



**UNIVERSIDAD DE CASTILLA-LA MANCHA**

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

**GRADO EN INGENIERÍA INFORMÁTICA**

TECNOLOGÍA ESPECÍFICA DE

INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

**“**[**Uso de tecnología CEP para la detección de desgaste en aerogeneradores**](https://www.esiiab.uclm.es/asig.php?curso=2017-18&codasig=42358&codtfg=1397)**”**

Enrique Brazález Segovia



Junio de 2018



**UNIVERSIDAD DE CASTILLA-LA MANCHA**

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

**GRADO EN INGENIERÍA INFORMÁTICA**TECNOLOGÍA ESPECÍFICA DE

INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

[**Uso de tecnología CEP para la detección de desgaste en aerogeneradores**](https://www.esiiab.uclm.es/asig.php?curso=2017-18&codasig=42358&codtfg=1397)

|  |  |
| --- | --- |
| Autor: | Enrique Brazález Segovia. |
| Directores: | Gregorio Diáz Descalzo. |

Junio 2018

A todos aquellos que me han ayudado y formado parte de estos 4 años de carrera. Gracias.

Declaración de Autoría

 Yo, Enrique Brazález Segovia, con DNI 74519608E, declaro que soy el único autor del Trabajo Fin de Grado titulado “Uso de tecnología CEP para la detección de desgaste en aerogeneradores” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está propiadamente atribuido a sus legítimos autores.

Albacete, a miércoles, 13 de junio de 2018.



 Fdo.: Enrique Brazález Segovia.

Resumen

Internet of Things es un concepto tecnológico que está revolucionando la forma de concibir nuestro día a día estos últimos años. La integración de sistemas informáticos en nuestros hogares, en nuestros puestos de trabajo, en la industria es muy notable, y en algunos casos, vital. Son muchas las tecnologías que toman participación en este concepto, y una de ellas es “Complex Event Processing” o “Procesamiento de Eventos Complejos” (CEP).

CEP, es una nueva forma de analizar datos. Una aplicación de Fast Data, que dota a los sistemas que lo soportan de una capacidad de sintésis, y análisis de datos abrumadora. Como toda tecnología tiene sus ventajas y desventajas, pero sin duda alguna, hemos de destacar que el procesamiento de eventos complejos dota de una de las capacidades de análisis más rápidas del mercado, permitiendo la toma de decisiones a tiempo real.

El escenario en el que nos encontramos en este trabajo es el de las energías renovables, en concreto, la energía eólica. A rasgos generales, el funcionamiento de un molino de viento es muy sencillo, a través de la fuerza que el viento ejerce sobre sus palas se generá energía mecánica sobre el rotor, y este a su vez electricidad a través de un generador. Lo que intentaremos en este trabajo de fin de grado es aplicar CEP, para la optimización de obtención de energía eléctrica a través aerogeneradores.

En este trabajo de fin de grado se desarrolla un sistema de predicción del desgaste en aerogeneradores en base a los datos obtenidos por su sistema de monitorización. El objetivo del sistema a desarrollar es, utilizando el procesamiento de eventos complejos y tecnologías de código abierto, implementar un sistema que nos permita predecir cuando el comportamiento los aerogeneradores dentro de un parque eólico es anómalo. Por tanto, aquel aerogenerador que tenga un comportamiento poco común tendrá mas probabilidades de tener alguna avería, actualmente, o a corto-medio plazo.

Abstract

Internet of Things is a technological concept that is revolutionizing the way we conceive our day to day in recent years. The integration of computer systems in our homes, in our jobs, in the industry is very remarkable, and in some cases, vital. There are many technologies that participate in this concept, and one of them is the "Processing of Complex Events" or "Procesamiento de Eventos Complejos” (CEP).

CEP, is a new way to analyze data. An application of Fast Data, which endows the systems that support it with syntax capability, and overwhelming data analysis. As each technology has its advantages and disadvantages, but without any doubt, we must emphasize that complex event processing provides one of the fastest analysis capabilities of the market, allowing real-time decision making.

The scenario in which we find ourselves in this work is renewable energy, specifically, wind energy. In general terms, the operation of a windmill is very simple, through the force applied by the wind in its blades, mechanical energy is generated in the rotor, generating electricity through a generator. What we will try in this end-of-degree project is to apply CEP, for the optimization of obtaining electricity through wind turbines.

In this final degree project, a wind weathering prediction system is developed based on the data obtained by its monitoring system. The objective of the system is to develop, using the processing of complex events and open source technologies, implement a system that allows us to predict when the behavior of wind turbines within a wind farm is anomalous. Therefore, the wind turbine that has an unusual behavior is more likely to be damaged, currently, or a short-medium term.

Agradecimientos

Me gustaría dar las gracias en primer lugar a mis padres, por brindarme la oportunidad de poder estudiar la carrera que quería desde un primer momento. Por apoyarme día sí, día también, desde que empecé mi camino como ingeniero informático, y hacerme ver que no todo en la vida es ser bueno en tu profesión, sino también hay que ser buena persona, buen hijo y buen amigo.

A mi director de trabajo de fin de grado, Gregorio Diáz Descalzo y a Juan Boubeta Puig, por ayudarme en el desarrollo de este proyecto e introducirme en el mundo del Procesamiento de Eventos Complejos.

A Ingeteam S.A por su colaboración tanto ocupando el rol de clientes, como de facilitarme formación en energías renovables.

A mis hermanos y a mi novia, que han sabido reconducirme y tranquilizarme en todo momento, aguantando mis nervios y mi mal humor por el estrés.

Y por último a mis amigos por su incondicional y gran apoyo.

Índice de Contenido

Declaración de Autoría iii

Resumen v

Abstract vi

Agradecimientos vii

CAPÍTULO 1. Introducción 21

1.1. Motivación 21

1.2. Objetivos 22

1.3. Estructura de la memoria 23

CAPÍTULO 2. Estado del arte 25

2.1. Fundamentos teóricos. 26

2.1.1. Arquitectura orientada a Servicios (SOA) 26

2.1.2. Arquitectura dirigida por eventos (EDA). 27

2.1.3. Arquitectura orientada a servicios dirigido por eventos (SOA 2.0) 28

2.1.4. Procesamiento de eventos complejos (CEP) 30

2.1.4.1 Tipos de procesamiento 31

2.2. Diagrama de funcionamiento 32

2.3. ESPER EPL. Lenguaje de programación para CEP. 34

2.3.1. Sintaxis 35

2.4. Discusión tecnológica 38

2.4.1. Discusión de lenguajes CEP. 38

2.4.2. Discusión de Enterprise service bus 40

2.5. Discusión de metodologías 43

2.5.1. Elección de Metodología Ágil. 48

CAPÍTULO 3. Entorno de trabajo 51

3.1. Medios Hardware. 51

3.2. Medios Software. 51

CAPÍTULO 4. Metodología. 55

4.1. Introducción 55

4.2. Metodología Scrum adoptada con prototipado 56

4.3. Plan de trabajo 57

4.4. Análisis de riesgos 58

4.4.1. Posibles riesgos 59

4.4.2. Ponderación 60

CAPÍTULO 5. Desarrollo 63

5.1. Sprint 0. Acercamiento a CEP 63

5.1.1. Reunión de planificación del sprint 63

5.1.1.1 Asuntos que tratar 64

5.1.1.2 Información recogida 64

5.1.1.3 Próximos pasos 65

5.1.2. Desarrollo del sprint 65

5.2. Sprint 1. Inicio del proyecto 67

5.2.1. Reunión de planificación del sprint 67

5.2.1.1 Asuntos que tratar 68

5.2.1.2 Información recogida 68

5.2.2. Planificación del sprint 69

5.2.3. Reuniones de Seguimiento 70

5.2.4. Desarrollo del sprint 71

5.2.4.1 Curso de Energía Eólica. 71

5.2.4.2 Primer estudio de la curva de potencia 74

5.2.4.3 Recogida de requisitos 77

5.2.4.4 Realizar anteproyecto y planificación del Trabajo de Fin de Grado. 80

5.2.4.5 Planificación del Trabajo de fin de grado. 80

5.2.4.6 Acoplamiento e implementación del ESB. 80

5.2.4.7 Sprint Burndown. 81

5.2.5. Reuniones al finalizar el sprint 83

5.2.5.1 Asuntos que tratar 83

5.2.5.2 Información recogida 83

5.2.5.3 Próximos pasos 85

5.2.6. Retrospectiva del Sprint 86

5.3. Sprint 2. Adaptación al caso real 88

5.3.1. Reunión de planificación del sprint 88

5.3.1.1 Asuntos que tratar 89

5.3.1.2 Información recogida 89

5.3.2. Planificación del sprint 90

5.3.3. Reuniones de Seguimiento 92

5.3.4. Desarrollo del sprint 93

5.3.4.1 Patrones en Storyboards. 93

5.3.4.2 Análisis de la curva de potencia y propuesta de patrones. 96

5.3.4.3 Implementación de los patrones. 101

5.3.4.4 Implementación del ESB 108

5.3.4.5 Sprint Burndown 111

5.3.8. Reuniones al finalizar el sprint 112

5.3.5.1 Asuntos que tratar 113

5.3.5.2 Información recogida 113

5.3.5.3 Próximos pasos 114

5.3.9. Retrospectiva del Sprint 115

5.4. Sprint 3. Desarrollo de la plataforma 117

5.4.1. Reunión de planificación del sprint 117

5.4.1.1 Asuntos que tratar 118

5.4.1.2 Información recogida 118

5.4.2. Planificación del sprint 119

5.4.3. Reuniones de Seguimiento 120

5.4.4. Desarrollo del sprint 121

5.4.4.1 Interfaces candidatas. 121

5.4.4.2 Patrones finales. 124

5.4.4.3 Diseño y arquitectura. 129

5.4.4.4 Sprint Burndown 145

5.4.5. Reuniones al finalizar el sprint 146

5.4.5.1 Asuntos que tratar 147

5.4.5.2 Información recogida 147

5.4.5.3 Próximos pasos 148

5.4.6. Retrospectiva del Sprint 149

5.5. Sprint 4. ultimando detalles y pruebas 150

5.5.1. Reunión de planificación del sprint 150

5.5.1.1 Asuntos que trataR 151

5.5.1.2 Información recogida 151

5.5.2. Planificación del sprint 152

5.5.3. Reuniones de Seguimiento 153

5.5.4. Desarrollo del sprint 154

5.5.4.1 Correcciones de Última hora 154

5.5.4.2 Gestión de notificaciones 156

5.5.4.3 Pruebas 162

5.5.4.4 Sprint Burndown 162

5.5.5. Reuniones al finalizar el sprint 163

5.5.5.1 Asuntos que tratar 163

5.5.5.2 Información recogida 163

5.5.5.3 Próximos pasos 164

5.5.6. Retrospectiva del Sprint 164

5.6. Resumen del desarrollo 166

CAPÍTULO 6. Experimentos y resultados 169

CAPÍTULO 7. Conclusiones y propuestas 171

7.1. Conclusiones 171

7.2. Trabajo futuro 171

Bibliografía 175

Anexos 181

A.1.1. Ejemplo de uso de la herramienta 181

A.1.2. Manual de usuario 181

Índice de Ilustraciones

[Ilustración 1. Funcionamiento de SOA a través de publish/suscribe. 24](file:///C:\Users\Kike%20Brazález\git\EolicSimulatorCEP\docs\TFG_CEP_Enrique%20Brazález.docx#_Toc516699981)

[Ilustración 2. Funcionamiento ideal SOA 2.0. 27](file:///C:\Users\Kike%20Brazález\git\EolicSimulatorCEP\docs\TFG_CEP_Enrique%20Brazález.docx#_Toc516699982)

[Ilustración 3. Funcionamiento básico del motor CEP. 31](#_Toc516699983)

[Ilustración 4. Esquema de "Eolic Event Consumer". 31](#_Toc516699984)

[Ilustración 5. Estructura de patrón en Esper EPL. 33](#_Toc516699985)

[Ilustración 6. Ventanas deslizantes y por lotes. 34](#_Toc516699986)

[Ilustración 7. Tablero Kanban. 45](#_Toc516699987)

[Ilustración 8. Principios de Iacovelli. 46](#_Toc516699988)

[Ilustración 9. Metodología SCRUM adaptada. 54](#_Toc516699989)

[Ilustración 10. Diagrama de Gantt de planificación de tiempo. 56](#_Toc516699990)

[Ilustración 11. Ejemplo de curva de potencia. 72](#_Toc516699991)

[Ilustración 12. Formaciones comunes y no comunes de un parque eólico. 74](#_Toc516699992)

[Ilustración 13. Jerarquía de factores. 74](file:///C:\Users\Kike%20Brazález\git\EolicSimulatorCEP\docs\TFG_CEP_Enrique%20Brazález.docx#_Toc516699993)

[Ilustración 14. Infraestructura del entorno de Eolic Event Consumer. 76](#_Toc516699994)

[Ilustración 15. Incorporación de CEP al sistema actual. 77](#_Toc516699995)

[Ilustración 16. Esquema prototipo 1. 79](#_Toc516699996)

[Ilustración 17. Sprint Burndown Inicio del proyecto. 80](#_Toc516699997)

[Ilustración 18. Plantilla de enunciado de patrones. 83](#_Toc516699998)

[Ilustración 19. Introducción de elaboración de Storyboard. 92](#_Toc516699999)

[Ilustración 20. Patrones en formato Storyboard. 93](#_Toc516700000)

[Ilustración 21. Final de la elaboración de los storyboards. 94](#_Toc516700001)

[Ilustración 22. Grafíco de potencia producida el día 1 de enero. 97](#_Toc516700002)

[Ilustración 23. Evento simple WindEvent. 99](#_Toc516700003)

[Ilustración 24. Versión 1.0 de RankingProd. 100](#_Toc516700004)

[Ilustración 25.Versión 1.0 de CambioInicio. 101](#_Toc516700005)

[Ilustración 26. Versión 1.0 de CambioSegundo. 101](#_Toc516700006)

[Ilustración 27. Ejemplo de utilizanción de CambioInicio y CambioSegundo. 102](#_Toc516700007)

[Ilustración 28. Versión 1.0 de Transición. 102](#_Toc516700008)

[Ilustración 29. Versión 1.0 de FluctuaProd. 103](#_Toc516700009)

[Ilustración 30. Diagrama de clases de los patrones de productividad V.1.0. 103](#_Toc516700010)

[Ilustración 31. Versión 1.0 de WindEventT. 104](#_Toc516700011)

[Ilustración 32. Versión 1.0 de BajadaDe10. 104](#_Toc516700012)

[Ilustración 33. Versión 1.0 de WindEventR. 105](#_Toc516700013)

[Ilustración 34. Versión 1.0 de Fuera. 105](#_Toc516700014)

[Ilustración 35. Diagrama de clases de los patrones de rendimiento V.1.0. 106](#_Toc516700015)

[Ilustración 36. Flujo del input V 1.0. 107](#_Toc516700016)

[Ilustración 37. Campana de Gauss de la potencia producida en un aerogenerador 107](#_Toc516700017)

[Ilustración 38. Flujo de adición de patrones V 1.0. 108](#_Toc516700018)

[Ilustración 39. Flujo del despliegue de eventos complejos V 1.0. 109](#_Toc516700019)

[Ilustración 40. Sprint Burndown Adaptación al caso real. 110](#_Toc516700020)

[Ilustración 41. Interfaz candidata de la Productividad. 121](#_Toc516700021)

[Ilustración 42. Interfaz candidata del Rendimiento. 122](#_Toc516700022)

[Ilustración 43. Patrón WindEventP. V 2.0. 123](#_Toc516700023)

[Ilustración 44. Patrón RankinProd V 2.0. 123](#_Toc516700024)

[Ilustración 45. Patrón Escala V 2.0. 124](#_Toc516700025)

[Ilustración 46. Patrón Transición V 2.0. 124](#_Toc516700026)

[Ilustración 47. Diagrama de clases de los patrones de rendimiento V.2.0. 124](#_Toc516700027)

[Ilustración 48. Patrón WindEventTr V 2.0. 125](#_Toc516700028)

[Ilustración 49. Patrón WindEventR V 2.0. 125](#_Toc516700029)

[Ilustración 50. Patrón EstadisticoDiario V 2.0. 125](#_Toc516700030)

[Ilustración 51. Patrón Fuera V 2.0. 126](#_Toc516700031)

[Ilustración 52. Patrón BajadaDeRendimiento V 2.0. 126](#_Toc516700032)

[Ilustración 53. Diagrama de clases de los patrones de Productividad V.2.0 127](#_Toc516700033)

[Ilustración 54. Diagrama de paquetes de EolicEventConsumer. 127](#_Toc516700034)

[Ilustración 55. Flujo del input V 2.0 128](#_Toc516700035)

[Ilustración 56. Diagrama de secuencia "Introducción de Información" 131](#_Toc516700036)

[Ilustración 57. Flujo de adición de patrones V 2.0 131](#_Toc516700037)

