestran los prototipos de la interfaz de usuario realizados en la fase de análisis. Estos prototipos son valiosos para proporcionar una visión gráfica o funcional del sistema informático a desarrollar y pueden resultar de gran utilidad para mostrar al cliente la funcionalidad que implementará el sistema. En el caso de nuestro proyecto, la fase de prototipado ha consistido en la creación de *wireframes*, que son bocetos que representan la estructura visual del sitio web que representa. Los *wireframes* que a continuación se exponen han sido realizados utilizando la herramienta *Balsamiq Mockups*[12].

En primer lugar, en la *figura 8* se muestra el *wireframe* del *dashboard* de predicción meteorológica.

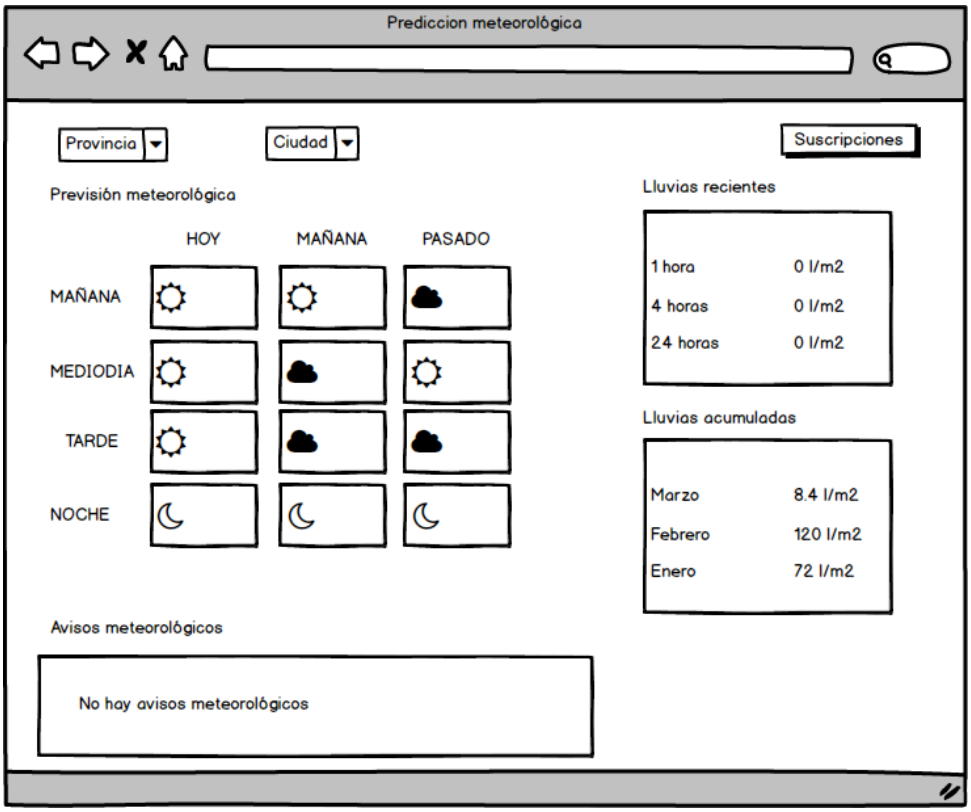

Figura 8. *Wireframe* del *dashboard* de la aplicación

Como puede observarse, la intención es mostrar toda la información integrada en una misma pantalla. Así, en la parte superior izquierda de la interfaz se encuentran

[12] *Balsamiq Mockups* es una herramienta de prototipado que permite una sencilla creación de *wireframes*, disponible en su página web *www.balsamiq.com*.

los selectores de las previsiones por provincia y ciudad, mientras que en la parte superior derecha se encuentra el enlace a la gestión de suscripciones.

En la parte central de la interfaz se sitúa la previsión meteorológica en forma de tabla. En ella, se incluirá la previsión obtenida para cada franja horaria[13] del día en curso y los dos siguientes días.

A la derecha de la previsión, se encuentran los componentes que muestran la información relativa a lluvias recientes y el histórico de lluvias acumuladas en los últimos meses. Finalmente, en la parte inferior de la interfaz se ubica el componente con los avisos meteorológicos.

En el caso de no poder obtener la información a mostrar de los diferentes servicios, en el lugar de los datos de los que no disponemos aparecerá un mensaje informando de que no se dispone de dicha información. Esta circunstancia se observa en la tabla de previsiones meteorológicas que se muestra en la *figura 9*.

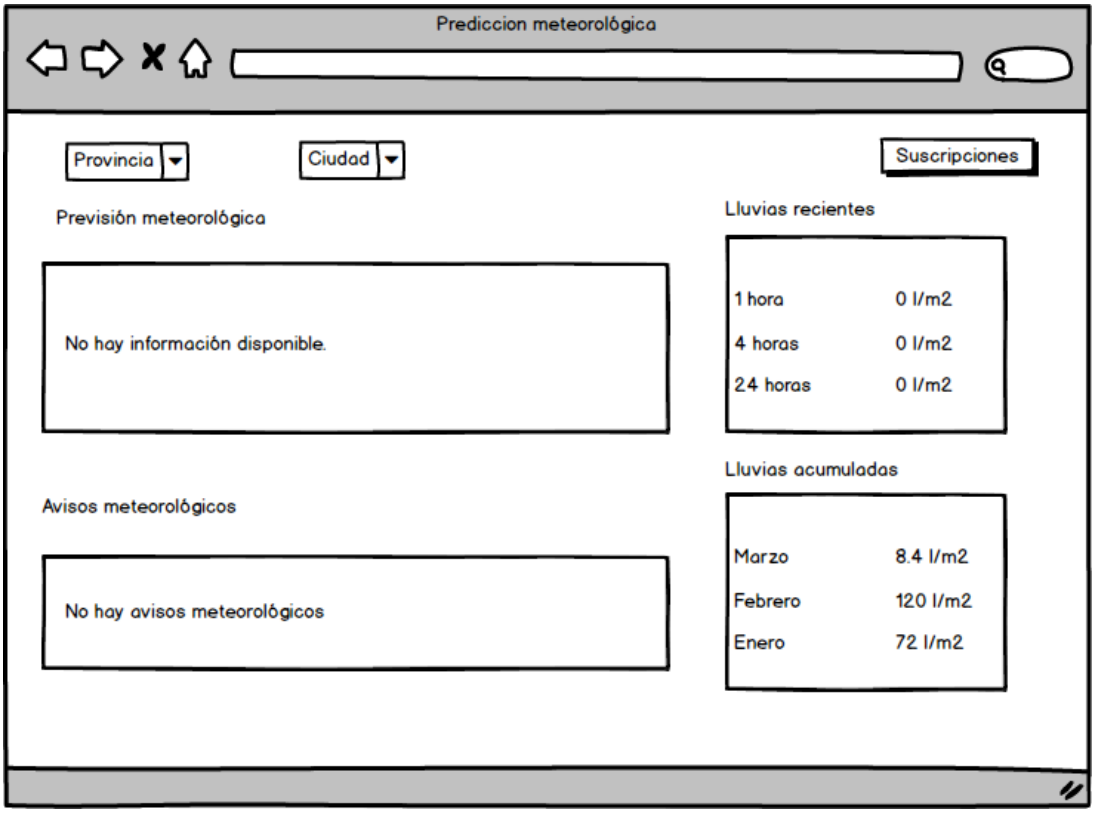

Figura 9. *Wireframe* de *dashboard* sin información disponible.

Además del *dashboard* principal de la aplicación encargado de monitorizar las predicciones obtenidas, también es necesaria la realización de un *wireframe* del *iframe* que se añadirá a la página principal de la intranet del Grupo Gimeno como enlace de entrada a la aplicación. Como se observa en la *figura 10*, el *iframe*

[13] Las franjas horarias utilizadas en la aplicación son mañana, mediodía, tarde y noche.

mostrará la ciudad y la previsión correspondiente a la franja horaria en la que se visualice.

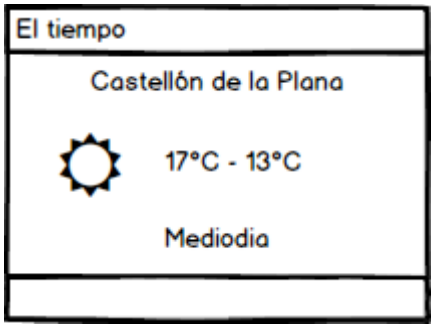

Figura 10. *Wireframe* de un *iframe* resumen de la previsión

Finalmente, en la *figura 11* se muestra la pantalla de gestión de suscripciones, donde mediante un *checkbox* para cada población de las que se obtienen predicciones. Estos checkboxes estarán distribuidos por provincias. Además, habrá un botón en una posición visible de la interfaz que servirá para volver al *dashboard* de predicciones.