[Ilustración 58. Diagrama de secuencia "Añadir un patrón" 133](#_Toc516700038)

[Ilustración 59. Flujo del despliegue de eventos complejos V 2.0 134](#_Toc516700039)

[Ilustración 60. Diagrama de secuencia "Detección & Despliegue". 135](#_Toc516700040)

[Ilustración 61. Diagrama de clases de la aplicación gráfica 137](#_Toc516700041)

[Ilustración 62. Sprint Burndown Desarrollo de la plataforma. 144](#_Toc516700042)

[Ilustración 63. Patrón Escala V 3.0. 154](#_Toc516700043)

[Ilustración 64. Patrón Transición V 3.0 155](#_Toc516700044)

[Ilustración 65. Patrón de la notificación "Fuera". 155](#_Toc516700045)

[Ilustración 66. Patrón de la notificación "Transición". 156](#_Toc516700046)

[Ilustración 67. Patrón de la notificación "BajadaM". 156](#_Toc516700047)

[Ilustración 68. Patrón de la notificación "BajadaD". 157](#_Toc516700048)

[Ilustración 69. Flujo del despliegue de eventos complejos V 3.0 158](#_Toc516700049)

[Ilustración 70. Diagrama de clases de la aplicación gráfica V 2.0. 159](#_Toc516700050)

Índice de Tablas

[Tabla 1. Comparativa de ESBs. 40](#_Toc516696107)

[Tabla 2. Comparativa entre metodología ágil y metodología tradicional. 43](#_Toc516696108)

[Tabla 3. Criterios de elección de metodologías ágiles de Iacovelli. 46](#_Toc516696109)

[Tabla 4. Resumen de la elección de metodologías ágiles por Iacovelli. 48](#_Toc516696110)

[Tabla 5. Métrica del PMBok de la Gestión de Riesgos. 58](#_Toc516696111)

[Tabla 6. Ponderación de riesgos. 60](#_Toc516696112)

[Tabla 7. Información base de sprint 0. 61](#_Toc516696113)

[Tabla 8. Resumen de desarrollo sprint 0. 64](#_Toc516696114)

[Tabla 9. Información base del sprint 1. 65](#_Toc516696115)

[Tabla 10. Resumen de desarrollo sprint 1. 67](#_Toc516696116)

[Tabla 11. Reuniones semanales del sprint 1. 68](#_Toc516696117)

[Tabla 12.Conclusiones de curso introductorio de energía eólica. 70](#_Toc516696118)

[Tabla 13. Información base del sprint 2. 86](#_Toc516696119)

[Tabla 14. Resumen de desarrollo sprint 2. 89](#_Toc516696120)

[Tabla 15. Reuniones semanales del sprint 2. 90](#_Toc516696121)

[Tabla 16. Factores enfrentados. 94](#_Toc516696122)

[Tabla 17. Mediciones del día 1 y 2 de enero. Rendimiento. 96](#_Toc516696123)

[Tabla 18. Mediciones del día 1 y 2 de enero. Productividad. 98](#_Toc516696124)

[Tabla 19. Información base del sprint 3. 115](#_Toc516696125)

[Tabla 20. Resumen de desarrollo sprint 3. 117](#_Toc516696126)

[Tabla 21. Reuniones del sprint 3. 119](#_Toc516696127)

[Tabla 22. Información base del sprint 4. 148](#_Toc516696128)

[Tabla 23. Resumen de desarrollo sprint 4. 150](#_Toc516696129)

[Tabla 24. Reuniones semanales del sprint 4. 151](#_Toc516696130)

[Tabla 25. Historia de usuario “Correcciones del Product Owner”. 153](#_Toc516696131)

[Tabla 26. Tipos de notificaciones. 159](#_Toc516696132)

[Tabla 27. Resumen de horas estimadas. 164](#_Toc516696133)

[Tabla 28. Resumen de horas estimadas y reales. 165](#_Toc516696134)

2. Introducción

“Internet of Things” (IoT), o “Internet de las cosas” en español, es el tema que ha inspirado este proyecto. Cómo poder comunicar todos los elementos que influyen en nuestro día a día puede parecer poco viable, debido a la gran inversión y trabajo que se necesita. No obstante, cada día que pasa dicho esfuerzo es menor [1]. Así, la evolución del IoT cambiará nuestro futuro. Nuestra forma de percibir la realidad cambiará totalmente y no sólo en nuestra vida cotidiana. si no que irá mucho más allá afectando a ámbitos como la educación, la comunicación, las empresas, la ciencia, y el gobierno. En este trabajo de fin de grado nos centraremos en el ámbito científico e industrial, en concreto el que afecta a las energías renovables.

* 1. Motivación

Este proyecto nace de la necesidad de optimizar la obtención de energía a un bajo coste y teniendo en cuenta la preservación del medio ambiente. Las fuentes de energía no renovables son aquellas que se encuentran de forma limitada en nuestro planeta dado que su consumo es mayor que su “regeneración” [2].

La integración de los sistemas de información en el sector industrial está al orden del día, es decir, la industria está vitalmente unida a los sistemas software en su enorme mayoría. Si la tecnología decae, cualquier actividad profesional relacionada está destinada al fracaso como es el caso de la generación de energía [3]. Existen multitud de aplicaciones software para controlar las herramientas de extracción de energía por medios renovables. Por ejemplo, en placas solares móviles podemos observar un sistema para ajustar la orientación con respecto al sol y optimizar su producción y, por otro lado, en el sistema de control de capacidad en un embalse podemos decidir cuándo nos conviene aprovechar la energía producida por su caudal.

A lo que energía renovable se refiere, hemos de destacar la energía eólica y la implicación que tienen los aerogeneradores en la misma. La integración del software y sistemas electrónicos en todos sus componentes tanto a nivel individual, así como, a nivel de parque eólico es abrumador. Un ejemplo destacable sería el flujo de información que se produce entre los aerogeneradores y las estaciones centrales. Sin embargo, todo ese volumen de información no se aprovecha nada más que para labores muy sencillas de mantenimiento y monitorización.

El objetivo de este proyecto es ir un paso más allá y en base a este gran volumen de información predecir el desgaste de un aerogenerador. Como dijo Bill Gates: “La información es poder”. Llevando esta frase a la práctica. Si tenemos toda la infraestructura de sensorización, ¿por qué no aprovechar esos grandes volúmenes de datos no solamente para labores de control y monitorización sino también para realizar predicciones?

* 1. Objetivos

Fruto de la motivación para la elaboración de este proyecto surgen los siguientes objetivos, aunque hemos de destacar en concreto el principal: “Implementar un sistema experto que nos permita predecir los eventos relacionados con el desgaste de un aerogenerador en base a los datos obtenidos por sus sistemas de monitorización”.

Para considerar como satisfecho este objetivo hemos de alcanzar las siguientes metas a, que serían:

1. Estudio de energía eólica y familiarizarnos con el entorno de los aerogeneradores, tanto desde un punto de vista técnico, como desde uno más general.
2. Controlar el “Enterprise Service Bus” de Mule, para poder analizar el flujo de datos.
3. Familiarizarnos con el lenguaje para implementar los patrones en Esper EPL.
4. Detectar qué eventos son de interés en las medidas dadas en un aerogenerador para implementar los patrones adecuados.
5. Desarrollo de una aplicación gráfica que nos permita visualizar todos los eventos simples, y complejos fruto de los patrones implementados.
6. Generar alertas en base a los patrones implementados.
   1. Estructura de la memoria

El esqueleto interno de esta memoria por capítulos es el siguiente:

**CAPÍTULO 1. Introducción**

Se trata del capítulo donde nos encontramos, y en el que se han desarrollado temas que sirven como introducción para el correcto entendimiento de los conceptos de este proyecto.

**CAPÍTULO 2. Estado del arte**

En esta sección veremos reflejados todos y cada uno de los fundamentos teóricos en los que se basa “Eolic Event Consumer”. Se definirá el funcionamiento del Procesamiento de Eventos Complejos y la energía eólica. Terminando con el estado del arte, donde se analizan las tecnologías existentes, revelando sus ventajas y desventajas, en busca de la mejor o la más viable para la solución del problema.

**CAPÍTULO 3. ENTORNO DE TRABAJO**

En este capítulo veremos qué elementos hardware y software hemos empleado, para el desarrollo de la aplicación, “Eolic Event Consumer”.

**CAPÍTULO 4. Metodología.**

Una vez presentado el Capítulo 3, donde se ponía de manifiesto una discusión de las metodologías existentes. En este punto describiremos con detalle la elegida, esquematizando los artefactos a entregar, quién ocupa qué roles, así como un Diagrama de Gantt del primer planteamiento del proyecto.

**CAPÍTULO 5. Desarrollo**

En este capítulo veremos reflejado el trabajo realizado sprint a sprint, con toda la información detallada, desde varios diagramas UML hasta lo recogido en las reuniones con el cliente y el equipo.

**CAPÍTULO 6. Experimentos y resultados**

En este capítulo quedan reflejadas las pruebas llevadas a cabo en el sistema, siendo contrastadas con el modelo de calidad ISO-IEC 9126. Describiendo algunos de los aspectos visuales y técnicos de la aplicación gráfica, y de la aplicación CEP que trabaja por debajo.

**CAPÍTULO 7. Conclusiones y propuestas**

Por último, se expondrán una serie de conclusiones, culminando con el planteamiento de futuros trabajos que pueden realizarse utilizando como punto de partida “Eolic Event Consumer”.

2. Estado del arte

Hoy en día la incorporación del análisis de datos en los sistemas informáticos es un elemento indispensable para su éxito. Con el transcurso del tiempo, el anhelo de conocimiento de ingenieros y tecnólogos produce que la evolución en la industria crezca cada día mas. Ultimamente, estos avances están relacionados con el desarrollo de sistemas que pretenden optimizar la productividad y eficiencia de cualquier actividad, ya bien a sea en nuestros hogares, a través de domótica, o a nivel comercial, a través del análisis de la tendencia en el mercado utilizando técnicas de Machine Learning.

Ese incremento en la productividad y eficiencia, con respecto la competencia, puede significar que nos posicionemos como pioneros en nuestra actividad profesional. Y esto también tiene su lugar en el contexto que nos encontramos, la energía eólica. Optimizar la producción energía frente al resto de multinacionales puede significar que nos coloquemos como cabeza en la lucha por ser líderes de producción energética renovable.

La incorporación de Internet of Things en los parques eólicos puede significar un cambio en el modo de producir energía eléctrica. Los sistemas de monitorización de los aerogeneradores producen una ingesta cantidad de información, de ahí que actualmente se intente utilizar toda esa cantidad de datos, para hacer análisis predictivo y optimizar la producción de energía.

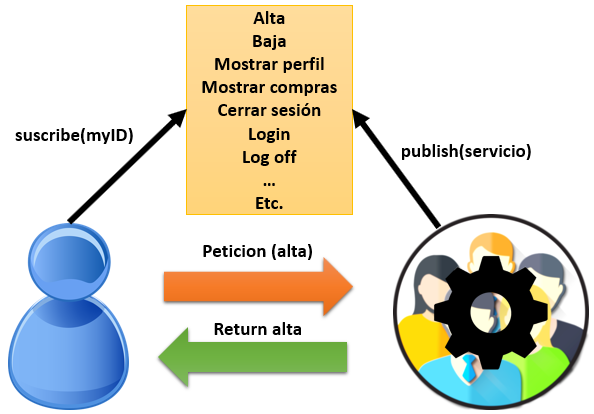
Cada vez son más las productoras de energía renovables como Endesa y Acciona, que integran en sus aerogeneradores estos sistemas predictivos, para poder optimizar su actividad. Cabe destacar en este campo a uno de los fabricantes más famosos de aerogeneradores, Gamesa, que el 29 de octubre de 2008[4], que consolidó una tecnología propia para el mantenimiento predictivo mediante el análisis de vibraciones a través del uso de acelerómetros para detectar el desgaste en los elementos mecánicos del aerogenerador. De ahí que más adelante estudiemos la implantación del procesamiento de los eventos complejos, en una arquitectura orientada a servicios, pero dirigida por eventos.

Volviendo al hilo principal, dividimos este capítulo en dos. Por un lado, tenemos la definición y desarrollo de los fundamentos teóricos de los que nace “Eolic Event Consumer”, como herramienta para el mantenimiento predictivo, y, por otro lado, un análisis comparativo de las posibles soluciones al problema planteado.

* 1. Fundamentos teóricos.

En este punto veremos todos y cada de los puntos teóricos que rodean este trabajo de fin de grado, tanto a nivel de arquitectura, como de software.

* + 1. Arquitectura orientada a Servicios (SOA)

La arquitectura orientada a servicios es un estándar basado en componentes que relacionan entre si diferentes funcionalidades de un nodo, estas funcionalidades se denominan “servicios”[5], por medio de tecnologías e interfaces que facilitan la interacción con el usuario.

Un servicio es un elemento reutilizable que puede ser parte o todo de un proceso de negocio, a mayor o menor escala. Un servicio puede ser una simple petición GET a un servidor, para obtener el saldo que nos queda en nuestro banco. Además, puede servir como parte en un sistema completo de logeo en una página web. Hemos de añadir que puede que ese componente del sistema de logeo, por sí solo no aporta ninguna funcionalidad, pero en conjunto con más servicios sí que lo hace. Tal y como vemos en la Ilustración 1, normalmente las arquitecturas orientadas a servicios se implantan a través de protocolos publish/suscribe, donde el usuario se suscribe al proveedor de servicios para solicitar esas funcionalidades.

Ilustración 1. Funcionamiento de SOA a través de publish/suscribe.

Una parte imprescindible para el buen funcionamiento de una arquitectura orientada a servicios es la interfaz, ha de definirse de forma neutral, independiente de lo que trabaja por debajo y proporciona el servicio.

La arquitectura orientada a servicios [6] permite que los componentes estén débilmente acoplados, reutilizando los servicios entre sí, optimizando su funcionamiento y compartiendo los componentes internos de la arquitectura con todos los usuarios. Las características que facilitan este débil acoplamiento son:

* Sigue estándares abiertos que permiten la fácil compatibilidad.
* Aseguran la compatibilidad con multitud de tecnologías.
* Separa el código de la interacción del usuario a través de interfaces.
* Modular. Puede ejecutarse como parte o todo de un proceso de negocio, y pueden ser remplazables.
* Sin estado. El servicio no mantiene el estado entre invocaciones, bajo la petición de un servicio se toma el parámetro introducido y se invoca aisladamente la función correspondiente.
* Comunicación. Permite la comunicación entre nodos distintos que proveen de servicios externos.
* Ubicación. El usuario descnoce la ubicación real de la máquina que ofrece el servicio.
  + 1. Arquitectura dirigida por eventos (EDA).

Según dos de los pioneros en lo que al procesamiento de eventos complejos se refiere, David Luckham y W. Roy Schulte, un evento es un suceso que está ha llamado a suceder, [7] y que tiene una relevancia en un contexto concreto. Por ejemplo, la notificación cada diez minutos de un sensor de temperatura, o la llegada de un correo electrónico a nuestra bandeja de entrada. Por tanto, si aplicamos su definición de evento para describir EDA, podemos decir que es un tipo de arquitectura cuyos componentes son los eventos, dirigidos y comunicados por ellos mismos.

Este patrón arquitectural permite la producción de eventos, así como su visualización y análisis [8]. Tenemos cuatro elementos que son claves para el funcionamiento de este tipo de arquitectura:

* **Productor de eventos.** Es aquel componente que tiene como misión generar los eventos a raíz de lo que ocurre en su entorno. Si lo aplicamos a “Eolic Event Consumer”, diremos que los productores de eventos son los aerogeneradores, mandando cada diez minutos un registro con todo lo que ha captado su sistema de monitorización.
* **Consumidores de eventos.** Un componente que consume la información generada por el productor para efectuar la toma de decisiones. En el caso de este trabajo de fin de grado, podríamos decir que es el motor inteligente que procesará toda esa información.
* **Canal.** Es el medio por el que se transmiten los eventos.
* **Acciones.** Se trata de las acciones reales que se llevan a cabo, fruto del análisis realizado por el consumidor de eventos.
  + 1. Arquitectura orientada a servicios dirigido por eventos (SOA 2.0)

El procesamiento de eventos complejos tiene lugar en una arquitectura orientada a servicios, pero dirigido por eventos. Esta surge como la evolución de SOA utilizando el mecanismo de EDA [9], y contiene tanto las características de uno como las del otro. En el punto 2.1.5, explicaremos con detenimiento el concepto del Procesamiento de Eventos Complejos.

El aspecto más relavante de este tipo de arquitectura, es que los servicios son distribuidos en base al flujo de eventos que circulan, y no como en SOA, en una sencilla arquitectura cliente-servidor. A través de este tipo de arquitecturas, se permite la construcción de aplicaciones con gran capacidad de respuesta, puesto que son más adaptables a entornos impredecibles [10]. Para el correcto funcionamiento de la integración de SOA y EDA, hace falta una capa homogénea que permita la comunicación entre los productores y los consumidores de eventos. Se denomina, “Enterprise Service Bus” (ESB), una capa de integración para poder encaminar el flujo de eventos y mensajes entre los diferentes actores de la arquitectura. En otras palabras, podríamos decir que se trata de una capa de middleware, que permite la comunicación y la administración de datos en todas las aplicaciones distribuidas del sistema.

En la Ilustración 2, podemos ver la arquitectura básica de una arquitectura SOA 2.0 donde, la vía común para todos los productores para comunicarse con los consumidores es a través del ESB. Los productores de eventos pueden ser desde el sistema de monitorización de un molino de viento o de una casa, enviando mediciones de velocidad de viento y temperatura, siendo muy sencillo el evento producido. Un ejemplo de aplicación de CEP en SOA 2.0, podría ser, la de controlar la temperatura en todas las habitaciones de nuestra casa. ¿Cómo controlar esa temperatura en las diferentes habitaciones? A través de la elaboración de patrones que detecten esos eventos simples, produciendo una alerta en forma de evento complejo. El enunciado de este patrón podría ser, suponiendo que hubiera un termómetro en cada habitación: “Cuando todos los termómetros hayan mandado un evento simple con temperatura mayor a 20 grados centígrados, se produzca ese evento complejo, cuya acción asociada haga que la calefacción se pare”.

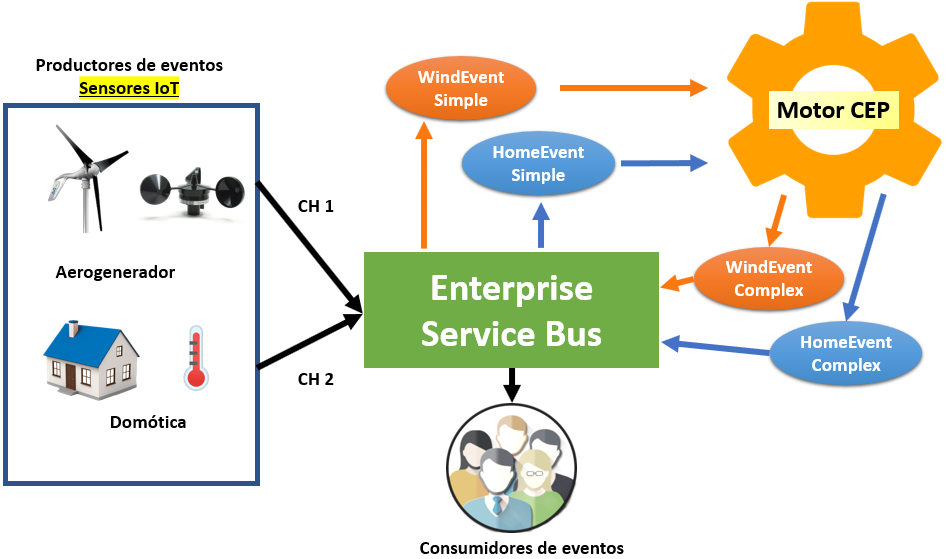


Ilustración 2. Funcionamiento ideal SOA 2.0.

Volviendo al hilo principal, aunque hayamos adelantado la noción de patrón, hemos de tener muy claro, que la consecución de eventos de un tipo no tiene como consecuencia sólo la llamada a una función en un servidor para llevar a cabo una acción, si no que va más allá, y puede generarse a su vez, otro evento que, junto a otros, sí que puede ser de interés. El objetivo de este tipo de arquitecturas es optimizar y agilizar la toma de decisiones, en base a la información obtenida a tiempo real y en tiempo de ejecución.

* + 2. Procesamiento de eventos complejos (CEP)

Dando respuesta a la pregunta lanzada en la motivación: “¿por qué no aprovechar esos grandes volúmenes de datos no solamente para labores de control y monitorización sino también para realizar predicciones?

Surge como posible solución el Procesamiento de Eventos Complejos (tecnología CEP) como aplicación de “Fast Data”. La mejor forma de tomar decisiones en estos ámbitos es hacerlo cuanto antes, de ahí que se examine toda la información a tiempo real para de esta forma dirigir satisfactoriamente la toma de decisiones o acciones correctivas en el ámbito industrial. Hay que saber distinguir que se habla de “Fast Data” y no de “Big Data” [11]. ¿Por qué? Fast Data tiene una ventaja, y es que no almacena ningún tipo de información. Conforme los datos son generados, una vez procesados por primera vez, se desechan, es decir, se analizan conforme van siendo generados para después desecharlos, lo que lo hace muy rentable. Si se almacenara hablaríamos de “Big Data”. Por lo tanto, como se genera, se procesa, y se desecha sin almacenarla hablamos de “Fast Data”, evitándonos el gran coste que llevaría salvaguardar tanta información.

Partiendo de que un evento complejo es un elemento que resume, representa o denota un conjunto de otros eventos diremos que CEP, tal y como Fast Data determina, es que puede dominar cualquier volumen de datos, a una velocidad en tiempo real y de orígenes muy diversos. CEP, es una tecnología emergente capaz de monitorizar múltiples flujos de eventos, procesarlos mediante la determinación de patrones implementados en Esper Event Processing Language (EPL) [12], y así tomar decisiones tanto predictivas como correctivas en base a la información que el contexto nos aporta.

Como conclusión a la problemática generada por la necesidad de optimizar la producción en un parque eólico, este trabajo de fin de grado tiene como meta la predicción del desgaste de aerogeneradores a través de un sistema SOA 2.0. SOA 2.0, es una arquitectura orientada a servicios, aunque dirigida por eventos [13], en la sección 2.1.4 se ha explicado su funcionamiento.

Cabe destacar que CEP [14] tiene varios aspectos comúnes con una arquitectura basada en servicios, donde el usuario sabe cual es el servicio para solicitar, sin embargo, en el contexto de CEP, son los productores de eventos los que manda las peticiones a los consumidores (usuarios) para que procesen toda esa información. Podemos decir que la arquitectura donde se implantan se implanta la tecnología basada en el procesamiento de eventos complejos es reactiva (bajo la ocurrencia de situaciones de interés) y desacoplada (al poder haber más de un único productor de eventos).

Debido a las pequeñas diferencias y similitudes con SOA, podemos concluir que SOA 2.0 nace como fruto de la combinación con una arquitectura dirigida por eventos, EDA, ajustándonos a ese funcionamiento reactivo y desacoplado.

Si analizamos el funcionamiento de CEP [15], podemos enumerar diferentes características como son:

* **Comunicación múltiple**. Por lo general los productores de eventos envían la información que es de interés a cada uno de los consumidores, que se han suscrito a ese productor.
* **Inmediatez.** La comunicación de los eventos es a tiempo real en el momento en el que se detecta alguna situación de interés.
* **Asincronismo.** El sistema que publica un evento no espera a que los sistemas que lo reciben lo procesen. Publica y continúa.
* **Eventos de grano fino.** Todo el análisis surge a raíz de eventos simples con datos de grano fino, pudiendo así, simplificar el flujo de información y facilitar el análisis y la toma de decisiones en base a la información obtenida. Cuanto más se simplifique el flujo de eventos, mejor
* **Nomenclatura y categorización.** Se pueden realizar jerarquías de eventos clasificándolos en base a su relevancia u otros criterios.
* **Procesamiento de Eventos Complejos.** El sistema interpretará y monitorizará las relaciones entre eventos. Por ejemplo, la agregación de eventos (un patrón de eventos implica un evento de mayor nivel) o relación de causalidad (un evento es originado por un evento previo).
  + - 1. Tipos de procesamiento

Por lo general hay tres tipos de procesamiento de eventos[16], todos y cada uno de ellos son complementarios entre sí, son los siguientes:

1. **Procesamiento de eventos simple.** El sistema se nutre de los eventos de más bajo nivel, iniciando acciones de muy baja transcendencia. Normalmente se utiliza para impulsar en tiempo real la información, disminuyendo el tiempo de demora a la hora de iniciar procesos paralelos.
2. **Procesamiento de eventos en streaming**. Suceden eventos comunes y notables. Los eventos ordinarios (pedidos, transmisiones de RFID) se analizan en busca de notoriedad y se transmiten a los consumidores de eventos. Se usa comúnmente para impulsar el flujo de información en tiempo real dentro y alrededor de la empresa agilizando la toma de decisiones.
3. **Procesamiento de eventos complejos.** Este es el tipo de procesamiento que trataremos en este trabajo de fin de grado. CEP se usa comúnmente para detectar y responder a anomalías, amenazas y oportunidades comerciales.
   1. Diagrama de funcionamiento

El funcionamiento del motor CEP, es sencillo. Una vez que se hayan implementado los patrones. Los añadiremos a la cola de patrones desplegados dentro del motor CEP, tal y como vemos en la Ilustración 3. Una vez arrancada nuestra aplicación, los productores de eventos irán mandando lo captado a través del ESB, para que el motor CEP analice los datos. Podemos decir que, hay tres etapas principales dentro de la situación normal de una aplicación de CEP en SOA 2.0, son las siguientes:

* **Producción.** Etapa de medición del entorno real, así como la normalización de los datos para que sea material legible para el motor CEP y su envío al ESB.
* **Detección y análisis.** La información transformada (si fuera necesario) llega al entramado inteligente de la aplicación de tal forma que se coteja toda la recibida con los patrones de eventos desplegados. Es importante destacar, que normalmente en tiempo de ejecución, el motor CEP es capaz de desplegar nuevos patrones para la detección de eventos complejos.
* **Respuesta y despliegue.** Una vez detectados los eventos complejos, se comienza la toma de decisiones, y se emprenden las acciones estipuladas.

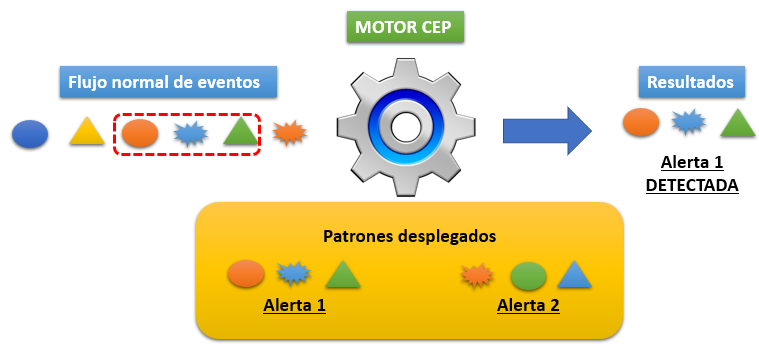


Ilustración 3. Funcionamiento básico del motor CEP.

Habiendo visto, cómo funcionan los motores CEP, y cómo se integra SOA 2.0 con el procesamiento de eventos complejos, podemos adelantar un primer esquema en la Ilustración 4, de la solución que “Eolic Event Consumer” proporciona al problema.

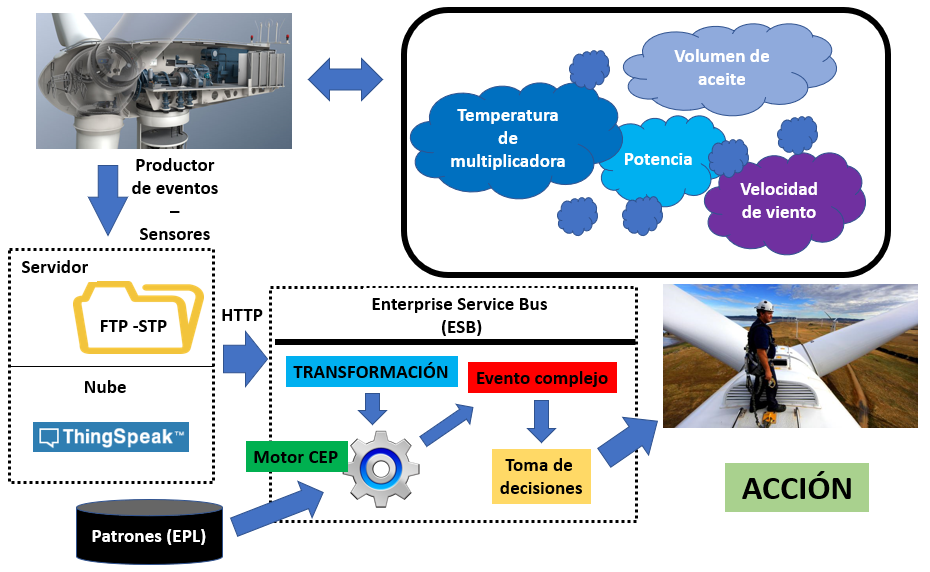


Ilustración 4. Esquema de "Eolic Event Consumer".

Como vemos, el origen de los datos proviene de los múltiples sensores fluyendo vía HTTP para transmitirlos al motor CEP. El motor CEP, procesa la información utilizando los patrones implementados. De esta forma, desplegaremos los eventos complejos en una página web aplicando medidas correctivas y preventivas dependiendo del evento complejo que se detecte en el flujo de datos.

* 1. ESPER EPL. Lenguaje de programación para CEP.

Para poder implementar esos patrones para la detección de eventos complejos usaremos Esper EPL. EPL, es el acrónimo de “Event Processing Language”, es un leguaje de programación declarativa con gran escalabilidad, con memoria eficiente, computación en tiempo de ejecución, SQL estándar, mínima latencia, transmisión en tiempo real con capacidad de motor de grandes volúmenes de datos y de gran variedad, permitiendo además el análisis de históricos. Un patrón de eventos, ciñéndonos a lo publicado por David Luckham y W. Roy Schulte [7], es un esquema de operadores relacionales y variables, intentando reflejar una situación real mediante su morfologia. Un patrón de evento puede hacer coincidir conjuntos de eventos relacionados al reemplazar variables con valores.

Esper ofrece un lenguaje específico de dominio (DSL) para el procesamiento de eventos. El lenguaje de procesamiento de eventos (EPL) es un lenguaje declarativo para tratar datos de eventos basados ​​en el tiempo de alta frecuencia. EPL cumple con el estándar SQL-92 y se amplía para analizar gran cantidad de eventos y su relación en el tiempo [17].

Además, Esper proporciona en su página web, una aplicación en línea para probar EPL, la URL es esta: <http://esper-epl-tryout.appspot.com/epltryout/index.html>

Esper ofrece un lenguaje de patrones de eventos para especificar coincidencias de patrones de eventos basados ​​en expresiones. Detrás del motor de coincidencia de patrones está la implementación de una máquina de estado. Este método de procesamiento de eventos coincide con las secuencias esperadas de presencia, ausencia, o combinaciones de eventos.

Esper también ofrece consultas de series de eventos que abordan los requisitos de análisis de series de eventos de las aplicaciones de CEP. Las consultas de series de eventos proporcionan las funciones de ventanas, agregación, unión y análisis para su uso con flujos de eventos. Estas consultas siguen la sintaxis de EPL. EPL ha sido diseñado para similitud con el lenguaje de consulta SQL, pero difiere de SQL en el uso de vistas en lugar de tablas. Las vistas representan las diferentes operaciones necesarias para estructurar datos en una serie de eventos.

* + 1. Sintaxis

Además, hemos de destacar que Esper EPL[18], sigue el esquema sintáctico de Ilustración 5, sin embargo, tiene incluido las librerías de java.lang, java.math, java.text, y java. util, que hacen a este lenguaje SQL sea aún mucho más interesante para el análisis de datos.

[@Name (“Pattern’s Name”)]

[expression\_declarations]

[context contextname]

[into table table\_name]

[insert into insert\_into\_def]

select select\_list

from stream\_def [as name] [, stream\_def [as name]] [,...]

[where search\_conditions]

[group by grouping\_expression\_list]

[having grouping\_search\_conditions]

[output output\_specification]

[order by order\_by\_expression\_list]

[limit num\_rows]

Ilustración 5. Estructura de patrón en Esper EPL.

Cada una de las sentencias de la Ilustración 5, tiene una función, y son las siguientes:

* **@Name (“XXX”).** Se añade el nombre del patrón a implementar, es opcional, sin embargo, es esencial para legibilidad del código.
* **INSERT INTO.** La creación de un flujo de eventos complejos, identificados por el nombre introducido en esta parte de la sentencia SQL.
* **SELECT**. Recoge las propiedades de los eventos simples/complejos del flujo capturado.
* **FROM**. Flujo de evento seleccionado.
* **FROM PATTERN.** Lugar donde se añaden las condiciones del patrón de los eventos del flujo seleccionado.
* **WHERE.** Clausula que se encarga de hacer un filtrado al flujo de eventos seleccionado en la clausula FROM.
* **GROUP BY**. Agrupaciones.
* **HAVING.** Clausula para el filtrado de los elementos agrupados.
* **ORDER BY.** Ordenación de los eventos por el criterio que se seleccione.