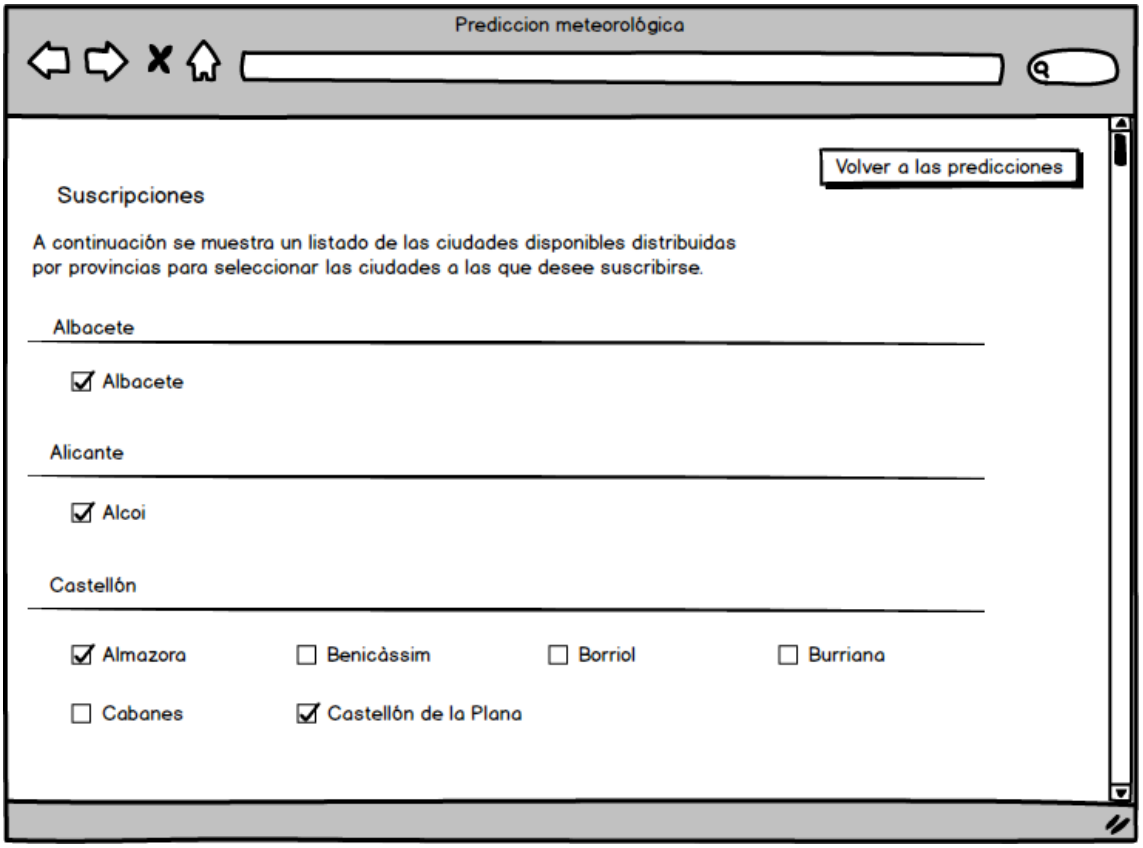

Figura 11. *Wireframe* de la gestión de suscripciones.

4.7 Diseño de la arquitectura del sistema

4.7.1 Arquitectura del sistema

En este apartado se define la arquitectura del sistema a alto nivel. En el caso del proyecto, se ha implementado una arquitectura cliente/servidor, formada por un cliente, que realizará peticiones *HTTP* para obtener datos sobre predicciones o suscripciones; y un servidor, que suministrará dichos datos mediante una API REST. La *figura 12* muestra un esquema de la arquitectura cliente/servidor.

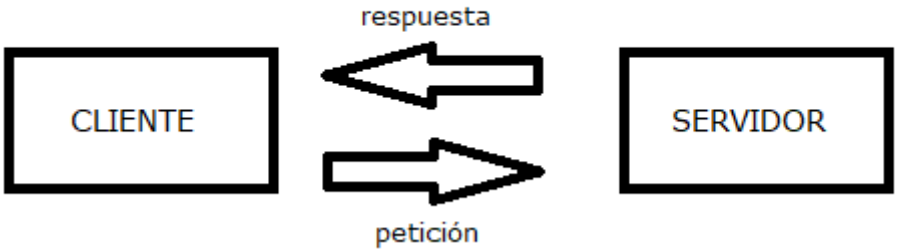

Figura 12. Arquitectura cliente/servidor.

De esta forma, el cliente realizará peticiones a los diferentes recursos REST del servidor. Dicho servidor, consultará su base de datos para generar la respuesta, que será enviada al cliente con los datos que ha solicitado. Además, el servidor también se encargará periódicamente de la adquisición de los datos de diferentes servicios web, las previsiones del tiempo de un servicio de ElTiempo24, los avisos meteorológicos de la página web de AEMET, las lluvias recientes de documentos Excel proporcionados por el S.A.I.H de la Confederación hidrográfica del Júcar y el histórico de precipitaciones de un servicio web propio del grupo con datos recopilados de sus estaciones depuradoras de aguas residuales (EDAR).

4.7.2 Diseño de la base de datos

Como se ha indicado en la sección 3.3.3, se ha utilizado una base de datos para persistir la información referente a las predicciones. En este apartado se muestra el modelo entidad-relación que representa el diseño de la base de datos, incluyendo atributos en cada una de las tablas e indicando relaciones entre las tablas y su cardinalidad.

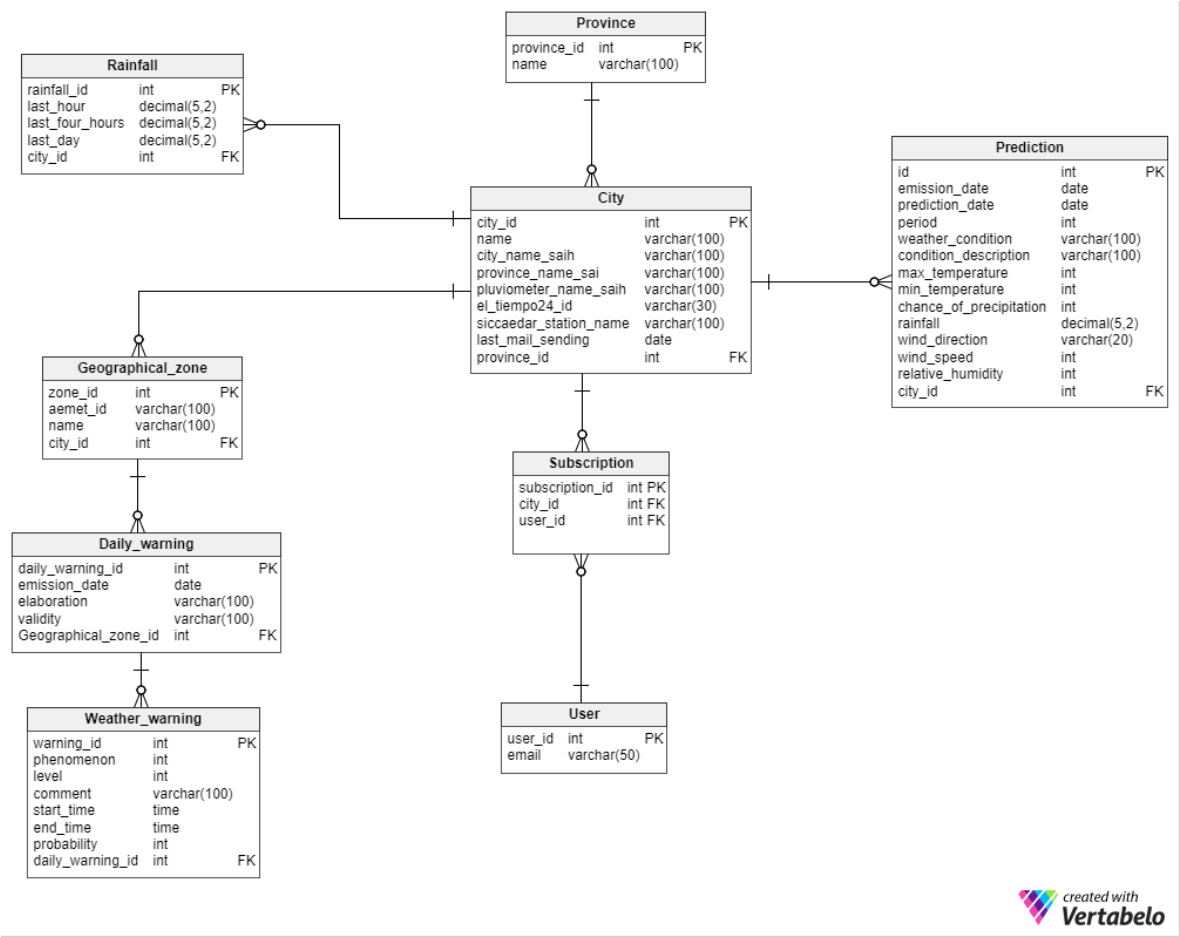

Figura 13. Modelo E/R de la base de datos de la aplicación

Como puede verse en el diagrama de la *figura 13*, se han añadido claves primarias autogeneradas a todas las tablas, además de añadir tablas como *daily_warning* y *geographical_zone* para facilitar el almacenamiento y la lectura de los avisos meteorológicos, ya que en la página web de AEMET estos se asocian a una zona geográfica en lugar de a una población.

Además, se han añadido algunos atributos ajenos al propio sistema. Los valores de estos atributos serán utilizados en el servidor para la obtención periódica de toda la información relativa a las predicciones de las diferentes fuentes de información.

También es importante tener en cuenta que, al utilizar *Hibernate JPA* para el mapeo objeto-relacional, el diseño de esta base de datos se ha tomado en gran consideración en el momento de la implementación en Java del modelo de datos de la aplicación, ya que es en este modelo de datos donde, mediante anotaciones de Java, se realiza dicho mapeo.

Capítulo 5

Implementación y pruebas

En este capítulo, tras haber realizado el análisis del sistema a desarrollar, se expone el proceso de implementación de la aplicación.

En primer lugar, se ofrecen detalles acerca de la implementación del proyecto, el entorno en el que se ha desarrollado, la arquitectura de la aplicación y la interfaz web. A continuación, se muestra la verificación y validación realizada durante el proyecto para asegurar una aplicación de calidad que se adapte a los requisitos definidos durante la fase de análisis del sistema.

5.1 Detalles de implementación

5.1.1 Entorno de desarrollo

Para el desarrollo de la aplicación se ha utilizado el entorno de desarrollo que habitualmente se utiliza en la empresa donde se ha realizado la estancia en prácticas, trabajando de forma remota sobre una máquina virtual con sistema operativo Ubuntu 14. Esta máquina virtual ya incluye las tecnologías mencionadas en el capítulo 2 de la memoria, de forma que no es necesario que el programador instale y configure dichas tecnologías. De esta forma, la máquina virtual tiene instalado el IDE *IntelliJ IDEA* enlazado con el gestor de repositorios *Git* del grupo.

5.1.2 Arquitectura de la aplicación

Como se ha introducido en el apartado 4.7.1, se ha seguido una arquitectura cliente/servidor para el desarrollo del proyecto. En esta sección se detallan todas las capas de la arquitectura del proyecto. Es necesario comentar que la arquitectura de la aplicación ha venido marcada desde un primer momento, al ser la arquitectura utilizada por los desarrolladores software Java de la empresa.

Como se observa en la *figura 14*, las capas que forman parte de la arquitectura de la aplicación son las siguientes:

- Cliente web: en el cliente se han utilizado las tecnologías explicadas en el apartado 3.4. de la memoria. Angular2 provee a la aplicación de los componentes necesarios para realizar peticiones a los servicios web REST de la aplicación. El cliente recibe como respuesta el conjunto de datos que se mostrarán en la interfaz web.

Figura 14. Arquitectura de la aplicación.

- **Servicios web / REST**: como se ha indicado en el apartado 3.1.5, los servicios web de la aplicación definen una serie de recursos sobre los que los usuarios pueden realizar las acciones GET, POST, PUT y DELETE.

- **Lógica de negocio**: los objetos de negocio son POJOs (*Plain Old Java Objects*), es decir, objetos Java sencillos independientes de cualquier *framework* o tecnología utilizada en otras capas de la arquitectura de la aplicación. Para la implementación de estos objetos de negocio, en la empresa se siguen una serie de principios documentados en su wiki, como el de responsabilidad única, priorizar composición sobre herencia, la inyección de dependencias, la implementación de métodos cortos y con nombres descriptivos, y el uso adecuado de excepciones.

- **DAO**: La capa de persistencia de la aplicación se ha implementado siguiendo el patrón de diseño DAO (Data Access Object). En este patrón se implementan unas clases DAO, cuyas instancias proporcionan métodos para añadir, recuperar, modificar y eliminar objetos del sistema de persistencia de la aplicación.

5.1.2 Patrones de diseño

En esta sección se detallan los dos patrones de diseño fundamentales utilizados durante el desarrollo: el patrón DAO y el patrón Modelo-Vista-Controlador.

5.1.2.1 Patrón DAO

La capa DAO[14] de una aplicación se encarga de gestionar la persistencia de los objetos. Esta capa está compuesta por una serie de objetos DAO que almacenan o recuperan objetos desde el sistema de persistencia. Los objetos DAO definen una interfaz que trata únicamente con objetos de negocio por lo que se consigue independizar el resto de la aplicación del mecanismo de persistencia utilizado; sea una base de datos relacional, archivos XML o cualquier otro tipo.

Los métodos que proporciona un objeto DAO para gestionar la persistencia dependen de las necesidades específicas de cada aplicación. Es habitual tener métodos para recuperar un objeto consultando por su identificador o para almacenar una versión actualizada de todas las propiedades de un objeto. Para evitar sobrecargar con un tráfico innecesario a la base de datos, también se pueden implementar métodos para leer objetos que sólo tengan inicializadas las propiedades que se van a utilizar en un momento dado.

5.1.2.2 Patrón Modelo-Vista-Controlador

El patrón de diseño Modelo-Vista-Controlador (MVC)[15] es un patrón arquitectónico que ayuda a desacoplar las clases de la aplicación al implementar interfaces gráficas de usuario. Su aportación consiste en agrupar las clases según tres roles:

- Modelo: son las clases responsables de gestionar los datos de la aplicación.
- Vista: son los componentes que proporcionan la interfaz gráfica de usuario.
- Controlador: son los componentes responsables de la gestión de la lógica de la aplicación. Recibe las acciones realizadas por el modelo y las trasmite a la vista.

Las principales ventajas que aporta este patrón es que desacopla el modelo de datos de su visualización, por lo que si es necesario modificar la vista de la aplicación, no será necesario alterar en absoluto el modelo.

---

[14] Fuente: wiki del Grupo Gimeno [4].

[15] Fuente: Head first: Design patterns [1]. Páginas 531-550.

La *figura 15* muestra un esquema de la dinámica que se establece entre modelo, vista y controlador.

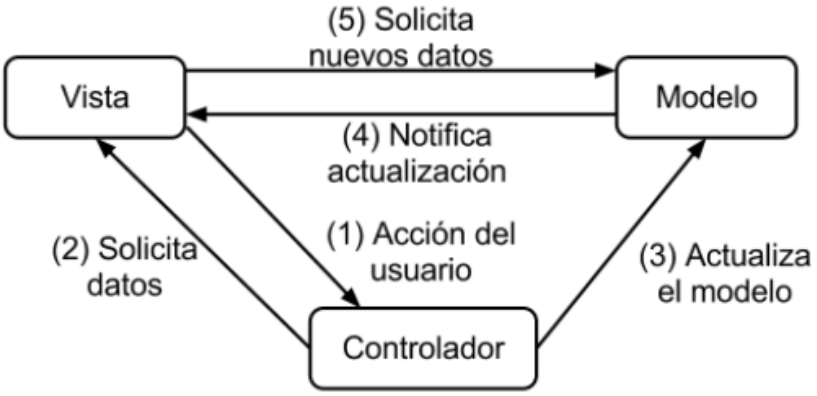

Figura 15. Dinámica de comunicación del patrón MVC.

5.1.3 Fuentes de datos

En este apartado se analizan las particularidades de cada una de las fuentes de información que la aplicación a desarrollar debe obtener. Como se ha indicado anteriormente en la fase de análisis, la información a monitorizar se obtiene de cuatro fuentes diferentes y en formatos diferentes, de manera que ha resultado imprescindible conocer las distintas fuentes de datos para implementar un sistema de calidad.

En primer lugar, las previsiones meteorológicas se han obtenido de un servicio de pago llamado ElTiempo24, este, mediante una petición a un recurso. En esta petición se incluye el código propio del servicio para la ciudad de la que se desean obtener las previsiones, el usuario que realiza la petición y una clave encriptada utilizando el algoritmo *md5*[16]. Para el tratamiento de los ficheros XML recibidos como respuesta con las predicciones de cada una de las poblaciones, se ha utilizado *Jackson*, una librería Java para el mapeado de ficheros JSON y XML.

En segundo lugar, los avisos meteorológicos se obtienen mediante *web scraping* de la web de AEMET. Esto se debe a que al inicio de las prácticas, AEMET lanzó una API REST gratuita para la consulta de la información que proporciona, pero a lo largo del desarrollo del proyecto nunca llegaron a estar funcionales los recursos de avisos meteorológicos. La técnica de *web scraping* es utilizada para la obtención automática de información de páginas web. Para realizar esta técnica, se ha utilizado *JSoup*, una librería Java que permite seleccionar de las páginas web el contenido del DOM que interesa obtener mediante la búsqueda de etiquetas y

[16] *Message Digest Algorithm 5* es un algoritmo de reducción criptográfico cuyo uso principal es comprobar que un archivo no ha sido modificado.

clases. Para ello, debe analizarse concienzudamente la estructura de la página web las secciones donde se encuentra la información relevante que se desea obtener.

En tercer lugar, las lluvias recientes se reciben a través de un fichero *.xls*. Para el tratamiento de este fichero, se ha utilizado Apache POI, una API de Java dedicada al tratamiento de documentos con formatos propios de Microsoft. Con la ayuda de esta API e iterando por filas el documento *.xls*, se realiza la lectura de los datos de las poblaciones de interés para las que se desea monitorizar la predicción. Como el documento solo contiene información de poblaciones pertenecientes a la cuenca hidrográfica del Júcar, para algunas poblaciones no disponemos de esos datos, en esos casos, el cliente ha definido que debe indicarse que no hay información disponible para dichas poblaciones.

Finalmente, el histórico de lluvias se obtiene de un servicio web REST propio del grupo. Para obtener los datos, basta con acceder al recurso correspondiente a la población de la que desee conocerse el histórico de lluvias.

5.1.4 Implementación de la interfaz de usuario

En esta sección se muestra la implementación de la interfaz de usuario de la aplicación. En el caso del proyecto a desarrollar, esta interfaz web no es muy extensa, puesto que el objetivo de monitorizar las predicciones es mostrar la información de forma compacta a la vez que intuitiva para que el usuario no tenga que navegar por diferentes páginas para consultar toda la información de interés.

La *figura 16* muestra el *dashboard* principal de la aplicación. Como puede observarse, toda la información referente a las predicciones meteorológicas se incluye en una única pantalla. La gran parte de la interfaz está ocupada por las previsiones meteorológicas. En ellas para cada periodo se muestra la información meteorológica exigida por el cliente.