Los elementos por los que Esper EPL se distingue de un lenguaje SQL estándar, son los operadores de patrón, y el uso de ventanas temporales, que permiten capturar el tráfico de eventos en un flujo durante un tiempo determinado para su análisis. Partiendo de aquí, definiremos cuales son los elementos más útiles, y de los que nacen la mayoría de los códigos en Esper EPL. Por un lado, tenemos las ventanas de datos, que retienen los eventos entrantes hasta que una política de caducidad indique que se liberarán los eventos. Por lo tanto, las ventanas de datos son un medio para indicar qué subconjunto de eventos queremos analizar. Un ejemplo de las más representativas son las siguientes:

1. **WIN:TIME\_BATCH(N SECONDS)**. Ventana por lotes que obtiene todos los elementos retenidos el tiempo especificado.
2. **WIN:LENGTH\_BATCH(N)**. Ventana que extrae el número de eventos especificado como parámetro.
3. **WIN:TIME(N SECONDS)**. Se declara una ventana deslizante de un período de tiempo especificado.
4. **WIN:LENGTH(N)**. Ventana deslizante vinculada al número de eventos.

Como podemos ver, hay dos tipos de ventanas temporales. Las ventanas que van por lotes, y las deslizantes. Las ventanas deslizantes capturan tantos eventos, como haya en X período de tiempo desde que se encuentra el primero, y, por otro lado, tenemos las ventanas que van por lotes, que separan el flujo en intervalos de X períodos de tiempo capturando todos los que nazcan en ese intervalo.

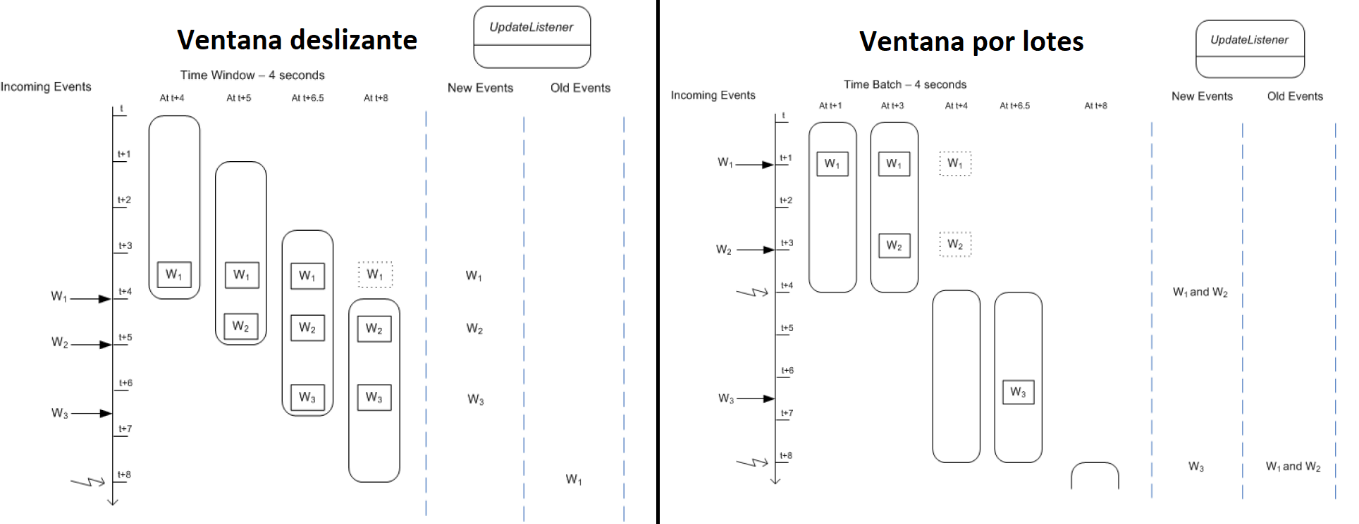


Ilustración 6. Ventanas deslizantes y por lotes.

Para verlo de una forma más gráfica, podemos ver en la Ilustración 6, como la ventana deslizante, captura desde que encuentra el primero todos los que hay en un período de tiempo X, y la que va por lotes, captura todos los eventos que surgen en intervalos de 4 segundos. Lo “negativo” de las ventanas por lotes, es que podemos hacer una de 4 segundos, y puede que del segundo 8 al 12 no surja ningún evento, de manera que se devolvería una ventana vacía.

Finalmente, tenemos los operadores de patrón, elementos que estudian a través de operadores lógicos y aritméticos el flujo de eventos de eventos capturados, y los temporizadores, para relacionarlos con el momento con el que fueron desplegados. Los elementos más destacables de los operadores de patrón son:

1. **EVERY**. Selecciona todos los eventos del tipo que se especifique.
2. **EVERY-DISTINCT**.Es exactamente igual, sin embargo, elimina las instancias repetidas.
3. **FOLLOW BY (“->”)**. Para especificar, el patrón que esté a la izquierda debe ser seguido por un segundo.
4. **UNTIL**. Se comprueba un patrón hasta que se cumpla.
5. **WHILE**. Se comprueba un patrón mientras se cumpla.

Para entender los operadores de patrón correctamente, Esper, sugiere varios ejemplos que pueden ser de interés, por ejemplo, suponiendo el siguiente flujo de eventos:

**A1, B1, C1, B2, A2, D1, A3, B3, E1, A4, F1, B4**

Si hacemos un filtrado utilizando los operadores de patrón, “every” y “follow by” podemos obtener los siguientes resulados según el orden en el que los escribamos:

1. Every (A -> B)

**Resultado I.** {A1, B1}, {A2, B3}, {A4, B4}

1. Every A -> B

**Resultado II.** {A1, B1}, {A2, B3}, {A3, B3}, {A4, B4}

1. A -> every B

**Reusltado III.** {A1, B1}, {A1, B2} {A1, B3} {A1, B4}

1. Every A -> every B

**Resultado IV.** {A1, B1}, {A1, B2}, {A1, B3}, {A2, B3}, {A3, B3}, {A1, B4}, {A2, B4}, {A3, B4}, {A4, B4}

Como hemos comentado anteriormente, el último elemento que hemos de destacar es el temporizador, para vincular los operadores de patrón con el tiempo en el que nacieron.

* **TIMER:INTERVAL(N SECONDS)**. Espera el tiempo especificado antes de que el valor se evalue a verdadero.
* **TIMER:AT(\*,\*,\*,\*,\*,\*)**. Retornará verdadero cuando llegue el momento indicado.
* **TIMER:WITHIN (N SECONDS)**. Se da falso cuando la condición del patron no se ha cumplido en un tiempo estipulado.
  1. Discusión tecnológica

En este punto veremos dos comparativas tecnológicas. Por un lado, la discusión de los lenguajes para procesar eventos complejos y el porqué elegimos Esper EPL, y, por otro lado, tenemos la comparativa de los diferentes Enterprise Service Bus del mercado.

* + 1. Discusión de lenguajes CEP.

Algunos lenguajes que manejan el procesamiento de eventos complejos son Esper EPL, Continuos Query Language (CQL) de Oracle, StreamSQL y Continous Computation Languaje (CLL).

En el caso de CQL de Oracle[19], es un lenguaje de consulta basado en SQL para el procesamiento de eventos complejos, sólo compatible con el software Oracle CEP. Oracle CEP es un servidor de Java para desarrollar aplicaciones dirigidas por eventos. Proporciona un entorno abundante y declarativo basado en Oracle SQL, con librerías de java, al igual que Esper EPL. Oracle CEP tiene un rendimiento alto y una latencia de microsegundos utilizando JRockit Real Time. Proporciona Oracle CEP Visualizer y Oracle CEP IDE, entornos para el desarrollo de una aplicación basada en EDA implementada sobre Java. Sin embargo, no nos hemos decantado por esta opción debido a dos factores:

* Nuestra experiencia con los últimos IDEs de Oracle como SQLDeveloper. Los entornos se quedan congelados por fallos en su implementación, no pudiendo así explotar al 100% su EPL.
* Coste. Al contrario que Esper EPL que es gratuito.

Stream SQL es el EPL que es acoplado en SQLstream s-Server desarrollado por SQLStream[20]. Está basado en SQL y su servidor está diseñado para satisfacer las necesidades de baja latencia, alto volumen y rápida integración de las empresas actuales en tiempo real. SQLstream s-Server procesa transacciones de diversas fuentes de forma continua, proporcionando análisis de transmisión y salida de datos a múltiples destinos. Este software es gratuito los primeros 60 días. A partir de ahí, solo podrá procesar un 1 GB/ día. El coste [21] de esta herramienta en máquinas de Azure, Amazon, Docker, y Linux oscila entre 80 € como mínimo, a como máximo unos 1465 € mensuales. También existe la posibilidad de hacerlo por consumo, es decir pagando $ 6.60 la hora. Esta opción queda descartada por los costes y por lo costoso que es utilizar su documentación.

Al igual que Stream SQL, Esper EPL y CQL, el lenguaje también se basa en SQL. Continuous Computation Language (CCL), es un lenguaje unicamente integrable en Sybase CEP, está desarrollado por SAP. Proporciona las mismas utilidades que el resto. Cabe destacar su precio, [22] que es de $ 775.48, y que no hay versión gratuita ni ninguna trial.

Elegimos Esper EPL por las siguientes razones:

* Totalmente gratuito, una solución de Open Source.
* Esper EPL proporciona un servicio web para probar de forma fácil y sencilla los patrones implementados, simulando el funcionamiento normal del motor CEP.
* Todo el lenguaje queda recogido como un simple “jar”, lo que lo hace muy manejable y se puede amoldar a cualquier ESB.
* El soporte de Esper es muy bueno, la documentación técnica de la sintáxis es muy didáctica y fácil de aprender, todo con muchos ejemplos permitiendo la fácil comprensión del funcionamiento tanto del lenguaje como de la tecnología basada en el procesamiento de eventos complejos.
* He recibido un curso por Juan Boubeta Puig, profesor titular en la Universidad de Cádiz, donde tuvimos un primer acercamiento a CEP. Al utilizar Esper EPL en el curso, y aprender de una forma tutorizada me siento más seguro utilizando esta opción.
  + 1. Discusión de Enterprise service bus

En este punto veremos veremos los ESB más representativos, y finalmente haremos un análisis comparativo eligiendo el que más se amolda a nuestras necesidades [23].

* **W2SO2 ESB**

A nivel de prestaciones es la mejor alternativa que tenemos frente al resto. WS02 permite ir añadiendo funcionalides en un proceso repetitivo, ahorrando tiempo y dinero.

Tiene un alto rendimiento y se integra fácilmente con cualquier sistema tradicional, se integra con el fin de aumentar el control de los diseñadores derivando algunas de las funciones del sistema propio a este ESB (transformaciones, empaquetamientos, etc), aumentando la productividad y minimizando los costes. El entorno gráfico se basa en Eclipse. Un ejemplo de este ESB es el instaurado en Ebay. Es uno de los ESB más relevantes debido a que existe una solución completa de Open Source, sin embargo, deja de serlo si lo pasamos a producción.

* **Mule ESB**

Mule destaca por su fácil instalación, su fácil puesta en marcha, y por la capacidad de integrar todo tipo de tecnologías, desde scripts de Groovy, Ruby, Java, Python, etc. Ofrece dos versiones, una solución de open source y otra solución comercial que permite mayores prestaciones.

Cabe destacar que la solución de Open Source no tiene ningún coste si se lanza a producción. Podemos utilizar esta versión de forma gratuita hasta en la fase de producción, sin embargo, si tenemos propósitos comerciales deberemos de pagar por la licencia “Enterprise”.

* **JBoss Fuse**

JBoss Fuse [24] es una plataforma de integración open source basada en Apache ServiceMix. Está basada en SOA y proporciona herramientas para integrar diferentes componentes de aplicaciones.

En junio de 2012, Red Hat anunció que había adquirido FuseSource de Progress Software Corporation. A continuación, Fuse ESB se renombró como JBoss Fuse, y desde entonces disponible en el portal de Red Hat.

JBoss Fuse es compatible con JBI y OSGi para su uso en organizaciones de TI empresariales. Cualquier motor de servicio o componente de enlace compatible con JBI o OSGi estándar (incluidos los motores BPEL, XSLT o JMX) puede implementarse en un contenedor JBoss Fuse y los componentes JBoss Fuse pueden desplegarse en otros ESB.

* **Adroit Logic UltraESB**

Es un ESB [25] integrable en IDEs como son IntelliJ IDEA, Eclipse y NetBeans. Permite que lo mensajes sean procesados a través de Java, o JSR 223 a través de una API. La versión inicial se publicó en enero de 2010, y el código fue abierto posteriormente con licencia pública general Affero. Las aplicaciones de administración y monitoreo son:

* Uconsole: La consola de gestión y supervisión basada en web.
* IMonitor: Se ejecuta como una aplicación web independiente y permite administrar y supervisar los nodos del ESB.
* **JBoss ESB**

Se trata de una distribución independiente, aunque se suele combinar con JBoss Fuse. Es compatible con cualquier servidor de aplicaciones JEE, debido a que es muy fácil de instalar. De cualquier forma, el ESB de JBoss, no tiene las mejores prestaciones frente a sus competidores, puesto que se creó con el fin de completar algunas de las carencias de JBoss Fuse.

Si nos fijamos en la Tabla 1, podremos ver un cuadro comparativo entre los diferentes Enterpise Service Bus que hemos visto [26]. Esta comparativa nos servirá como herramienta para elegir el ESB que más se ajuste a nuestras necesidades. Puede que nuestra elección no sea la mejor, pero sí la que se ajuste a lo que necesitamos para llevar este proyecto a cabo, en base a los recursos y conocimientos de programación de los que disponemos.

Tabla 1. Comparativa de ESBs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Carácterísticas | WSO2 ESB and SOA Platform | [Mule ESB](http://www.mulesoft.com/mule-esb-open-source-esb) | JBoss Fuse | [Adroit Logic UltraESB](http://www.adroitlogic.org/) | [JBoss ESB and SOA Platform](http://www.redhat.com/products/jbossenterprisemiddleware/soa/) |
| Soporta Enterprise Integration Patterns (EIP) | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png |
| Ofrece todas las funciones de un ESB (i.e. servicios web, transformación de mensajes, protocolo de mediación, enrutiamiento de contenido) |  | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png |
| Ofrece una plataforma SOA completa y cohesiva (es decir, ESB, Message Broker, Governance Registry, Business Process Server, Data Services Server, Application Server) | Resultado de imagen de tick verde png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png | Resultado de imagen de tick verde png |
| Gobernabilidad SOA | Resultado de imagen de tick verde png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png |
| Workbench gráfico del ESB | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick rojo png | Resultado de imagen de tick verde png |
| Oferta de plataforma integrada en Cloud (iPaaS) | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png |
| Conectores y adaptadores Cloud | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick rojo png | Resultado de imagen de tick rojo png | Resultado de imagen de tick verde png |
| Rendimiento | High | Moderate | Moderate | High | Moderate |
| Seguridad y gestión de identidad | Resultado de imagen de tick verde png | Limited | Limited | Limited | Limited |
| Modelo de negocio abierto | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick verde png | Resultado de imagen de tick rojo png |

Tras analizar y comparar los Enterprise Service Bus hemos decidido que la opción más asequible es Mule ESB por las siguientes razones:

* Es compatible con máquinas virtuales que adopten CEP a través de la librería de Esper EPL.
* Proporciona gran capacidad para la transformación de mensajes en diferentes formatos (.db, csv, excell, json, etc).
* Proporciona gran cantidad de conectores de comunicación con aplicaciones vía http, bases de datos relacionales (MySQL) y no relacionales (MongoDB), SMTP, ajax, amazon services, FTP, y muchos más.
* La solución de open source es gratis y soporta la carga de trabajo de un parque eólico.
* Se puede establecer en un servidor de forma sencilla.
* AnyPoint Studio es muy fácil de aprender.
* La comunidad y los foros de MuleSoft son muy activos de forma que hay muchísimas fuentes de documentación muy completas.
* Gran escabilidad puesto que se pueden combinar varios ESBs en paralela de Mule.
* Si se quieren más prestaciones, siempre cabe la posibilidad de pagar por la versión Enterprise.
* Teníamos conocimiento previo del uso de Mule ESB en un curso acerca del Procesamiento de Eventos Complejos impartido, por lo que no nos supuso un reto tan complicado aprender a utilizarlo.
  1. Discusión de metodologías

La eterna batalla entre las metodologías ágiles y las tradicionales tiene cabida en este trabajo de fin de grado también. La elección de la metodología para desarrollar un sistema software no es una decisión sencilla, debemos de barajar todos y cada uno de los factores que influyen directa e indirectamente en el proyecto. Antes de elegir una, debemos de definir cuales son las bases tanto de las metodologías tradicionales y las metodologías ágiles.