Al lado derecho de las predicciones, se muestra la información referente a las lluvias acumuladas y al histórico de lluvias, en este caso, la información de lluvias recientes y el histórico no se ofrece para la población, que como puede observarse en el desplegable de la parte superior izquierda de la interfaz, es Pamplona. Debajo de las previsiones, se muestran los avisos. En el caso de la figura 16, no hay ningún aviso para la población, por lo que se muestra un mensaje indicándolo.

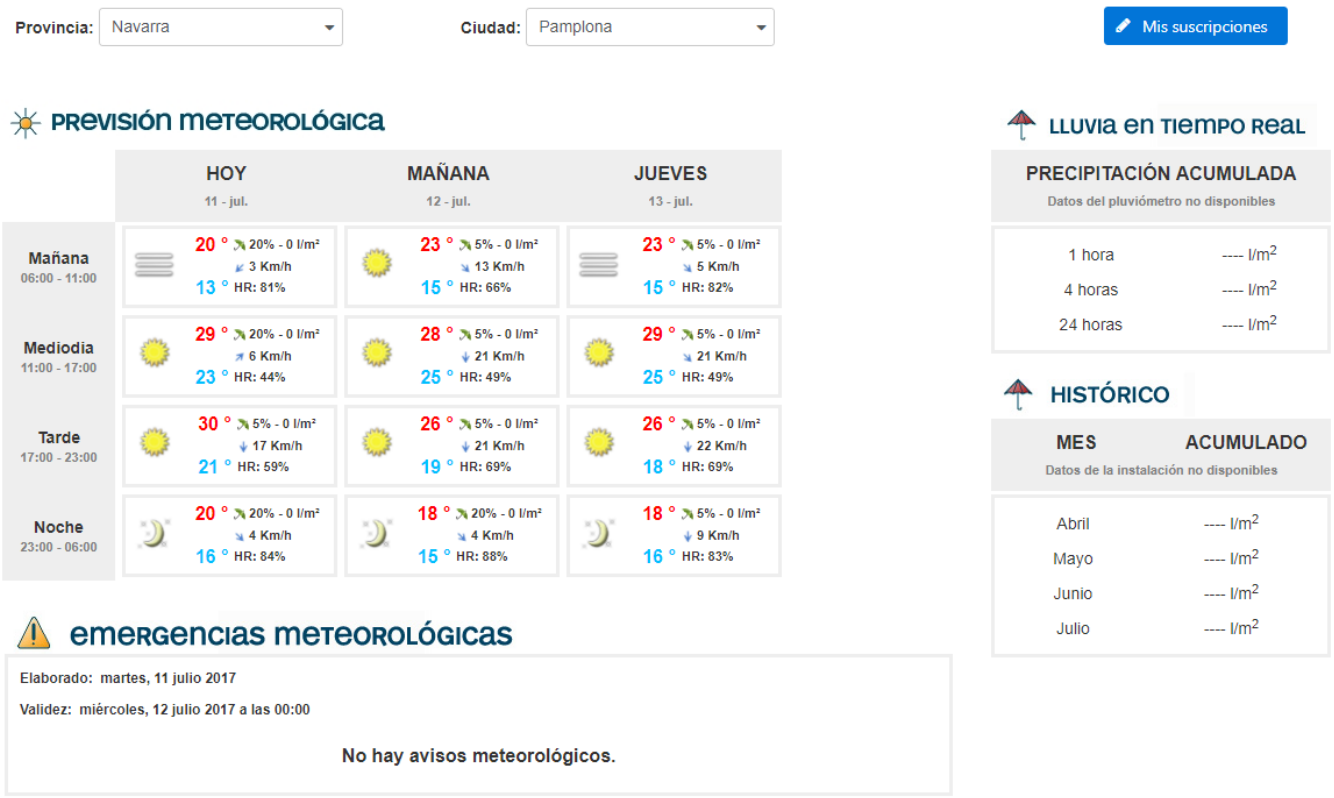

Figura 16. *Dashboard* principal de monitorización de predicciones.

Como puede observarse en la *figura 16*, en la parte superior izquierda se encuentran los selectores para elegir por provincia y población las predicciones que se desean visualizar. Gracias a Angular2, al seleccionar en el selector de provincias la provincia deseada, se cargan dinámicamente las ciudades con predicciones de esa provincia en el selector de ciudades para que el usuario pueda seleccionar de qué ciudad ver las predicciones. La *figura 17* y *18* muestra el comportamiento de cada uno de estos selectores.

Figura 17. Selector de provincias.

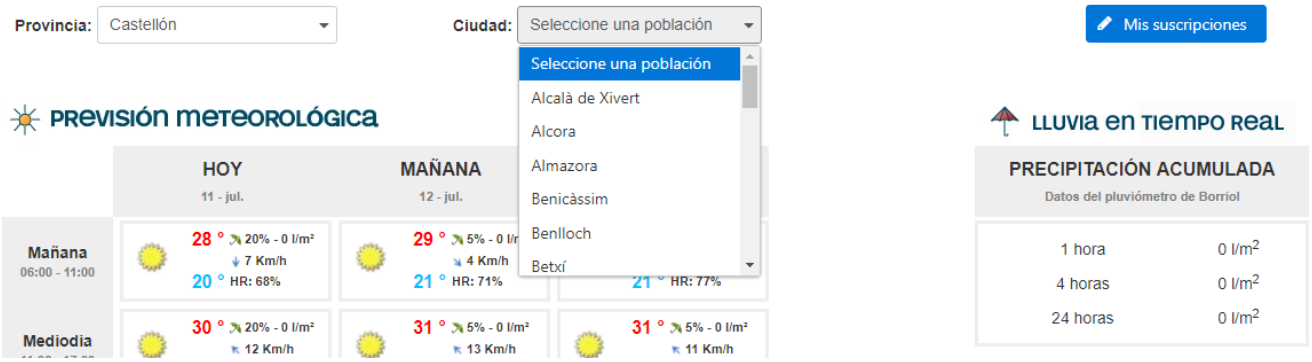

Figura 18. Selector de ciudades

Seguidamente, la *figura 19* muestra un *dashboard* de predicciones de una población para la que si se han emitido avisos meteorológicos. Como puede observarse, esta ciudad también dispone de datos para las lluvias recientes, pero no del histórico de lluvias.

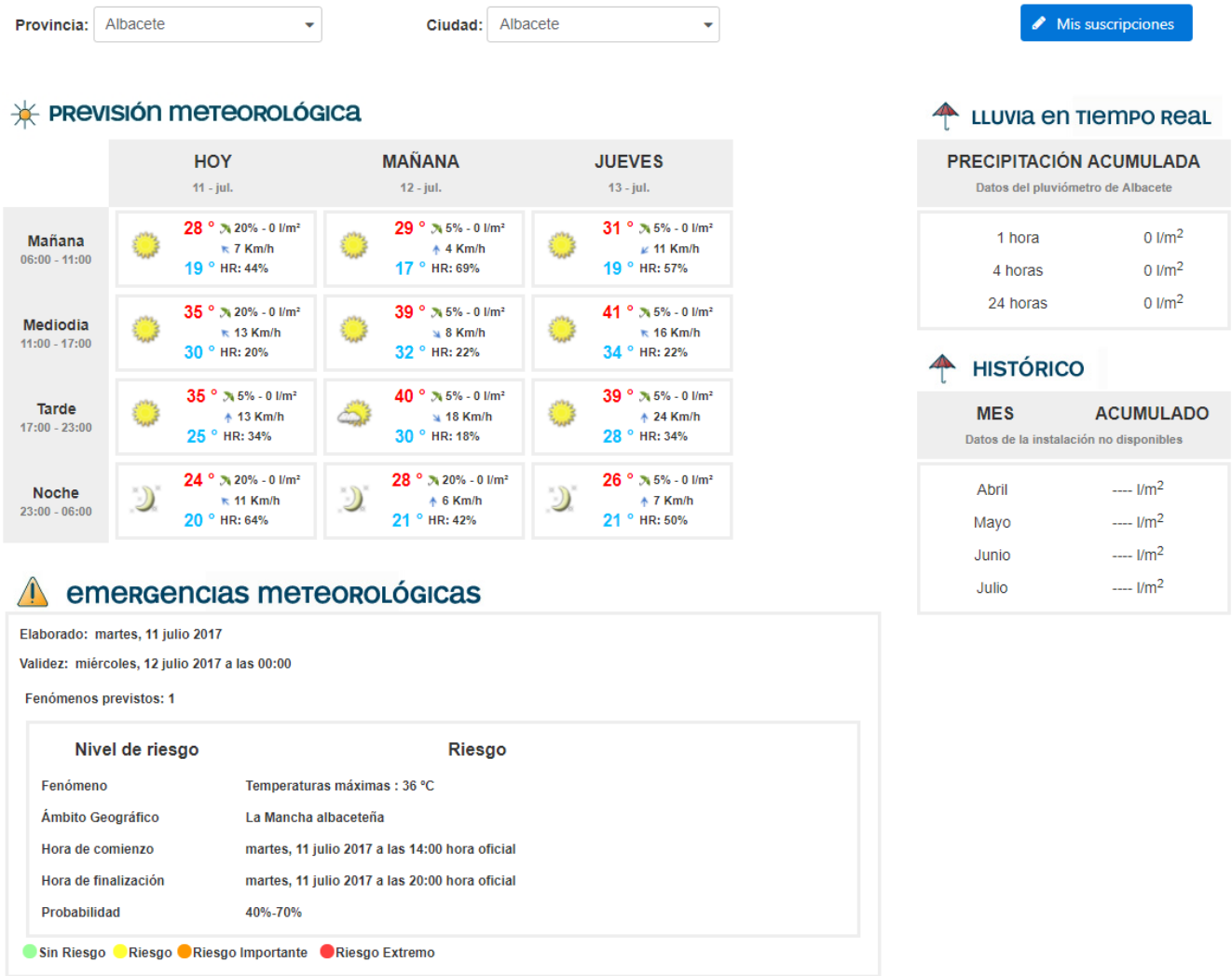

Figura 19. Dashboard de una población con avisos meteorológicos

Una vez analizado el *dashboard* principal encargado de monitorizar las predicciones meteorológicas, en todas las figuras expuestas en ese apartado puede verse en la parte superior derecha de la imagen el botón de enlace a las suscripciones. Al pulsar sobre dicho botón, la aplicación redirige al usuario a la gestión de las suscripciones, que puede observarse en la *figura 20*.