Piattini en 1996 [27], definía el concepto de metodología, como el “conjunto de pasos y procedimientos que deben seguirse para el desarrollo de software”. Aunque el concepto no está claro, es evidente que el uso de una metodología de trabajo garantiza la sistematización del desarrollo. Otro de los pioneros en ingeniería del software, Maddison, en 1983, definía metodología, como el conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores. De manera que una metodología debe dictaminar:

* La estructura en etapas del proyecto.
* Tareas de cada etapa.
* Artefactos que tienen lugar tras la realización de las tareas.
* Roles y deberes.
* Qué restricciones existen.
* Cuáles son las mejores herramientas que se pueden aplicar, y cuales son las que nosotros podemos aplicar.
* Cómo gestionar y controlar un proyecto.

Dejado esto claro, enfrentamos dos tipos de metodologías:

1. **Metodología tradicional.**

Las metodologías tradicionales se centran en el buen trabajo del proceso de desarrollo del software, con el objetivo de obtener un software eficiente [28]. Para esto se hace hincapié en la planificación completa del trabajo que se ha de llevar a cabo, comenzando así el ciclo de desarrollo del software. Primero planificamos todas las tareas, y a continuación comenzamos a desarrollar. Se centra como he dicho al comienzo de este punto en el control del proceso, fijando desde un primer momento, los roles, tareas, artefactos, etapas y modelado. Algunos ejemplos de metodología tradicional pueden ser RUP, MSF, Win-Win Spiral Model e Iconix.

1. **Metodología ágil.**

Los procesos ágiles son una buena elección cuando se trabaja con requisitos desconocidos o variables. Si no existen requisitos estables, no existe una gran posibilidad de tener un diseño estable y de seguir un proceso totalmente planificado, que no vaya a variar ni en tiempo ni en dinero. En estas situaciones, un proceso adaptativo será mucho más efectivo que un proceso predictivo. Por otra parte, los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previas a la entrega final. Algunos ejemplos de metodología son XP, Scrum, Crystal Clear, DSDM, FDD, XBreed, y Extreme Modeling.

Si comparamos estas dos metodologías podemos sacar las conclusiones extraídas en la Tabla 2 [29].

Tabla 2. Comparativa entre metodología ágil y metodología tradicional.

|  |  |
| --- | --- |
| Metodologías ágiles | Metodologías tradicionales |
| Desarrollos cortos (menos tiempo) | Desarrollos medio-largos (más tiempo) |
| Impuestas internamente, las decisiones parten del equipo. | Impuestas externamente, dependen más del cliente. |
| Proceso menos controlado debido a que se rige por menos principios. | El proceso es controlado por muchos sectores, con multitud de normas y directrices. |
| No existe un contrato tradicional, en principio. | Existe un contrato prefijado. |
| Muy flexible a la hora de incluir cambios. | No tan flexible, ya que incluir un cambio es más lento. Resistencia al cambio. |
| El cliente es parte del equipo de desarrollo, continuamente en contacto con él. | El cliente interactúa con el equipo de desarrollo únicamente a través de reuniones. |
| Grupos pequeños, no más de una decena de personas, trabajando cerca geográficamente. | Grupos grandes con posibilidades de que se encuentren lejos entre sí. |
| Pocos artefactos. | Más artefactos. |
| Pocos roles. | Más roles. |
| Menos hincapié en la arquitectura del software. | La arquitectura del software es esencial y se expresa mediante modelos. |
| Desarrollo incremental e integración continua. | Modelo de desarrollo en cascada. |
| Menos caro. | Más caro. |

Hecha esta comparación entre las dos modalidades de metodologías existentes, hemos elegido una metodología ágil, debido a:

* Poco tiempo para desarollar el sistema.
* El cliente debe involucrarse activamente con el desarrollo del proyecto debido al desconocimiento del Team en materia de energías renovables.
* El criterio de aceptación debe ser flexible, ya que además de las preferencias del cliente también tiene que ser regido y evaluado por un tribunal académico.
* Desde un primer momento se sabe que se quiere hacer una aplicación CEP, sin embargo, en ningún momento se sabe qué patrones se van a implementar ni qué va a encuadrar el sistema, por tanto, el resultado final no puede regirse por ningún contrato.
* Debe ser flexible al cambio.
* No hay tiempo para hacer toda la documentación que una metodología tradicional requeriría.
* No hay ningún tipo de fondo económico para desarrollar el sistema.

Una vez hecha la elección del tipo de metodología a utilizar, pasamos a ver cuáles son las metodologías ágiles que existen para poder seleccionar la que mejor se adapta a nuestras necesidades. Las metodologías ágiles que estamos planteándonos utilizar, en base a las que más se usan comúnmente son las siguientes:

1. **SCRUM.**

SCRUM [30] fue propuesto a principios de los 90 por Jeff Sutherland y Ken Schwaber. La base principal del éxito de SCRUM es la descentralización de los equipos, puesto que son autoorganizados. Los requisitos funcionales que se describían en las metodologías tradicionales, se descomponen en historias de usuario, y estas a su vez, en tareas.

Dividen las funcionalidades en tareas, y estas tareas deberán de estar resueltas al terminar un período de tiempo determinado, denominado “sprint”. El ciclo de vida del proyecto será dividido en sprints. Los roles dentro de una metodología SCRUM son:

* Scrum Master. Jefe de proyecto.
* Product Owner. Cliente.
* Team. Equipo de desarrollo.

1. **Kanban.**

Kanban [31] surgió como uno de los componentes claves en el Sistema de Producción Toyota, origen a su vez de Lean and Just In Time.

En Kanban todos los requisitos funcionales estarían planteados como tareas en un flujo de trabajo: el trabajo se divide en artículos (tareas) que jugarían en un tablero, el tabler tiene varias columnas que representan las etapas de este flujo de trabajo. Un ejemplo del tablero la podemos ver en Ilustración 7.



Ilustración 7. Tablero Kanban.

Es importante destacar que el número de artículos están en progreso son finitos, depende del número de personas en el equipo. El tiempo promedio que se tarda en completar un artículo, también conocido como tiempo de ciclo, es analizado y optimizado para que el proceso sea eficiente y eficaz. La eliminación de artículos considerados como basura, o desechos es primordial (elementos que no aportan ni valor, ni funcionalidad).

1. **Extreme Programming.**

[eXtreme Programming](http://enjoylife.com.ar/novedades/que-es-la-programacion-extrema-xp/) [32] (XP), ante todo busca ofrecer la máxima calidad posible en el software desarrollado a través de técnicas de análisis, desarrollo y testeo adecuadas. Cada miembro del equipo de XP participa activamente para la planificación de la iteración, al contrario que el resto de las metodologías, donde sí que hay una persona encargada de coordinar al equipo, aquí no. Las iteraciones son muy cortas bajo demanda del cliente, a partir de varias prácticas que son:

* Test Driven Development. Desarrollo dirigido por los tests.
* Customer Testing. Las pruebas funcionales con el cliente son básicas para el éxito del desarrollo del sistema.
* Integración continua.
* Pequeños lanzamientos.
* Programación de pares. Se programa en grupos de dos personas, ambas personas se nutren como programadores, aportan, se ayudan y optimizan la solución.
  + 1. Elección de Metodología Ágil.

Vistas tres de las metodologías más utilizadas hemos de decantarnos por la que más se ajusta a nuestras necesidades. Para ello [33], existen frameworks que en base a nuestras preferencias nos permiten elegir la metodología ideal. Un ejemplo es Adrian Iacovelli, autor de “Framework for Agile Methods Classification”. En su publicación se establece una clasificación basada en cuatro principios, definidos en la Ilustración 8.

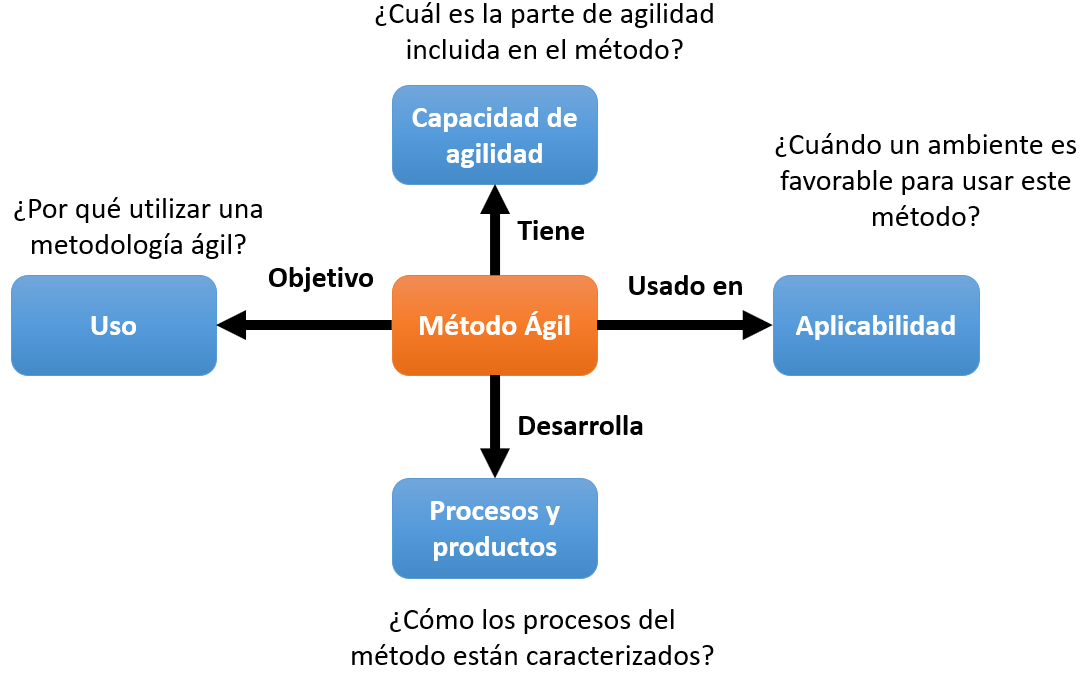


Ilustración 8. Principios de Iacovelli.

En la Tabla 3, quedan reflejadas las diferencias entre las metodologías ágiles definidas anteriormente. Como vemos, en color verde quedan señaladas aquellas similitudes con nuestro proyecto y en base a su resultado elegiremos una metodología u otra.

Tabla 3. Criterios de elección de metodologías ágiles de Iacovelli.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Orientada al desarrollo** | **Orientada a la gestión de proyectos** | |
| ***XP*** | ***SCRUM*** | ***KANBAN*** |
| **USO** | Respeto de las fechas de entrega. | F | V | F |
| Cumplimiento de requisitos. | V | V | V |
| Respeto al nivel de calidad. | F | F | F |
| Satisfacción del usuario final. | F | V | F |
| Satisfacción del usuario final. | V | V | V |
| Entornos turbulentos. | F | V | F |
| Aumento de la productividad. | V | V | V |
|  | | | | |
| **CAPACIDAD DE AGILIDAD** | Iteraciones cortas. | V | V | V |
| Colaboración. | V | V | V |
| Centrado en las personas. | V | V | V |
| Refactoring político. | V | F | F |
| Prueba política. | V | V | F |
| Integración de los cambios. | V | V | V |
| De peso ligero. | V | V | V |
| Lo RF pueden cambiar. | V | V | V |
| Los NO RF pueden cambiar. | F | F | V |
| El plan de trabajo puede cambiar | V | F | V |
| Los recursos humanos pueden cambiar. | V | F | V |
| Cambiar los indicadores | V | F | F |
| Intercambio de conocimientos. | ALTO | BAJO | BAJO |
|  | | | | |
| **APLICABILIDAD** | Tamaño del proyecto. | PEQUEÑO | PEQUEÑO  GRANDE | PEQUEÑO |
| La complejidad del proyecto. | BAJA | ALTA | BAJA |
| Los riesgos del proyecto. | BAJA | ALTO | BAJO |
| El tamaño del equipo. | PEQUEÑO | PEQUEÑO | PEQUEÑO |
| Grado de interacción con el cliente. | ALTA | ALTA | BAJA |
| Grado de interacción con los usuarios. | BAJO | ALTA | BAJO |
| Grado de interacción entre el equipo. | ALTA | ALTA | BAJA |
| Grado de integración de la novedad. | ALTA | ALTA | BAJA |
| La organización del equipo. | AUTO | AUTO | AUTO |
|  | | | | |
| **PROCESOS Y PRODUCTOS** | **Nivel de abstracción de las normas y directrices** | | | |
| Gestión de proyectos. | F | V | F |
| Normas y orientaciones concretas sobre las actividades y productos. | V | F | F |
| Descripción de procesos. | V | F | F |
| **Las actividades cubiertas por el método ágil** | | | |
| Puesta en marcha del proyecto. | F | F | F |
| Definición de requisitos. | V | V | F |
| Modelado. | V | V | V |
| Código. | V | V | V |
| Pruebas unitarias. | V | V | V |
| Pruebas de integración. | V | V | V |
| Prueba del sistema | V | V | V |
| Prueba de aceptación | F | F | F |
| Control de calidad | F | F | F |
| Sistema de uso | F | F | F |
| **Productos de las actividades del método ágil.** | | | |
| Modelos de diseño. | F | V | F |
| Comentario del código fuente. | V | V | V |
| Ejecutable. | V | V | V |
| Pruebas unitarias. | V | V | V |
| Pruebas de integración. | V | V | V |
| Pruebas de sistema. | V | F | V |
| Pruebas de aceptación. | F | F | F |
| Informes de calidad. | F | F | F |
| Documentación de usuario. | F | F | F |

El resumen de Tabla 3, queda reflejado en Tabla 4. Viendo los resultados podemos decir que la mejor metodología a utilizar para este proyecto según Iacovelli, es SCRUM. Hecho este estudio acerca de las metodologías existentes, coincidimos con Iacovelli y emprenderemos SCRUM con unas pequeñas modificaciones que las explicaremos más adelante en el Capítulo 4

Tabla 4. Resumen de la elección de metodologías ágiles por Iacovelli.

|  |  |  |
| --- | --- | --- |
| Metodologías | Positivos | Negativos |
| XP | 34 | 17 |
| SCRUM | 49 | 2 |
| Kanban | 36 | 15 |

Con esto, concluimos el capítulo relacionado con el estado del arte, y a continuación pasamos a concretar el entorno de trabajo.

1. Entorno de trabajo

En este apartado hablaremos de todos los medios tanto software como hardware tenemos pensado utilizar para llevar a cabo el proyecto en primera instancia.

* 1. Medios Hardware.

Los medios hardware que utilizaremos para llevar a cabo el proyecto serán los siguientes:

* **Ordenador personal.** Se trata de un ordenador ASUS bastante común, utilizado durante el transcurso de la carrera. Procesador i7-5610QM CPU de 2,30 GHz y memoria RAM de 6 GB, con un tipo de sistema de 64 bits y con un sistema operativo Windows 10.
* **PLC de aerogenerador.** Se trata del PLC que controla todas las mediciones de las centralitas PT100 que nutren el sistema de monitorización del aerogenerador. Esta será la fuente de datos del sistema, aunque en primer lugar trataremos un conjunto de datos sintético en formato Excell o .db con el que simularemos el flujo de información, debido a que no tenemos acceso directo a ese PLC.
  1. Medios Software.

En esta sección veremos cuáles son los elementos software que tomarán participación en nuestro entorno de trabajo. Desde los programas, IDEs y demás, hasta las librerías particulares que hacen singular este proyecto.

* **Eclipse.** Eclipse es un entorno de desarrollo integrado de Java (IDE), con gran cantidad de plugins de Open Source con lenguajes integrados como C / C++ y PHP. Puede combinar fácilmente el soporte de idiomas y otras funciones en cualquiera de los paquetes que la nube de Eclipse facilita en los paquetes predeterminados, y Eclipse Marketplace permite una personalización y extensión prácticamente ilimitadas. Hemos elegido Eclipse y no otro entorno porque el ESB está implementado sobre Eclipse, de esta forma podemos unificar paralelamente todo el código, tanto el del simulador como el del Enterprise Service Bus [34].
* **EGit**. EGit es un proveedor de Eclipse Team para el sistema de control de versiones de Git. Git es un SCM distribuido, lo que significa que cada desarrollador tiene una copia completa de todo el historial de cada revisión del código, lo que hace que las consultas contra el historial sean muy rápidas y versátiles. El proyecto EGit está implementando herramientas Eclipse además de la implementación JGit Java de Git [35].
* **Team Foundation Service.** Se trata de un producto Software que nos permitirá llevar a cabo la gestión del trabajo. En concreto la enumeración de historias de usuario, de tareas, y generación de artefactos como sería el Sprint Burndown.
* **Anypoint Studio, ESB de Mulesoft**. Es el entorno de desarrollo de integración basado en Eclipse de MuleSoft para diseñar y probar aplicaciones Mule. Puede implementar la aplicación y ejecutarla en su servidor Mule. El mismo editor también le permite editar archivos de definición de API (en RAML y WSDL) y crear dominios que definan recursos compartidos. Anypoint Studio ofrece dos pestañas paralelas que puede utilizar para diseñar y crear sus aplicaciones: Editor visual; y un editor XML [36]**.**
* **Librería poi-3.11 para el tratamiento de documentos Excel.** Se trata de una librería de Apache para la manipulación de documentos Excel que en un principio se generarían por el PLC del aerogenerador y nutrirían al motor CEP.
* **Librería de Esper EPL.** Esper ofrece un lenguaje específico de dominio (DSL) para el procesamiento de eventos. El lenguaje de procesamiento de eventos (EPL) es un lenguaje declarativo para tratar datos de eventos basados ​​en el tiempo de alta frecuencia. EPL cumple con el estándar SQL-92 y se amplía para analizar series de eventos y con respecto al tiempo [17].
* **CakePHP.** Es un framework libre [37], y de código abierto, para el desarrollo rápido de aplicaciones en PHP, tiene las siguientes prestaciones:
  + Compatible con PHP 5.2.6
  + Contiene CRUD para interaccionar con la base de datos.
  + Generación automática de código a través de consola.
  + Arquitectura MVC.
  + Plantillas disponibles y gran comunidad de ayuda.
  + Ayudantes para AJAX, JavaScript, formularios HTML y más.
  + Componentes de Email, Cookie, Seguridad Sesión y otros.
  + Trabaja con composer por debajo.
* **XAMPP**. Es un paquete de Apache [38] que destaca por su fácil instalación ya que contiene MariaDB, PHP y Perl. Además, tiene una comunidad de más de 10 años, lo que ayudará en caso de tener alguna duda a la hora integrarlo en nuestro sistema.
* **Balsamiqs**. Es un software que nos permitirá bocetar con un mayor nivel el portal web que implementaremos en las etapas finales del proyecto.