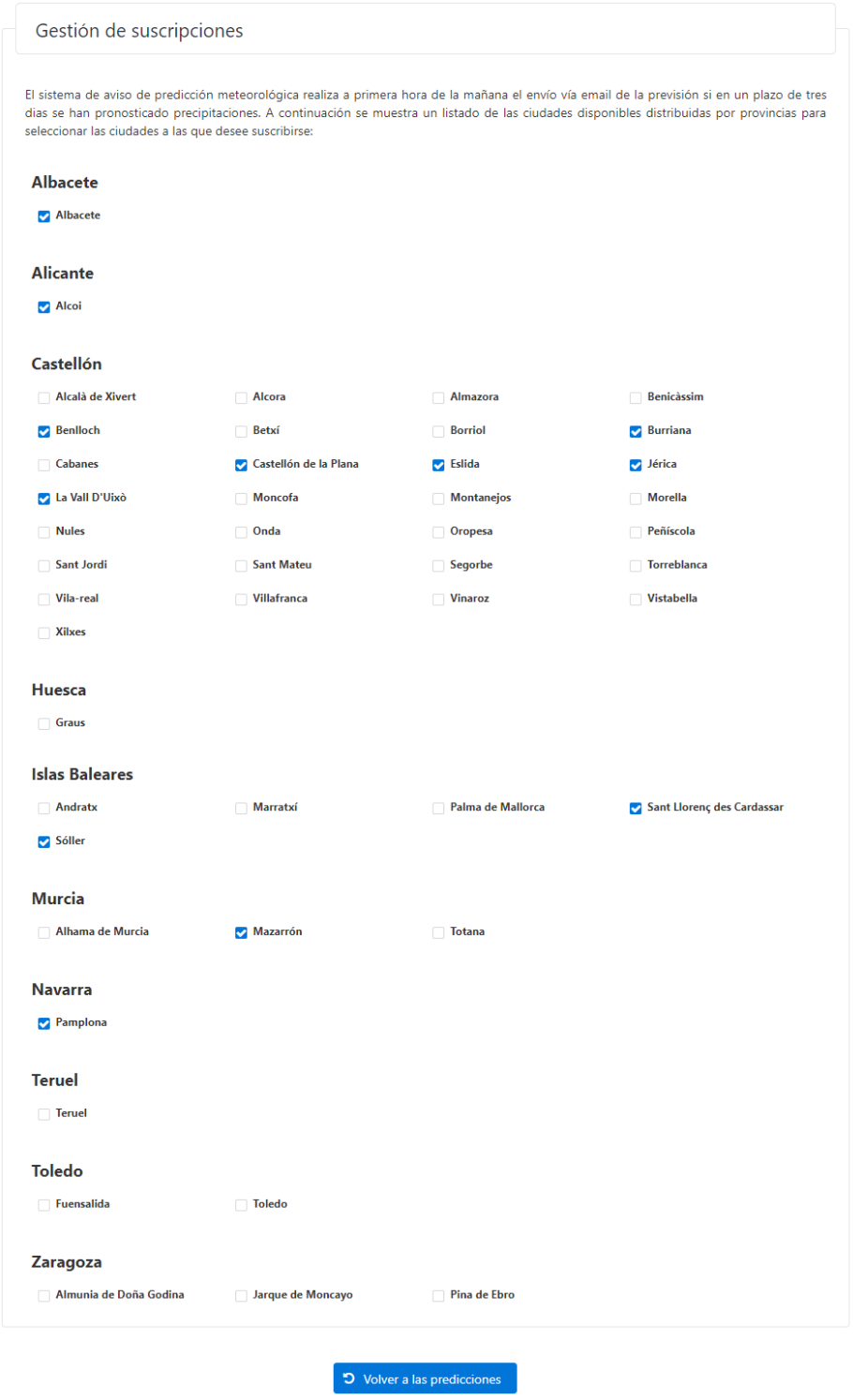

Figura 20. Interfaz de gestión de suscripciones.

Como puede observarse en la *figura 20*, cada población tiene asociado un *checkbox*, si dicho *checkbox* se encuentra seleccionado, eso significa que el usuario está suscrito a las predicciones de esa población. Además, cuando un usuario accede a la gestión de suscripciones, la aplicación selecciona automáticamente los *checkbox* de las ciudades a las que el usuario está suscrito.

Para suscribirse a nuevas predicciones o eliminar suscripciones, basta con seleccionar o deseleccionar el *checkbox* correspondiente a la suscripción sin necesidad de enviar ningún formulario, gracias al dinamismo que proporciona *Angular2* junto a *TypeScript*. En ambos casos, la aplicación pide una confirmación de la operación, como puede verse en las *figuras 21* y *22*.

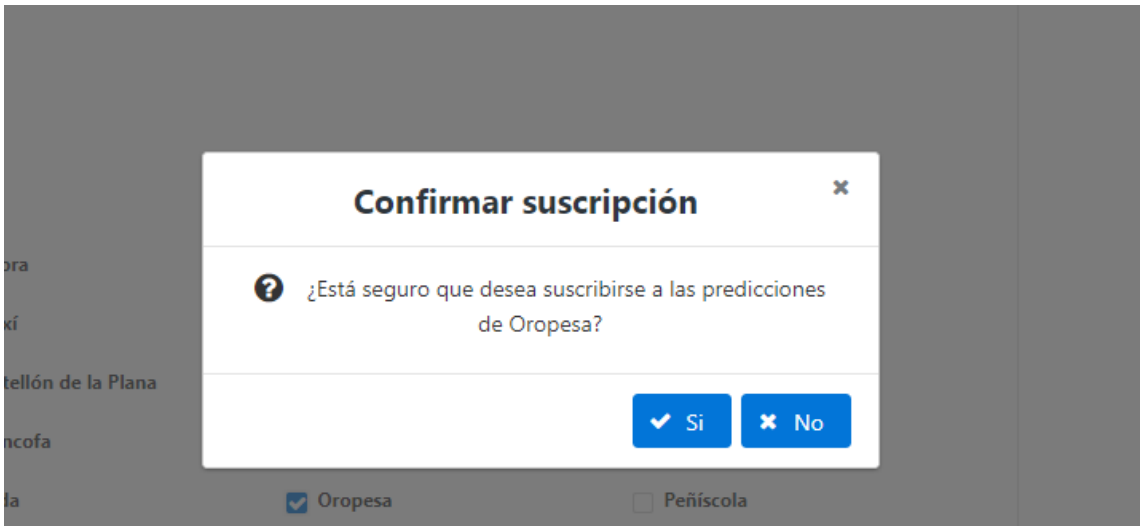

Figura 21. Confirmación de suscripciones.

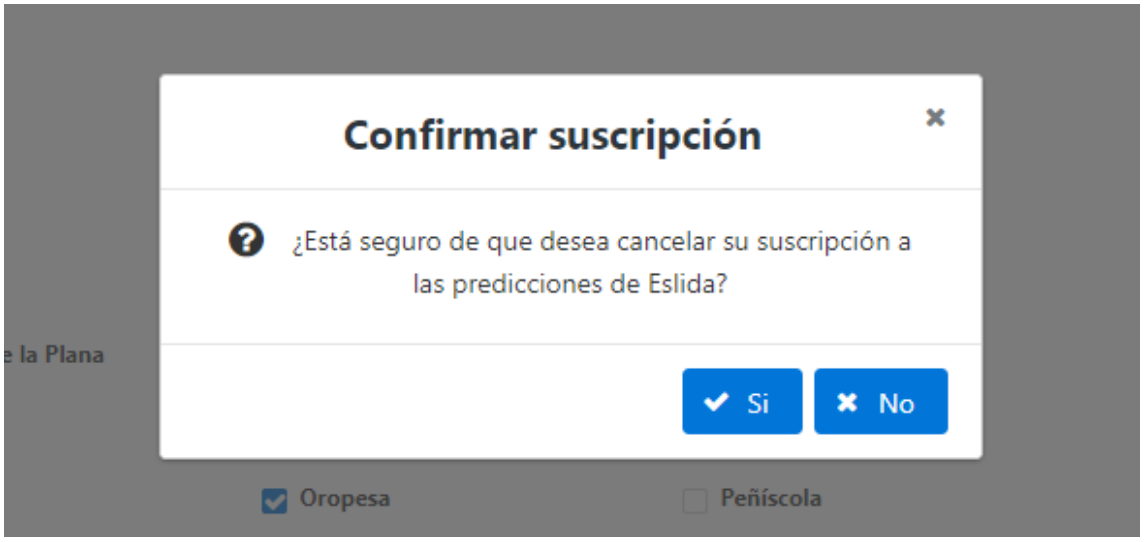

Figura 22. Confirmación de la cancelación de una suscripción

En ambos casos, si el usuario confirma la operación, el sistema ofrecerá un mensaje informativo al usuario, indicando si la acción se ha realizado con éxito o si por el contrario no se ha podido realizar la acción indicada por el usuario como se observa en la *figura 23*, donde no se ha podido realizar una suscripción, y *24*, donde una suscripción se ha cancelado satisfactoriamente.

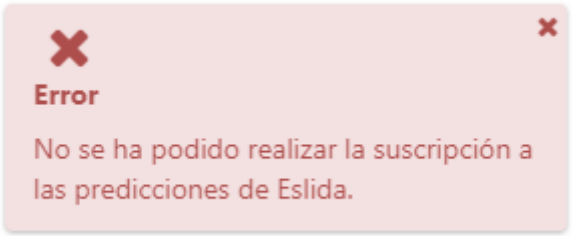

Figura 23. Confirmación de operación fallida.

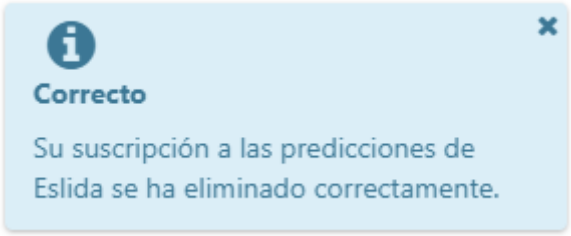

Figura 24. Confirmación de operación exitosa.

Tras mostrar la gestión, creación y eliminación de suscripciones, la *figura 25* muestra el *iframe* que se añadirá a la página principal de la intranet del Grupo Gimeno, desde donde los usuarios podrán acceder a la aplicación. Como puede observarse, dicho *iframe* muestra la previsión del periodo en el que se consulta de forma simplificada, mostrando el día y periodo de la previsión, la población a la que corresponde, la condición meteorológica esperada y la temperatura máxima y mínima para dicho periodo.

Figura 25. *Iframe* de resumen de predicción.

Para finalizar con este apartado, la *figura 26* muestra un ejemplo de correo electrónico informativo. Como se ha explicado anteriormente, estos correos sirven

para informar a los usuarios suscritos a una población de que se esperan lluvias en los próximos tres días, indicando la probabilidad de que se produzcan las lluvias, así como la cantidad de lluvia esperada.

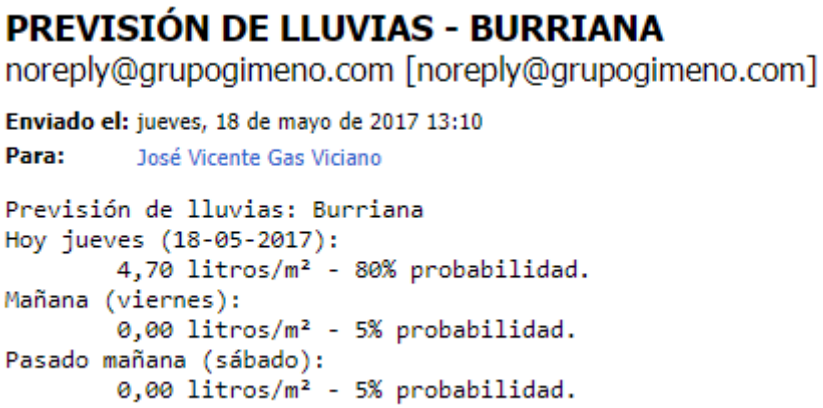

Figura 26. Mensaje informativo generado por la aplicación.

5.2 Verificación y validación

El objetivo de la verificación y validación del sistema es corroborar su correcto funcionamiento a todos los niveles. Esto proporciona información valiosa acerca de la calidad del sistema que se está desarrollando, además de asegurar que se cumplen sus requisitos funcionales.

Aunque existen diferentes tipos de pruebas según su alcance, en el proyecto se han desarrollado pruebas unitarias, de integración y de aceptación, que se explican en los siguientes apartados.

5.2.1 Pruebas unitarias

Las pruebas unitarias se encargan de comprobar el correcto funcionamiento interno de un componente concreto de la aplicación. Estas pruebas sirven para corroborar la correcta implementación de la lógica interna de cada uno de los componentes del sistema.

Para la implementación de pruebas unitarias se ha utilizado *JUnit*, una librería de pruebas unitarias para Java. Mediante el uso de anotaciones Java, se implementan clases de prueba con métodos que testearán cada uno de los componentes del sistema.

Estas pruebas han sido desarrolladas antes de la implementación de las tareas ya que, como se ha indicado en el apartado 3.5.3 de este documento; la implementación de las tareas ha sido guiada por pruebas.

5.2.2 Pruebas de integración

Una vez aprobadas las pruebas unitarias, se realizan las pruebas se integración. El objetivo de estas es verificar que un conjunto de componentes (no todos los del sistema) interactúan de forma correcta y su funcionamiento es el esperado.

Para la realización de este tipo de pruebas, la empresa utiliza *Jenkins*, un sistema de integración continua que se encarga de forma periódica y cada vez que una nueva versión de la aplicación es subida al repositorio *Git* del proyecto de recompilar todo el código de la aplicación, ejecutar las pruebas de unidad ya implementadas, y comprobar que su funcionamiento es satisfactorio.

En caso de que las pruebas unitarias fallen, *Jenkins* se encarga de enviar un correo electrónico a los desarrolladores implicados indicando el error. En este caso, los desarrolladores deberán detectar porqué se produce el problema y solventarlo para que las pruebas de integración sean satisfactorias.

5.2.3 Pruebas de aceptación

El objetivo de las pruebas de aceptación es comprobar a nivel general y sin entrar al detalle el correcto funcionamiento de la aplicación y asegurarse de que ésta cumple con todos los requisitos funcionales obtenidos durante el análisis del sistema.

Estas comprobaciones se han realizado tras la implementación de cada una de las tareas de desarrollo del proyecto. Una vez implementada una tarea el propio desarrollador comprueba las pruebas de aceptación. Además, antes de dar por finalizada la tarea, el *Scrum Manager* volverá a realizar dichas pruebas para asegurar el correcto funcionamiento antes de dar la tarea por correctamente finalizada.

A continuación, se muestran todas las pruebas de aceptación llevadas a cabo durante el desarrollo del proyecto:

Monitorización de datos (CU01, CU02, CU03, CU04, CU08):

- Se muestran los datos de las previsiones meteorológicas de una población.
- Se muestran los datos de los avisos meteorológicos de una población.
- Se muestran los datos de las lluvias recientes de una población.
- Se muestran los datos del histórico de lluvias de una población.
- Se puede seleccionar la población de la que se desea ver las predicciones.
- Si alguno de los datos no está disponible, se muestra un mensaje informativo en el lugar donde debería aparecer la información.
- Aunque algunos de los datos no estén disponibles, el resto de datos que sí lo están siguen monitorizándose.

Adquisición de datos (CU10, CU11, CU12, CU13):

- Los datos de las previsiones meteorológicas se obtienen a diario una única vez y se almacenan en la base de datos.
- Los datos de los avisos meteorológicos se obtienen de forma periódica cada 15 minutos y se almacenan en la base de datos.
- Los datos de las lluvias recientes se obtienen de forma periódica cada 15 minutos y se almacenan en base de datos.
- Los datos del histórico de lluvias se obtienen cada vez que se hace una petición de las predicciones de una población por parte de un usuario.
- Si alguno de los procesos de lectura y almacenamiento de información falla, se registra el error para que el responsable de la aplicación esté al corriente del fallo.
- Si la obtención de previsiones meteorológicas falla, se vuelve a realizar el proceso en 30 minutos.

Gestión de suscripciones (CU05, CU06, CU07):

- Desde la pantalla de monitorización se puede acceder a la gestión de suscripciones.
- Desde la interfaz de gestión de suscripciones se puede volver a la pantalla de monitorización de predicciones.
- Al iniciar la gestión de suscripciones se cargan las suscripciones actuales del usuario.
- Al realizar una nueva suscripción se muestra un mensaje de confirmación antes de registrarla.
- Tras realizar una nueva suscripción se muestra un mensaje informativo indicando el éxito o el fracaso de la operación.
- Al cancelar una suscripción se muestra un mensaje de confirmación antes de eliminarla.
- Tras cancelar una suscripción se muestra un mensaje informativo indicando el éxito o el fracaso de la operación.

Envío de emails informativos (CU09):

- El usuario sólo recibe correos electrónicos cuando esté previsto que la ciudad a la que está suscrito registre lluvias en los tres siguientes días.
- Los correos electrónicos muestran claramente a que ciudad hacen referencia.
- Los correos electrónicos muestran la cantidad de lluvia esperada para cada uno de los tres días siguientes.

Capítulo 6

Conclusiones

En esta memoria se ha documentado el desarrollo de una aplicación web encargada de monitorizar las predicciones del tiempo para distintas poblaciones. Además de la implementación propiamente dicha, se ha realizado una adecuada planificación del proyecto, así como un análisis detallado del sistema que posteriormente se ha implementado.

Personalmente, considero que la dificultad del proyecto, más allá del desarrollo de la interfaz web, ha sido conocer, comprender y aprender a utilizar la gran variedad de tecnologías utilizadas para su desarrollo.