1. Metodología.

En el comienzo de este capítulo veremos la organización del proyecto, cuáles son los roles de los participantes, un diagrama de Gantt orientativo, así como una breve descripción de este, y a continuación el documento de gestión de riesgos que se ha generado al inicio del proyecto. Por último, en este capítulo explicaremos que peculiaridades tiene la elección de la metodología elegida, así como la descripción de los hitos y artefactos que se han generado.

* 1. Introducción

El desarrollo del proyecto desde el mes de enero cuenta con 23 semanas. En cuanto a la metodología de trabajo hemos de destacar que se usará una metodología ágil, en concreto será SCRUM, combinándolo con prototipado.

Se basará en una metodología SCRUM donde los roles están bastante bien identificados, sin embargo, el Team, solo tiene un integrante y no se van a hacer reuniones diarias para ver cual es la situación actual del proyecto. Por lo tanto, no podremos decir que es un SCRUM “puro”, sino “adaptado”. Los integrantes del grupo ajustando los roles dentro del equipo son:

* **Jefe de proyecto.** Gregorio Diaz Descalzo. Profesor dentro de la Politécnica de Albacete, en la Escuela de Ingenieros Informáticos. Coordinador entre Team y Product Owner para que haya buena comunicación y no haya ningún problema a la hora de establecer los requisitos y compromisos para y con el proyecto tanto a nivel de organización como a nivel técnico.
* **Product Owner.** Francisco José Polo Sánchez. Integrante de la empresa Ingeteam. Cliente en cuestión dentro del desarrollo del proyecto, solicita el sistema y proveerá de infraestructura de trabajo al Team. Una vez hecha la recogida y elicitación de requisitos, será el encargado en aprobarlos. En caso de haber algún error, se hallará un acuerdo y se tomarán las medidas pertinentes para su aprobación.
* **Team.** Enrique Brazález Segovia. Estudiante de 4º de Ingeniería Informática, responsable de defender el trabajo de fin de grado.
  1. Metodología Scrum adoptada con prototipado

La metodología que iremos trabajando a lo largo de todo el proyecto es SCRUM con prototipado, en la Ilustración 9 podemos ver un esquema de cómo se organiza, así como de las diferentes reuniones y artefactos que se generan.

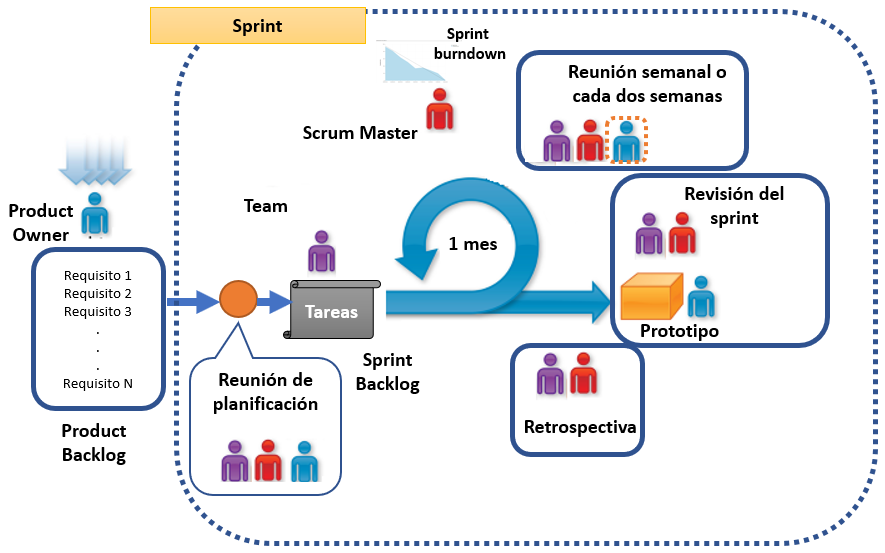


Ilustración 9. Metodología SCRUM adaptada.

Nuestro plan de trabajo se va a dividir en varias iteraciones (sprints), y al acabar cada iteración tendremos un producto software operativo y funcionando. Con el paso de las iteraciones, el producto software gozará de más funcionalidades, ajustándose a la solución final del problema.

El proyecto comienza como una idea o visión proporcionada por el Product Owner, encargado de enunciar cuales son las necesidades que quiere que el sistema cubra. Estas necesidades y funcionalidades son definidas como requisitos y se ordenan por prioridad en una pila denominada **“Product Backlog”.**

El Product Owner se reunirá con el Team y el SCRUM Master para presentar esa pila de requisitos, esta reunión es denominada como **“Reunión de planificación del sprint”**. A continuación, Scrum Master y Team, eligirán un subconjunto de requisitos traduciéndolos en objetivos y funcionalidades que tendrá el prototipo de esa iteración.

Una vez hecha esa presentación y enunciado de objetivos, comienza el sprint. Posteriormente el Team, planifica el sprint, creando una serie de tareas que se recogerán en el denominado **“Sprint Backlog”**. Estas tareas estarán estimadas en horas, de manera que, nos podríamos hacer una idea mirando al **“Sprint backlog”** del desarrollo del prototipo.

Cada una de estas tareas están organizadas por historias de usuario, es decir, cada hito a cumplir sería una historia de usuario. Siempre y cuando todas las tareas de una historia de usuario estén cerradas, podremos dar por cubierto ese objetivo. Durante el proyecto el Team deberá de anotar día a día las horas que realiza de cada tarea, para llevar un control del tiempo que le tiene que dedicar al trabajo restante. El Team deberá de anotar las horas que va realizando día a día para ver el tiempo estimado que tiene que dedicar para desarrollar el prototipo de esa iteración. Como fruto de esta consumición de horas se genera el Sprint Burndown, gráfico que define en el eje X la línea temporal del sprint y en el eje Y el número total de horas que quedan para terminarlo.

A lo largo del desarrollo de los sprint, se harán diferentes reuniones de seguimiento del trabajo cada semana o cada dos, para poner en conocimiento todos los avances del proyecto a los involucrados, y en caso necesario, para que el Team pregunte dudas tanto al Scrum Master como al Product Owner.

Al finalizar el Sprint, tendremos dos reuniones:

* **Reunión de revisión del sprint**. El equipo muestra al Product Owner, y al Scrum Master, el desarrollo realizado durante el sprint. Esta reunión servirá para poder verificar que vamos en el camino correcto.
* **Reunión de retrospectiva del sprint**. Es organizada con el equipo a continuación de la de revisión, y antes de la de planificación del siguiente sprint. En esta reunión nos preguntamos a nosotros mismos qué ha ido bien y mal en este sprint, y ser más efectivos en la siguiente.
  1. Plan de trabajo

Tal y como hemos comentado anteriormente, la metodología va a estar entrelazada con el prototipado, de forma que tras acabar cada iteración en el proyecto tendremos un producto operativo y funcionando. La organización del trabajo tal y como queda esquematizado en el diagrama de Gantt de la Ilustración 10 es la siguiente:

* **Inicio del proyecto**. En esta iteración se planteará el proyecto y se interiorizarán todos los aspectos técnicos acerca del ESB, la energía eólica, etc. Además, plantearemos la planificación del proyecto, realizaremos una primera versión de los requisitos a alcanzar, y se elaborará un primer prototipo del flujo de datos usando un intermediario vía Web como es “Thingspeak”. Esta aplicación externa servirá de intermediario entre el generador de eventos y el motor CEP.
* **Adaptación al caso real.** A través del ESB generaremos otro prototipo que no dependa de” Thingspeak” y empezaremos a implementar de forma real todos y cada uno de los patrones ajustándonos al modelo de aerogenerador concreto. Insertaremos las alertas obtenidas de los eventos complejos en una base de datos MySQL.
* **Desarrollo de la plataforma.** Pasaremos a la elaboración de la aplicación gráfica que servirá como pantalla de visualización de los eventos detectados a través del framework CakePHP.
* **Tiempo de margen para la memoria y flecos sueltos.** Es una pequeña iteración que servirá como tiempo de reserva por si surge algún imprevisto que retrase el proyecto.
* **Presentación y defensa del trabajo de fin de grado.**



Ilustración 10. Diagrama de Gantt de planificación de tiempo.

* 1. Análisis de riesgos

Partiendo de las nociones en la quinta edición del PMBOK [39],un riesgo es un problema potencial que puede o no ocurrir, y suele afectar a futuros acontecimientos. Los riesgos implican cambios, y debemos estudiar pese a que mi rol no sea el de “Scrum Manager”, qué implicaciones tiene sobre nuestro trabajo la aplicación de dichos cambios. Además, los riesgos nos hacen enfrentarnos a diversas elecciones, como escoger las herramientas a emplear, la estructuración de nuestro código, modificaciones en el tipo de licencias que manejamos o a importancia que debemos de darle a la calidad de sistemas software en dicho instante.

Este plan consiste en una serie de pasos que ayudan al equipo de software a comprender y gestionar la incertidumbre en la toma de decisiones tanto a nivel organizativo como de desarrollo.

* + 1. Posibles riesgos

Hay multitud de riesgos que pueden darse, sin embargo, he querido destacar solo los que he visto más relevantes:

1. **Problemas con la comprensión de EPL.** Podría ocurrir, que, una vez hecho el estudio del contexto, y analizados cuáles son los fenómenos eólicos que se pueden medir a través de patrones, no sepamos implementarlos. Puede que la sintáxis nos de muchos problemas y que no entendamos el paradigma de la programación orientada por eventos.
2. **Problemas con la programación con CakePHP.** Podría ocurrir, que, debido a que no tenemos ninguna noción de PHP, y ni mucho menos del framework de CakePHP, la implementación de la aplicación gráfica se retrase o que ni si quiera se lance. Es un riesgo que se ha de afrontar debido a que el sistema interno del cliente está implementado de esta forma.
3. **El cliente no tenga detectados qué fenómenos son interesantes detectar.** Debido a la complejidad de hallar patrones, puede caber la posibilidad de que el cliente no tenga asociadas las mediciones anómalas a un tipo de avería concreta. En esta situación seremos nosotros los que nos encarguemos de enunciar los patrones, y que estos sean validados por el cliente.
4. **El cliente no se interese por el proyecto.** Puede que la empresa no tenga conocimiento del funcionamiento de las metodologías ágiles, y no se implique en el proyecto tanto como debiera. Puede caber la posibilidad de que nuestro coordinador con la empresa cliente, se desvincule y perdamos ese primer contacto con Ingeteam.
5. **Enfermedad del Team.** Podría ocurrir, que, el Team enferme, por lo que irremediablemente se retrase la entrega del proyecto a Julio, o incluso a septiembre.
6. **Cambiar de una base de datos relacional a otra que no lo es.** Podría ocurrir, que, debido a la cantidad ingente de datos con las que trabajaremos, el modelo de dato cambie. Por tanto, no podremos cuadrar las alarmas en una base de datos tipo MySQL, y nos tendremos que pasar a MongoDB.
   * 1. Ponderación

En la Tabla 5, podemos observar en las filas, la probabilidad de que ocurra ese riesgo, y en las columnas las consecuencias que suponen para el proyecto:

Tabla 5. Métrica del PMBok de la Gestión de Riesgos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Gestión de riesgos en “EolicEventConsumer” | INSIGNIFICANTE  (1) | MENOR  (2) | MODERADA  (3) | MAYOR  (4) | CATASTRÓFICA  (5) |
| RARO (1) | BAJO | BAJO | MODERADO | ALTO | ALTO |
| IMPROBABLE (2) | BAJO | BAJO | MODERADO | ALTO | EXTREMO |
| POSIBLE (3) | BAJO | MODERADO | ALTO | EXTREMO | EXTREMO |
| PROBABLE (4) | MODERADO | ALTO | ALTO | EXTREMO | EXTREMO |
| CASI SEGURO (5) | ALTO | ALTO | EXTREMO | EXTREMO | EXTREMO |

* **Bajo.** Los riesgos bajos deben ser objeto de seguimiento por parte únicamente de un miembro del Team.
* **Moderado.** Los riesgos moderados deben ser objeto de seguimiento adecuado por parte tanto del Enrique, como de Fernando.
* **Alto.** Los riesgos altos requieren la atención del Scrum Manager y del Team.
* **Extremo.** Los riesgos extremos han de comunicarse al cliente directamente después de a ver sido planteados por el Scrum Manager y el Team.

En el caso de “Eolic Event Consumer”, podemos clasificar estos riesgos, según la siguiente categoría:

1. **Problemas con la comprensión de EPL.** Tiene una probabilidad probable (4) de que ocurra, ya que enfrentarnos a un lenguaje que trabaja con una lógica distinta a la tradicionalmente puede suponer problemas. Las consecuencias de dicho riesgo podemos clasificarlas como menores (3), puesto que, pese a que desconocemos el lenguaje, acudiremos a un curso de iniciación, y tendremos contacto con profesores que conocen el lenguaje.
2. **Problemas con la programación con CakePHP.** Tiene una probabilidad casi segura (5), puesto que el Team no ha tocado nunca PHP, ni mucho menos un framework tan avanzado como es CakePHP. Tampoco está familiarizado con el servidor Apache, y MySQL, tecnologías imprescindibles para instaurar un servicio web que se amolde al de nuestro cliente. Las consecuencias de este riesgo podemos verlas como catastróficas (5) puesto que, si este riesgo se convierte en realidad, retrasaremos el despliegue mucho tiempo e incluso se tome la decisión de instaurar otra tecnología directamente.
3. **El cliente no tenga detectados, aunque sea en lenguaje formal los patrones.** Este riesgo podemos categorizarlo con probabilidad posible (3), ya que el cliente tiene muy claras cuales pueden ser las señales de un comportamiento anómalo en el aerogenerador, sin embargo, dudamos en sí es cierto. Las consecuencias de este riesgo son moderadas (3), ya que con nuestra ayuda solo será cuestión de tiempo redactar formalmente estos eventos de interés.
4. **El cliente no se interese por el proyecto.** Este riesgo tiene una probabilidad posible (3), debido a que no hay ninguna inversión económica. Este factor puede hacer que el cliente no le de ninguna prioridad a este proyecto. Sin embargo, las consecuencias que conllevaría la desvinculación con el cliente serían mucho mayores (4), debido a que, sin el enunciado de los patrones, ni con expertos de energía eólica no podremos implantar un sistema que detecte esas máquinas con comportamiento irregular.
5. **Enfermedad del Team.** Se trata de un riesgo que puede ser clasificado con una probabilidad posible (3), debido a que cualquier persona del equipo puede padecer una enfermedad. Debido a que es un trabajo de fin de grado en el que estaremos a jornada completa trabajando en él, en un período de tiempo muy corto las consecuencias de este riesgo son moderadas (3), pudiendo llegar a ser castróficas.
6. **Cambiar de una base de datos relacional a otra que no lo es.** Si clasificamos este riesgo diremos que tiene una probabilidad posible (3) de que suceda, ya que la escalabilidad en un sistema de Big Data es primordial, y puede que al trabajar con distintos modelos de datos se tenga que utilizar una base de datos no relacional tipo MongoDB. Las consecuencias de este riesgo son menores (2), significaría una modificación a nivel de estructura del código, por tanto, un incremento del número de tareas, y a su vez de tiempo.