Como opinión personal, valoro de forma muy positiva la estancia en la empresa. Además de una primera toma de contacto con el mundo laboral en el sector del desarrollo de software, considero que he ampliado mis conocimientos de forma notable en tecnologías para mí novedosas como *Angular2* o *TypeScript*. Aunque ha sido mi primera experiencia en un proyecto de gran envergadura, opino que la base formativa adquirida durante el Grado en Ingeniería Informática me ha permitido adquirir nuevos conocimientos de forma más rápida.

Además considero que he adquirido más aptitudes para el trabajo en equipo. Para ello ha sido fundamental la gran acogida que me han brindado tanto los compañeros del equipo de trabajo como sus responsables.

6.1 Extensiones del proyecto

Para terminar con esta memoria, resulta interesante comentar una serie de posibles mejoras a aplicar en un futuro en el proyecto. A continuación se describen las más interesantes.

- Puesto que el entorno de producción donde se desplegará la aplicación dispone de un clúster de servidores, será conveniente separar los distintos módulos de la aplicación en distintos proyectos. Así, sería útil separar en proyectos diferentes el cliente de mensajes de correo electrónico, los procesos de importación de datos y el modelo de datos (que será común para el resto de subproyectos.

- Buscar datos de lluvias acumuladas en otros S.A.I.H. de las distintas cuencas hidrográficas con la finalidad de tener datos de lluvias recientes para todas las poblaciones.

- Una vez esté completamente operativo el API REST de AEMET, sería muy útil obtener los avisos meteorológicos a través de ella, puesto que al obtenerlos

mediante *web scraping*, un cambio en la estructura interna de la página web de AEMET provocaría la necesidad de modificar la implementación de los importadores de datos.

- La implementación de nueva funcionalidad en la interfaz web que permita a los administradores del sistema añadir nuevas poblaciones a monitorizar de forma sencilla sin necesidad de manipular directamente la base de datos. Esta mejora, aunque pudiera parecer un gran avance, realmente no lo es tanto puesto que no será ni mucho menos habitual añadir nuevas poblaciones a monitorizar.

Bibliografía

1. Bates, B.; Freeman, E.; Robson, E; Sierra, K. *Head first: Design patterns*. O'Reilly. 2009.
2. Martin, R.C. *Clean Code: A Handbook of Agile Software Craftmanship*. Prentice Hall. 2008.
3. Brisbin, J; et alt. *Spring Data: Modern Data Access for Enterprise Java*. O'Reilly. 2012.
4. Wiki del Grupo Gimeno. Aplicación web privada.
5. Schwaber, K.; Sutherland, J. *The Scrum Guide*. Accesible a través de http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf. 2016.
6. Carroll, ER. Estimating Software Based on Use Case Points. Accesible a través de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.2621&rep=rep1&type=pdf. 2005.

Anexo A

Requisitos funcionales

Requisito funcional	
Identificador	CU01
Nombre	Consultar previsión de una población
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir la visualización de las previsiones meteorológicas a tres días vista para una población.
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU02, CU03, CU04, CU08, CU10
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa.
Trigger	El usuario desea consultar la monitorización de las predicciones meteorológicas.
Secuencia normal	1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra la previsión meteorológica emitida el día en curso de la población.
Excepciones	El sistema no dispone de previsiones del día actual para la población. 1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra un mensaje informativo indicando que no dispone de los datos solicitados. 3 – El usuario visualiza el resto de datos de las predicciones.
Frecuencia esperada	Muchas veces al día (Entre 500 y 1000).
Importancia	Alta
Prioridad	Alta
Comentarios	Los datos de las previsiones de una población se mostrarán siempre junto a los datos de los avisos, lluvias recientes e histórico de lluvias de la misma población.

Tabla 9. Requisito funcional CU01

Requisito funcional	
Identificador	CU02
Nombre	Consultar avisos de una población
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir la visualización de los avisos meteorológicos para una población
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU1, CU03, CU04, CU08, CU11
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa.
Trigger	El usuario desea consultar la monitorización de las predicciones meteorológicas.
Secuencia normal	1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra los avisos meteorológicos emitidos para el día en curso de la población.
Excepciones	No hay avisos meteorológicos para la población. 1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra un mensaje informativo indicando que no hay ningún aviso meteorológico para la población. 3 – El usuario visualiza el resto de datos de las predicciones. No hay datos sobre los avisos meteorológicos para la población. 1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra un mensaje informativo indicando que no hay ningún aviso meteorológico para la población. 3 – El usuario visualiza el resto de datos de las predicciones.
Frecuencia esperada	Muchas veces al día (Entre 500 y 1000).
Importancia	Alta
Prioridad	Alta
Comentarios	Los datos de los avisos meteorológicos de una población se mostrarán siempre junto a los datos de las previsiones, lluvias recientes e histórico de lluvias de la misma población.

Tabla 10. Requisito funcional CU02

Requisito funcional	
Identificador	CU03
Nombre	Consultar lluvias recientes de una población
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir la visualización de la información relativa a las lluvias recientes acumuladas en una población.
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU01, CU02, CU04, CU08, CU12
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa.
Trigger	El usuario desea consultar la monitorización de las predicciones meteorológicas.
Secuencia normal	1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra las lluvias recientes de la población.
Excepciones	No hay datos sobre las lluvias recientes para la población. 1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra un mensaje indicando que no hay información sobre las lluvias recientes disponible. 3 – El usuario visualiza el resto de datos de las predicciones.
Frecuencia esperada	Muchas veces al día (Entre 500 y 1000).
Importancia	Alta
Prioridad	Alta
Comentarios	Los datos de lluvias recientes de una población se mostrarán siempre junto a los datos de las previsiones, avisos e histórico de lluvias de la misma población.

Tabla 11. Requisito funcional CU03

Requisito funcional	
	CU04
Nombre	Consultar histórico de lluvias de una población
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir la visualización de la información relativa al histórico de lluvias de una población.
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU01, CU02, CU03, CU08, CU13
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa.
Trigger	El usuario desea consultar la monitorización de las predicciones meteorológicas.
Secuencia normal	1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra el histórico de lluvias de los últimos meses de la población.
Excepciones	No hay datos sobre el histórico de lluvias de la población. 1 – El usuario selecciona ver la monitorización de las predicciones de una población. 2 – El sistema muestra un mensaje indicando que no hay información disponible sobre el histórico de lluvias. 3 – El usuario visualiza el resto de datos de las predicciones.
Frecuencia esperada	Muchas veces al día (Entre 500 y 1000).
Importancia	Alta
Prioridad	Alta
Comentarios	Los datos del histórico de lluvias de una población se mostrarán siempre junto a los datos de las previsiones, avisos y lluvias recientes de la misma población.

Tabla 12. Requisito funcional CU04

Requisito funcional	
Identificador	CU05
Nombre	Suscribirse a predicciones de una población
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir a los usuarios realizar suscripciones a las predicciones de una población
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU06, CU07, CU09
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa y no está suscrito a las predicciones de la población a la que desea suscribirse.
Trigger	El usuario desea suscribirse a las predicciones de una población.
Secuencia normal	1 – El usuario indica que desea realizar una suscripción. 2 – El sistema muestra al usuario las diferentes poblaciones a las que puede suscribirse. 3 – El usuario selecciona una población a la que suscribirse. 4 – El sistema registra la suscripción e indica al usuario que la suscripción se ha realizado con éxito.
Excepciones	Surge un error durante el registro de la suscripción. 1 – El sistema informa al usuario de que no se ha podido realizar la suscripción. 2 – El sistema vuelve a mostrar al usuario las diferentes poblaciones a las que puede suscribirse.
Frecuencia esperada	Varias veces al día (10 – 20 veces).
Importancia	Alta
Prioridad	Media
Comentarios	

Tabla 13. Requisito funcional CU05

Requisito funcional	
Identificador	CU06
Nombre	Eliminar suscripción a una población
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir a los usuarios cancelar sus suscripciones a las predicciones de una población
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU05, CU07, CU09
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa y debe estar suscrito a las predicciones de la población de la que desea cancelar la suscripción.
Trigger	El usuario desea cancelar su suscripción a las predicciones de una población
Secuencia normal	1 – El usuario indica que desea cancelar una suscripción. 2 – El sistema muestra al usuario las diferentes poblaciones a las que puede suscribirse además de las poblaciones a las que está suscrito. 3 – El usuario selecciona la suscripción que desea cancelar. 4 – El sistema elimina la suscripción e indica al usuario que la operación se ha realizado con éxito.
Excepciones	Surge un error durante la eliminación de la suscripción. 1 – El sistema informa al usuario de que no se ha podido eliminar la suscripción. 2 – El sistema vuelve a mostrar al usuario las diferentes poblaciones a las que puede suscribirse y las poblaciones a las que está suscrito.
Frecuencia esperada	Pocas veces al día (1-5 veces)
Importancia	Alta
Prioridad	Media
Comentarios	

Tabla 14. Requisito funcional CU06

Requisito funcional	
Identificador	CU07
Nombre	Consultar suscripciones
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir a los usuarios consultar a qué poblaciones están suscritos.
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU05, CU06, CU09
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa.
Trigger	El usuario desea consultar qué suscripciones tiene registradas.
Secuencia normal	1 – El usuario indica que desea consultar sus suscripciones. 2 – El sistema muestra al usuario las diferentes poblaciones disponibles, diferenciando las que está suscrito de las que no.
Excepciones	
Frecuencia esperada	Varias veces al día (10-20 veces).
Importancia	Alta
Prioridad	Media
Comentarios	