Si resumimos este análisis de riesgos obtendríamos los resultados de la Tabla 6, donde se refleja de forma gráfica que nos encontramos ante riesgos muy peligrosos. Puesto que el sistema ha de desarrollarse en menos de 5 meses, la mayor parte de estos riesgos se han considerado como altos y extremos. El poco tiempo para su desarrollo ha afectado notablemente nuestras valoraciones.

Tabla 6. Ponderación de riesgos.

|  |  |
| --- | --- |
| Riesgo | Ponderación |
| Problemas con la comprensión de EPL. | EXTREMO |
| Problemas con la programación con CakePHP. | EXTREMO |
| El cliente no tenga detectados en lenguaje formal los patrones. | ALTO |
| El cliente no se interese por el proyecto. | EXTREMO |
| Enfermedad del Team. | ALTO |
| Cambiar de una base de datos relacional a otra que no lo es. | MODERADO |

1. Desarrollo

En este capítulo veremos todos los artefactos generados a lo largo del desarrollo del proyecto utilizando una metodología ágil, en concreto, SCRUM con prototipado, tal y como hemos comentado en el Capítulo 4

* 1. Sprint 0. Acercamiento a CEP

Este sprint 0 es un período de introducción a este proyecto de fin de grado, ya que unicamente se realizaron labores de aprendizaje y acercamiento a CEP. En la Tabla 7 podemos ver la fecha de inicio y fin, y las personas que se involucraron en este sprint.

Tabla 7. Información base de sprint 0.

|  |  |
| --- | --- |
| **Fecha de inicio y fin** | 25 de octubre de 2017 – 22 de diciembre de 2017 |
| **Personas involucradas** | * Gregorio Diáz Descalzo. * Pedro Luis Salazar. * Enrique Brazález Segovia. |

* + 1. Reunión de planificación del sprint

Esta reunión sirvió como inicio, y para ver qué proyectos de los ofertados por Ingeteam son amoldables a la tecnología CEP.

* Dirección: I3A
* Hora y fecha de la reunión. A las 16:00 el día 26 de octubre de 2017.

Los asistentes de la reunión son:

* Pedro Luis Salazar. Encargado del departamento de I+D+i en el año 2017.
* Gregorio Diaz. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que tratar

Esta reunión tiene la finalidad de poder encontrar un proyecto dentro de los ofertados por el cliente para que pueda ser solucionado utilizando la tecnología CEP. Además, también el Scrum Master explicará CEP por primera vez al Product Owner y al Team. Los puntos de esta reunión fueron:

* ¿Qué es CEP?
* Presentación de proyectos.
  + Energía eólica.
  + Energía fotovoltaica.
  + Pozos petrolíferos.
* Situación actual del Team, y fecha de inicio del desarrollo.
* Curso CEP ofertado por el máster profesional de la ESIIAB.
  + - 1. Información recogida

Al comienzo de la reunión Gregorio, hizo una pequeña introducción disponiendo las ventajas y desventajas, así como el objetivo que tiene la tecnología CEP a los asistentes de la reunión. A continuación, se presentaron diferentes proyectos en los que se podría amoldar CEP. Los proyectos fueron los siguientes.

* **Pozos petrolíferos.** Analizar las vibraciones de la aguja que penetra la superficie terrestre previniendo su rotura o las cargas de trabajo excesivas, y optimizar la extracción de petróleo.
* **Energía fotovoltaica.** Analizar el desgaste de las plantas fotovoltaicas y optimizar la producción a raíz de los datos obtenidos por sus sistemas de monitorización.
* **Energía eólica.** Analizar el desgaste de los aerogeneradores de un parque eólico para poder optimizar la producción en base a los datos obtenidos por sus sistemas de monitorización.

Elegimos el proyecto relacionado con la energía eólica por las siguientes razones:

* Las mediciones de los pozos petróliferos no son regulares y no es rentable conectarnos al flujo de datos real de la aguja.
* Los datos provenientes de las placas fotovoltaicas son muy escasos, los sistemas de monitorización del cliente en este tipo de plantas solo tienen lecturas de temperaturas y producción.
* Los datos en el caso de un aerogenerador se leen cada 10 minutos, lo que nos asegura una lectura regular.
* Es viable la conexión en tiempo real con el flujo de datos de un parque eólico por medio de la subestación.
* Los datos de un aerogenerador son muchos, desde temperaturas de cada uno de sus componentes, así como la dirección del viento, velocidad del viento, potencia producida y la presión atmosférica.
* El cliente cuenta con amplia experiencia en el ámbito de la energía eólica, y cuenta con un gran volumen de datos para que nuestra aplicación entrene.

Una vez elegido el ámbito en el que íbamos a aplicar CEP, se habló sobre la fecha en la que podríamos empezar el proyecto, y como se iba a trabajar. En este momento, se explicó que se iba a aplicar una metodología SCRUM y que el comienzo oficial del desarrollo del TFG empezaría el día 8 de enero de 2018. Por último, Gregorio, sugirió acudir a un seminario acerca de la tecnología CEP en la Politécnica de Albacete, impartido por Juan Boubeta Puig, docente en la universidad de Cádiz.

* + - 1. Próximos pasos

En esta reunión se han acordado las siguientes tareas:

* Informarnos sobre cómo funciona CEP.
* Acudir al seminario de Procesamiento de Eventos Complejos de la universidad.
* El día 9 de enero acudir a las oficinas para comenzar el trabajo de fin de grado con un seminario de energía eólica.
* En la reunión del 8 de enero, se enunciarán los requisitos que el sistema debe de cumplir, para que el Team los traduzca a funcionalidades.
  + 1. Desarrollo del sprint

Debido a que se trataba de un sprint introductorio en el que solo hubo una reunión, únicamente se realizaron tareas de aprendizaje, para poder adentrarnos en el mundo del Procesamiento de Eventos Complejos y conocer lo mejor posible esta tecnología.

En la Tabla 8 podemos ver el resumen de lo llevado a cabo en este sprint, reflejando la estimación en horas de cada una de las tareas realizadas

Tabla 8. Resumen de desarrollo sprint 0.

|  |  |  |
| --- | --- | --- |
| Historias de usuario | Tareas | Horas |
| Introducción a Tecnología CEP | Acudir al seminario de CEP del máster | 9 |
| Instalación del software | 4 |
| Realizar ejercicios de CEP | 9 |

En este sprint simplemente definimos cuál iba a ser el marco tecnológico que iba a tener este trabajo de fin de grado. Se adquirieron los conocimientos teóricos acerca la tecnología CEP vistos en el CAPÍTULO 2.

Esta iteración debido a que es introductoria no tiene reunión de revisión del sprint ni retrospectiva, ya que la única tarea que había que llevar a cabo era asistir al curso de la universidad. Scrum Master, Product owner y Team, acordaron comenzar oficialmente el desarrollo de este proyecto a principios de enero.

* 1. Sprint 1. Inicio del proyecto

En esta iteración se planteará el proyecto y se interiorizarán todos los aspectos técnicos acerca del ESB, la energía eólica, etc. Además, plantearemos la planificación del proyecto, realizaremos una primera versión de los requisitos a alcanzar, y se elaborará un primer prototipo del flujo de datos, usando un intermediario vía Web, “ThingSpeak”. Esta aplicación externa servirá de intermediario entre el generador de eventos y el motor CEP. En la Tabla 9, podemos ver las fechas que abarca este sprint y sus participantes.

Tabla 9. Información base del sprint 1.

|  |  |
| --- | --- |
| **Fecha de inicio y fin** | 8 de enero de 2018 – 18 de febrero de 2018 |
| **Personas involucradas** | * Francisco José Polo Sánchez. * Pedro Luis Salazar. * Gregorio Diáz Descalzo. * Vicente Requena Montejano. * Antonio Fernández Diez. * Enrique Brazález Segovia. * Fernando Luján Martínez |

* + 1. Reunión de planificación del sprint

Esta reunión sirve como punto de partida en el desarrollo de este trabajo de fin de grado, puesto que se enuncian los requisitos que la aplicación ha de cumplir.

* Dirección: I3A
* Hora y fecha de la reunión. A las 16:00 el día 8 de octubre de 2017.

Los asistentes de la reunión son:

* Pedro Luis Salazar. Encargado del departamento de I+D+i en el año 2017.
* Gregorio Diaz. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que tratar

Los puntos de esta reunión fueron los siguientes:

* Enunciado de requisitos por parte del Product Owner.
* Planificación del sprint realizando las estimaciones de tiempo oportunas.
  + - 1. Información recogida

En la primera parte de la reunión, el Product Owner propuso con detalle el problema a solucionar, y cómo se organiza el modelo de parque eólico que queremos analizar utilizando la tecnología CEP. La situación y la problemática es la siguiente.

Un parque éolico está compuesto por aerogeneradores. Cada uno de estos aerogeneradores cada diez minutos recoge todos los datos que sus sensores captan. Esta información que se despliega cada diez minutos se recoge en forma de registro con 75 columnas. Estas columnas tienen mediciones de la temperatura de la multiplicadora, velocidad de viento, potencia producida, nivel de aceite en los depósitos del aero, identificador de máquina, etc. Un parque estándar con 31 aerogeneradores genera diariamente 4464 lecturas y anualmente se almacenan aproximadamente mas de un millón y medio de registros. Esta ingente cantidad de datos no es rentable almacenarla, de forma que se necesita un sistema que consiga analizar esta información a tiempo real para evitar su almacenamiento, o al menos, el que no es significativo. Se requiere que se analicen esos datos con el propósito de detectar con anterioridad defectos en el mecanismo de los aerogeneadores, evitando su desgaste. La idea inicial es que se conecte este sistema con los PLCs de los aerogeneradores por medio de las subestaciones de los parques, sin embargo, no hay parque que pueda soportar tal comunicación a día de hoy. Por el momento, deberemos de trabajar con paquetes de datos sintéticos y simular el flujo de los datos.

El Product Owner quiere que el sistema sea externo al que tiene implantado actualmente, para así, evitar dependencias con el sistema que está por desarrollar. Se presupone que la aplicación inteligente estará albergada en un servidor UNIX, y que la información de las alertas detectadas por él estará desplegada en una página web implementada con el framework CakePHP, aunque está abierto a otro tipo de tecnología web.

Enunciada la problemática y requisitos del cliente, a continuación, el Team, se separará del resto de integrantes de la reunión para planificar el sprint.

* + 1. Planificación del sprint

La organización de este sprint queda recogida en la Tabla 10, enunciando cada una de las historias de usuario, así como sus tareas y estimación de horas de trabajo.

Tabla 10. Resumen de desarrollo sprint 1.

|  |  |  |
| --- | --- | --- |
| Historia de usuario | Tarea | Horas |
| Primer estudio de la curva de potencia | Establecer curva de todos los datos | 4 |
| Realizar el anteproyecto | Adaptar plantilla al caso concreto | 2 |
| Elaborar anteproyecto | 15 |
| Refactoring de anteproyecto | 3 |
| Recogida de requisitos y análisis de contexto | Reunión de inicio del proyecto | 2 |
| Recogida de requisitos de la reunión | 3 |
| Planteamiento de la recogida de requisitos | 4 |
| Buscar y elaborar plantilla para storyboard | 2 |
| Estructurar el desarrollo | 1 |
| Reunión 2 de incio de proyecto | 1 |
| Arquetipos | 2 |
| Documentar reuniones | 7 |
| Acoplamiento e implementación del ESB | Asimilación del código del ESB y todos los componentes de AnyPoint Studio. | 5 |
| Implementación de los primeros flujos de datos e incorporación de librería Esper-EPL. | 10 |
| Realizar tutorial de MuleSoft para familiarizarse. | 5 |
| Realizar filtrado de datos. | 5 |
| Planificación del trabajo de fin de grado | Elaboración de cronograma. | 2 |
| Elección y aprendizaje de repositorio. | 2 |
| Elección y aprendizaje de TFS en metodologías ágiles. | 3 |
| Curso de Energía Eólica | Clase intensiva de Energía Eólica | 4 |
| Interiorizar y estudiar el curso ofrecido por el Product Owner. | 3 |
| Creación y acoplamiento del simulador con la Nube | Asimilación del código propuesto y dominar las librerías de Thingspeak y poi. | 3 |
| Creación de modelo contrastando con datos sintéticos. | 3 |
| Implementación y modificación del simulador. | 10 |
| Realizar scripts de conversión de archivos .DB a .xlsx | 3 |
| Fallo en paradox para transformar en .xlsx | 10 |

* + 1. Reuniones de Seguimiento

En la Tabla 11 quedan reflejadas las reuniones semanales que tuvieron lugar a lo largo de este sprint.

Tabla 11. Reuniones semanales del sprint 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la reunión | Fecha | Participantes | Asuntos que tratar |
| Conocemos al cliente, primera toma con el experto. | 8/1/2018 | Vicente Requena Montejano.  Enrique Brazález Segovia.  Antonio Fernández Diez. | * Presentación. * Objetivos del proyecto. * CEP. * Situación actual del contexto. * Clase introductoria de energía eólica. * ¿Qué tipo de aerogenerador vamos a analizar? |
| Inicio del proyecto de trabajo de fin de grado. | 11/1/2018 | Gregorio Diaz Descalzo.  Enrique Brazález Segovia.  Fernando Luján Martínez. | * Criterios de evaluación de la convocatoria. * Aclaración de las fechas de las que disponemos. * Explicación de cómo hacer el anteproyecto. * Definir planificación del proyecto. * ¿Qué metodología vamos a utilizar? ¿Qué iteraciones nos vamos a marcar? * Explicación de sistema de citas. |
| Dudas y problemas | 23/1/2018 | Gregorio Diaz Descalzo.  Enrique Brazález Segovia.  Fernando Luján Martínez. | * Dudas a la hora de realizar el anteproyecto, posibles restricciones, aspectos positivos, aspectos negativos. * ¿Dónde está Pedro? * No sabemos de qué información bebe el motor de datos, hay que decidir cómo se nutre al motor que procesa los eventos. * Recogida de requisitos |
| Coordinación y análisis de la situación | 1/2/2018 | Vicente Requena Montejano.  Francisco José Polo Sánchez.  Antonio Fernández Diez.  Gregorio Diaz Descalzo.  Enrique Brazález Segovia.  Fernando Luján Martínez. | * ¿Quién es el encargado de coordinar el proyecto como Pedro se fue? * Explicar otra vez cómo es la mecánica de la tecnología CEP. * ¿Dónde vamos a integrar la aplicación? * ¿Cómo tenemos que mostrar la ocurrencia de los eventos complejos? * ¿Cuál es la nueva forma de trabajar después de irse Pedro? * Ingeboards y SCADA. |
| Clase tutorizada de energía eólica de Francisco José Polo | 2/2/2018 | Francisco José Polo Sánchez.  Enrique Brazález Segovia. | * Explicación del modelo de datos utilizado. * ¿Qué es el “SystemNumber”? * Nociones de energía eólica para tener en cuenta. |

* + 1. Desarrollo del sprint

En este punto describiremos qué artefactos se generaron en este sprint, destacando los puntos más importantes de cada una de las historias de usuario nombradas anteriormente.

* + - 1. Curso de Energía Eólica.

Vicente Requena es el encargado de proporcionar este curso, donde vemos cómo es el funcionamiento base de un aerogenerador, así como todos los factores que le pueden influir. Como fruto de la comprensión y estudio individual de este curso, se obtuvieron las conclusiones redactadas en la Tabla 12.

Antes de comenzar con las conclusiones, hemos de destacar que el estudio se ha cerrado hacia un aerogenerador estándar de 1500 KW. Actualmente, el acceso a los datos obtenidos por los PLCs a tiempo real es complicado, ya que no hay una infraestructura actual que lo permita sin tener que invertir economicamente.