Tabla 15. Requisito funcional CU07

Requisito funcional	
Identificador	CU08
Nombre	Seleccionar predicciones por provincia y población
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe permitir al usuario poder seleccionar la población de la que consultar sus predicciones, realizando la búsqueda por provincia y población.
Nivel	Tarea principal
Actor principal	Usuario
Actores secundarios	-
Relaciones	CU01, CU02, CU03, CU04
Precondición	El usuario dispone de una sesión iniciada en la intranet de la empresa.
Trigger	El usuario desea consultar las predicciones de una población concreta.
Secuencia normal	1 – El usuario indica que quiere seleccionar una población para consultar sus predicciones. 2 – El sistema muestra las diferentes provincias de las que dispone de predicciones. 3 – El usuario elige una provincia. 4 – El sistema muestra las diferentes poblaciones de las que se dispone de predicciones en la provincia indicada por el usuario. 5 – El usuario selecciona una población. 6 – El sistema muestra las predicciones meteorológicas de la población indicada por el usuario.
Excepciones	
Frecuencia esperada	Muchas veces al día (500-1000 veces).
Importancia	Alta
Prioridad	Media
Comentarios	

Tabla 16. Requisito funcional CU08

Requisito funcional	
Identificador	CU09
Nombre	Enviar correos electrónicos informativos
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe notificar, mediante correo electrónico a los usuarios suscritos a las predicciones de una población; que se esperan lluvias en un plazo de tres días en dicha población.
Nivel	Tarea principal
Actor principal	Servidor de correo
Actores secundarios	Usuario
Relaciones	CU05, CU06, CU07
Precondición	Deben existir previsiones para la población el día actual. Además, en esa predicción deben esperarse lluvias en un plazo de tres días.
Trigger	El sistema comprueba que se esperan lluvias en una población y que no se han enviado correos informativos de esa población ese mismo dia.
Secuencia normal	1 – El sistema indica al servidor de correo los detalles del mensaje a enviar. 2 – El servidor de correo realiza el envío del mensaje. 3 – El servidor de correo notifica al sistema que el envío se ha realizado correctamente.
Excepciones	No puede realizarse el envío de los correos electrónicos. 1 – El sistema registra la incidencia que ha provocado que no pueda realizarse el envío de correos. 2 – El sistema vuelve a intentar realizar el envío en un plazo de cinco minutos.
Frecuencia esperada	Muchas veces al dia (150-300 veces)
Importancia	Alta
Prioridad	Alta
Comentarios	

Tabla 17. Requisito funcional CU09

Requisito funcional	
Identificador	CU10
Nombre	Obtener datos de previsiones meteorológicas.
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe obtener de forma periódica, mientras no haya previsiones para cada una de las poblaciones, las previsiones meteorológicas
Nivel	Tarea principal
Actor principal	Servicio ElTiempo24
Actores secundarios	-
Relaciones	CU01
Precondición	No deben existir en el sistema previsiones emitidas el día en curso para la población de las que van a obtenerse los datos.
Trigger	El sistema ejecuta un proceso programado para la obtención de las previsiones meteorológicas.
Secuencia normal	1 – El sistema comunica al servicio ElTiempo24 que desea obtener las previsiones de las poblaciones. 2 – El servicio ElTiempo 24 ofrece las previsiones para las poblaciones. 3 – El sistema almacena la información obtenida.
Excepciones	No pueden obtenerse las previsiones 1 – El sistema registra la incidencia que ha provocado que no puedan obtenerse las predicciones. 2 – El sistema vuelve a intentar obtener los datos en un plazo de treinta minutos.
Frecuencia esperada	Una vez al día para cada población (siempre que no se produzcan excepciones
Importancia	Alta
Prioridad	Alta
Comentarios	

Tabla 18. Requisito funcional CU10

Requisito funcional	
Identificador	CU11
Nombre	Obtener datos de avisos meteorológicos.
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe obtener de forma periódica los avisos meteorológicos de las poblaciones a monitorizar.
Nivel	Tarea principal
Actor principal	Web AEMET
Actores secundarios	-
Relaciones	CU02
Precondición	-
Trigger	El proceso programado inicia la obtención de datos de los avisos meteorológicos
Secuencia normal	1 – El sistema comunica a La web AEMET que desea obtener los avisos meteorológicos de las poblaciones. 2 – La web AEMET ofrece los avisos para las poblaciones. 3 – El sistema almacena la información obtenida.
Excepciones	No pueden obtenerse los avisos 1 – El sistema registra la incidencia que ha provocado que no puedan obtenerse los avisos. 2 – El sistema vuelve a intentar realizar el envío en un plazo de quince minutos.
Frecuencia esperada	Varias veces al dia (100 veces)
Importancia	Alta
Prioridad	Alta
Comentarios	

Tabla 19. Requisito funcional CU11

Requisito funcional	
Identificador	CU12
Nombre	Obtener datos de lluvias recientes.
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe obtener de forma periódica la información referente a las lluvias recientes de las poblaciones a monitorizar.
Nivel	Tarea principal
Actor principal	Servicio S.A.I.H.
Actores secundarios	-
Relaciones	CU03
Precondición	-
Trigger	El proceso programado inicia la obtención de datos de las lluvias recientes.
Secuencia normal	1 – El sistema comunica al servicio S.A.I.H. que desea obtener las lluvias recientes de las poblaciones. 2 – El servicio S.A.I.H. ofrece la información referente a las lluvias recientes para las poblaciones. 3 – El sistema almacena la información obtenida.
Excepciones	No pueden obtenerse las lluvias recientes. 1 – El sistema registra la incidencia que ha provocado que no puedan obtenerse las lluvias recientes. 2 – El sistema vuelve a intentar realizar el envío en un plazo de quince minutos.
Frecuencia esperada	Varias veces al día (100 veces)
Importancia	Alta
Prioridad	Alta
Comentarios	

Tabla 20. Requisito funcional CU12

Requisito funcional	
Identificador	CU13
Nombre	Obtener datos del histórico de lluvias.
Autor	José Vicente Gas Viciano
Versión	1
Descripción	El sistema debe obtener el histórico de lluvias de las poblaciones a monitorizar.
Nivel	Tarea principal
Actor principal	Servicio SiccaEdar
Actores secundarios	Usuario
Relaciones	CU04
Precondición	-
Trigger	Un usuario desea ver la monitorización de las predicciones de una ciudad.
Secuencia normal	1 – El sistema comunica al servicio SiccaEdar que desea obtener las lluvias recientes de una población. 2 – El servicio SiccaEdar. ofrece la información referente al histórico de lluvias para la población. 3 – El sistema muestra la información obtenida
Excepciones	No pueden obtenerse las lluvias recientes. 1 – El sistema registra la incidencia que ha provocado que no puedan obtenerse las lluvias recientes. 2 – El sistema informa al usuario de que no han podido obtenerse el histórico de lluvias.
Frecuencia esperada	Muchas veces al dia (500-1000 veces)
Importancia	Alta
Prioridad	Alta
Comentarios	

Tabla 21. Requisito funcional CU13

Anexo B

Requisitos de datos

Requisito de datos	
Identificador	RD01
Nombre	Predicción
Datos	Fecha de emisión, fecha de predicción, periodo, condición meteorológica, descripción, temperatura máxima, temperatura mínima, probabilidad de lluvia, lluvia, dirección del viento, velocidad del viento, humedad relativa.
Comentarios	Representa la previsión meteorológica para una población en una fecha y un periodo determinado, obtenidas del servicio web de ElTiempo24.

Tabla 22. Requisito de datos RD01

Requisito de datos	
Identificador	RD02
Nombre	Aviso meteorológico
Datos	Fenómeno, nivel, fecha, hora de inicio, hora de fin, probabilidad, comentarios
Comentarios	Representa un aviso por emergencia meteorológica de una población en una fecha concreta, obtenidos de la página web de AEMET.

Tabla 23. Requisito de datos RD02

Requisito de datos	
Identificador	RD03
Nombre	Lluvia acumulada
Datos	Lluvia acumulada en la última hora, lluvia acumulada en las últimas 4 horas, lluvia acumulada en las ultimes 24 horas
Comentarios	Representa las lluvias recientes de una población, obtenidos de la página web de la CHJ.

Tabla 24. Requisito de datos RD03

Requisito de datos	
Identificador	RD04
Nombre	Histórico de lluvias
Datos	Lluvia mes actual, lluvia mes anterior, lluvias dos meses antes, lluvias tres meses antes
Comentarios	Representa el total de lluvias acumuladas en los últimos meses, obtenidos del servicio web de SiccaEdar.

Tabla 25. Requisito de datos RD04

Requisito de datos	
Identificador	RD05
Nombre	Población
Datos	Nombre, predicciones meteorológicas, avisos meteorológicos, lluvias recientes e histórico de lluvias.
Comentarios	Representa las diferentes poblaciones para las que se obtienen los datos relativos a las predicciones meteorológicas.

Tabla 26. Requisito de datos RD05

Requisito de datos	
Identificador	RD06
Nombre	Provincia
Datos	Nombre, ciudades de la provincia
Comentarios	Representa cada una de las provincias de las que existen ciudades de las cuales se monitoriza la predicción meteorológica.

Tabla 27. Requisito de datos RD06

Requisito de datos	
Identificador	RD07
Nombre	Suscripción
Datos	Ciudad suscrita, email usuario suscriptor.
Comentarios	Representa cada uno de los registros de los usuarios de la intranet del Grupo Gimeno a las predicciones de una ciudad.

Tabla 28. Requisito de datos RD07