Tabla 12.Conclusiones de curso introductorio de energía eólica.

|  |  |  |
| --- | --- | --- |
| Nombre | Descripción | Trababjamos con |
| Potencia nominal | Se trata de la potencia sacada por el aerogenerador en su rendimiento más alto. | 1500 W |
| Rango de viento | - | 4 -25 m/s |
| Tensión | La tensión que sale del generador para la estación distribuidora. | 690 V – 12000 V |
| Giro del rotor | El giro del rotor | 10 – 30 rpm |
| Palas | Son 3 palas que van acopladas al eje central, dos posiciones extremas. Atención los grados son respecto el eje central de la junta de las palas “CONO”. A más palas más coste, 3 es lo óptimo en cuanto a relación dinero/optimización | 2 posiciones  0º Barlovento (máximo rendimiento)  90 º Bandera (mínimo rendimiento) |
| Nacelle | Góndola | - |
| Rotor | Cono y palas | - |
| Pitch | Giro de las aspas sobre el cono para cambiar su grado (de 0º a 90º) | Es un pitch eléctrico |
| Multiplicadora | Al ser un aerogenerador con dipolo la velocidad del rotor es muchísimo menor a la que le hace falta al generador para producir energía eléctrica, de forma que necesitamos una multiplicadora para adecuar ambas velocidades a la de sincronismo. | Trabajamos con un aerogenerador con multiplicadora eléctrica. |
| Etapa planetaria | Comunica directamente la parte acoplada al rotor, y transmite la velocidad de este a la fase paralela | La lubricación es muy importante, debido a la generación de partículas de suciedad que producen desgaste. |
| Etapa paralela | Conjunto de engranajes para derivar una velocidad baja transmitida por la fase planetaria al generador con la velocidad de sincronismo adecuada. | La lubricación es muy importante, debido a la generación de partículas de suciedad que producen desgaste. |
| Lubricación y refrigeración en multiplicadora | Es importante debido a que SIEMPRE ha de estar a una temperatura y nivel de aceite adecuado. La temperatura es controlable, pero es INEVITABLE el cambio de aceite debido a su gran uso. | Dos alertas posibles:  ¿Se han desgastado los filtros?  ¿Va a estallar la resistencia calefactora?  Prestar atención en:   * Rodamientos. * Engranajes. * Resistencia calefactora. * PT100 – Sensor térmico del sistema de lubricación. |
| Junta rotativa | Evita que los cables de las partes del aerogenerador que se comunican, se lien. | Puede haber más de una junta rotativa. |
| Generador doble alimentado | Se trata del elemento que hace la conversión y genera la energía que se mandará a la estación eléctrica.  Posee un circuito de producción para la red (el normal), y otro circuito que realimenta al generador para producir energía en caso de haber poco viento, de esta forma ayudaremos al rotor a girar. | Tiene una tensión que va a la estación de 690 V de forma que hay que subirla para que no haya pérdidas  1500 rpm  10 % deslizamiento  2 – 4 pares de polos.  Tener en cuenta:   * PT100. * Cuerpo de anillos. * ¿Refrigeración aire o agua? |
| Polipasto | Grúa de ayuda para llevar herramienta pesada en pequeñas reparaciones. |  |
| Sistema del YAW | El YAW es el giro de la góndola en 360º con respecto el eje vertical del aerogenerador. | Es un sistema de YAW eléctrico. |
| Freno de fricción | Frena el movimiento de YAW en algún momento. | Alertas que se pueden detectar:  ¿Excesos de calor?  ¿Número de veces utilizado? |
| Dispositivo cuentavueltas | Cuentavueltas de góndola para evitar enrollamiento de los cables de potencia. | Máximo dos vueltas, se tiende a desenrollar tras las dos. |
| Grupo hidráulico | Controla toda la hidráulica del rotor y las palas. | Control de aceite.  Acumulador de nitrógeno.  Preostatos.  PT100. |
| Armario TOP | Donde se ubica toda la lógica software del sistema, desde el movimiento del YAW, los controles de las temperaturas y sistemas hidráulicos hasta las rotaciones del PITCH. | - |
| Armario GROUND | Controla toda la potencia y las funcionalidades de mantenimiento. | - |
| Transformador | Elevación de la tensión de 690 V obtenidos por el generador, hasta el voltaje requerido para que no haya perdida de energía. | - |

* + - 1. Primer estudio de la curva de potencia

Todo el análisis de datos gira en torno a la curva de potencia. Esta curva esquematiza el rendimiento del aerogenerador, enfrentando el viento en ese momento contra la potencia producida tal y como vemos en la Ilustración 11. Podemos ver que hay tres fases dentro de la curva de potencia, el arranque, la situación normal y la etapa en la que está a pleno rendimiento.

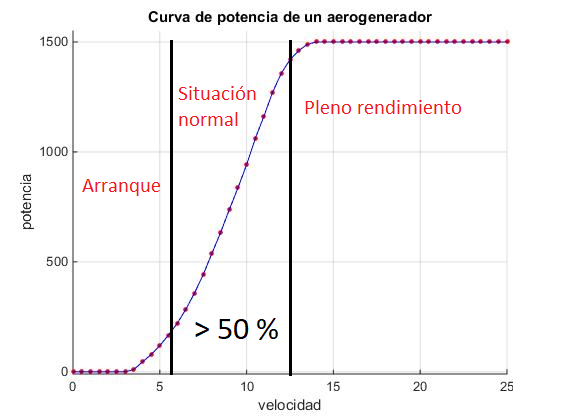


Ilustración 11. Ejemplo de curva de potencia.

Cabe destacar dos aspectos muy relevantes de las etapas de la curva de potencia. Por un lado, diremos que más de la mitad de las mediciones de un aerogenerador se encuentran en la fase de “situación normal”, por lo que los activos están diseñados para trabajar en esas condiciones. Por otro lado, tenemos la etapa de “pleno rendimiento”, que es cuando la velocidad de viento excede de los 15 m/s, donde se alcanza la potencia nominal. Esto se debe a que los aerogeneradores pueden sufrir averías si el rotor excede ciertas velocidades, incluso podría caerse al suelo debido a la fuerza generada por las palas.

Muchos de los patrones que vamos a implementar analizan los fenómenos que se dan en las curvas de potencia de los aerogeneradores. Por ello, debemos de estudiar los factores que afectan a la curva de potencia. Los más relevantes son:

* **Formaciones en parques eólicos.** Tal y como vemos en la Ilustración 12, podemos observar que no es trivial la formación que hacen los aerogeneradores en un parque eólico. En un primer momento podríamos pensar que cuantos mas aerogeneradores, más viento aprovechamos y más dinero obtenemos. Sin embargo, esto no es así, ya que una vez que el viento pasa a través de los aerogeneradores, genera turbulencias perdiendo fuerza, por lo que no es igual de aprovechable que al principio. La formación en la que son organizados los aerogeneradores son fruto del balance de los costes y el rendimiento en cada posición. Cuando se construye un parque eólico se intenta encontrar aquella formación que permite obtener la máxima energía posible con el mínimo gasto de recursos (cableado, número de aerogeneradores, infraestructura, etc). La situación geográfica tiene un papel clave para aprovechar la máxima energía generada por el viento. Por lo tanto, dada una formación de aerogeneradores y una situación geográfica concreta, unos están destinados a producir más que otros. Esto se debe, a que como están mejor posicionados, les llega mayor cantidad de viento.

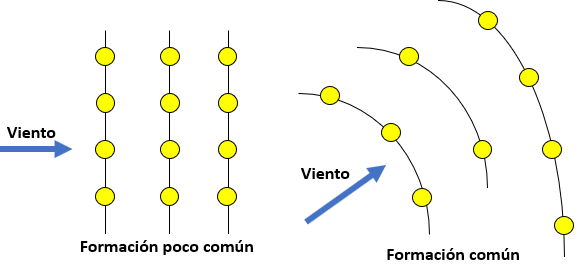


Ilustración 12. Formaciones comunes y no comunes de un parque eólico.

Por ejemplo, suponiendo que tenemos la misma densidad de aire en todo el parque, un aerogenerador ubicado en una llanura no le llega la misma cantidad de viento que uno que está en lo alto de un cerro. El aire que fluye a través de la montaña es impulsado hacia la parte de arriba del terreno por lo que aprovechamos el doble de su fuerza reduciendo el espacio por donde el viento fluye.

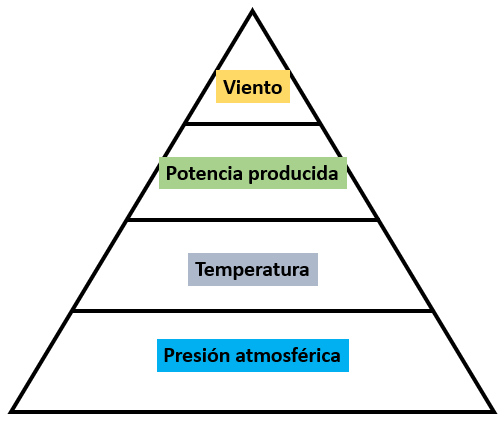
* **Factores involucrados.** A la hora de analizar los datos proporcionados por SCADA hemos de tener en cuenta una pequeña jerarquía de factores. Esta jerarquía es la vista en la Ilustración 13. A más viento mayor es la potencia producida. A más temperatura del aire, menor densidad tiene el viento, de forma que, al tener menos fuerza, la potencia producida es menor. Es decir, cuanto más frío sea el viento más fuerza produce, y mayor es la potencia producida. Por otro lado, la presión también es importante, aunque en menor medida. Cuanta más presión hay, mayor es la fuerza que produce el viento sobre las palas del aerogenerador, por lo que la potencia producida es mayor.

Ilustración 13. Jerarquía de factores.

* **Defectos en la curva de potencia.** La curva de potencia se dibuja a través de una nube puntos dependiendo de las mediciones reales de un aerogenerador, sin embargo, estas mediciones tienen ruido. Este ruido podría ser por ejemplo 7 puntos con un viento de 20 m/s, pero con potencia reducida igual a 0. Esto podría ser porque el aerogenerador en cuestión está siendo reparado y se haya parado para ejercer labores de mantenimiento. Deberemos de quitar ese ruido a la hora de analizar los datos. Los defectos que podemos predecir examinando la curva de potencia son:
  + Error referenciado de pitch 0º en PLC > 0º en físico.
  + Suciedad en palas.
  + Errores en el tren mecánico.
  + Defectos en las palas.
  + Defectos eléctricos generados.
* **Mediciones de los fabricantes para vender aeros.** No podemos comparar nuestras conclusiones con los datos que los fabricantes otorgan, ya que se obvian muchos detalles que pueden ser muy relevantes.
  + - 1. Recogida de requisitos

Se introdujeron dos conceptos nuevos que por parte de los desarrolladores y el jefe de proyecto eran desconocidos. Hablamos de Ingeboards y SCADA. Tal y como vemos en la Ilustración 14, observamos dos sistemas o estructuras, que colaboran entre sí. Por un lado, tenemos SCADA, que es el sistema que tiene almacenados todos los datos recogidos por los sistemas de monitorización de los aerogeneradores, además, de los logs de averías, Por otro lado, tenemos Ingeboards, plataforma donde se visualiza la información dada por SCADA de una forma más legible. A través de Ingeboards, los técnicos encargados de las reparaciones de los aerogeneradores mandarán al sistema los informes técnicos que han llevado a cabo una vez terminadas sus reparaciones. Gracias a Ingeboards se puede determinar el rendimiento de los técnicos.

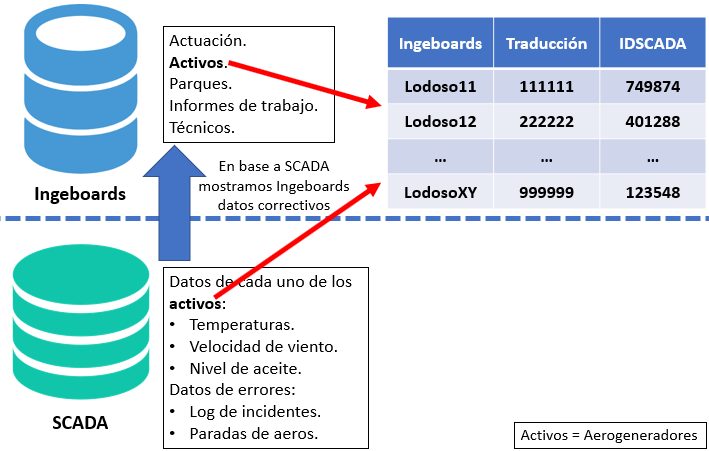


Ilustración 14. Infraestructura del entorno de Eolic Event Consumer.

Otro detalle muy importante entre la comunicación entre ambas aplicaciones es que cada uno de los aerogeneradores están identificados por un número, este número es el “SystemNumber”. Este identificador es perteneciente al sistema de SCADA. Sin embargo, en “Ingeboards” tiene otro identificador. Por ejemplo, en SCADA podríamos denominar a un aerogenerador con el id “92874621”, y en Ingeboards podríamos nombrar a ese mismo con el alias “MadridA1”.

Nuestra aplicación tiene que interactuar con ambos sistemas tal y como vemos en la Ilustración 15. Eolic Event Consumer trabajará debajo de “Ingeboards”, y paralelamente con “SCADA”. De esta forma, tras procesar los datos que son enviados desde “SCADA” y detectar un evento de interés, se insertaría una tupla a una tabla de la base de datos de “Ingeboards”, desplegando la información en forma de alarma.

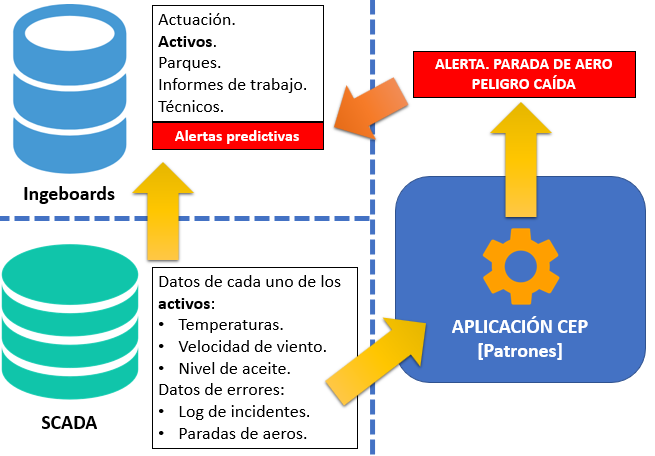


Ilustración 15. Incorporación de CEP al sistema actual.

Habiendo visto esto, como punto de partida, diremos que “NO TODOS” los campos captados por el sistema de monitorización son representativos para analizar. Solo tendremos en cuenta los factores más significativos, estos son la temperatura, la presión, la velocidad del viento y la potencia producida. Todos y cada uno de estos factores son los más importantes si examinamos el rendimiento de un aerogenerador. En concreto, haremos mucho hincapié en examinar la curva de potencia, que representa la potencia producida frente a la velocidad del viento.

No obstante, esto será nuestro punto de partida, en base a las conclusiones sacadas tras analizar estos cuatro factores iremos teniendo en cuenta el resto.

A la hora de analizar los datos deberemos de agruparlos por aerogenerador ya que las mediciones que se nos proporcionan son de un parque eólico. Cada parque eólico y sus medidas se han de contextualizar debido a que la situación geográfica es distinta. Tendremos que filtrar los datos puesto que hay mediciones que no son significativas y son despreciables de forma que estas pueden afectar negativamente a nuestras conclusiones. Ambos sistemas, tanto SCADA como Ingeboards, trabajan sobre bases de datos MySQL, de forma que a la hora de detectar un evento de interés a partir de los datos de SCADA, podemos insertar una tupla en la tabla “Alarmas” de Ingeboards con la información correpondiente.

* + - 1. Realizar anteproyecto y planificación del Trabajo de Fin de Grado.

Se repasaron con atención todos los criterios para la evaluación del trabajo de fin de grado. En concreto, se repasaron todos los capítulos de la memoria del trabajo de fin de grado para que no haya dudas en su redacción.

Los aspectos más importantes del anteproyecto son, por un lado, la parte de planificación y por otro, la justificación de las competencias correspondientes a la intensificación de ingeniería del software. Hablando de las fechas nos imponemos como fecha límite para tenerlo todo terminado, el día 17 de junio de 2018, a efectos de la presentación para la defensa, que va desde el día 25 al 29 de junio de 2018. Hay que destacar que el día 18 de junio se ha de hacer el depósito, de ahí que tengamos que tener todo preparado para el día 17. Utilizaremos SCRUM combinándolo con prototipado, de forma que tendremos 4 iteraciones y al final de cada iteración tendremos un prototipo listo y funcionando.

* + - 1. Planificación del Trabajo de fin de grado.

Debido a la ausencia de Pedro se acordó establecer una reunión con Antonio y Vicente para solucionar todos los problemas de coordinación y enmarcar los límites de nuestro proyecto. De esta forma, fijaríamos qué debe de incluir nuestro proyecto.

La problemática de la entrada de datos era muy grande, debido a que no teníamos acceso directo al PLC. Por lo tanto, tomamos la decisión de que se simularía a través de un servicio FTP, donde se colocarían archivos de un formato X (.db, csv, o Excel) en un directorio, y el Enterprise Service Bus se encargaría de procesarlo para enviar los eventos al motor CEP. En base a la facilidad de procesamiento, utilizaríamos archivos en formato csv.

Finalmente, decidimos tomar el enunciado de patrones en formato de storyboards, y en lenguaje natural aquellos aspectos que no se pudieran llevar a cabo con esta técnica.

* + - 1. Acoplamiento e implementación del ESB.

El esquema de este prototipo es el de la Ilustración 16. Podemos ver como introducimos un archivo “.csv” en el directorio “input” del simulador, recogiendo cada una de las tuplas correspondientes a los diez minutales de los aerogeneradores y mandándolos a un canal de Thingspeak. Thingspeak se encargará de ir almacenando todos y cada uno de los diezminutales en la nube. El ESB hará una petición vía HTTP al canal de ThingSpeak cada 4 segundos, para solicitar el último diezminutal que haya guardado. El ESB encolará los datos obtenidos por la respuesta del canal para enviarlos al motor CEP.

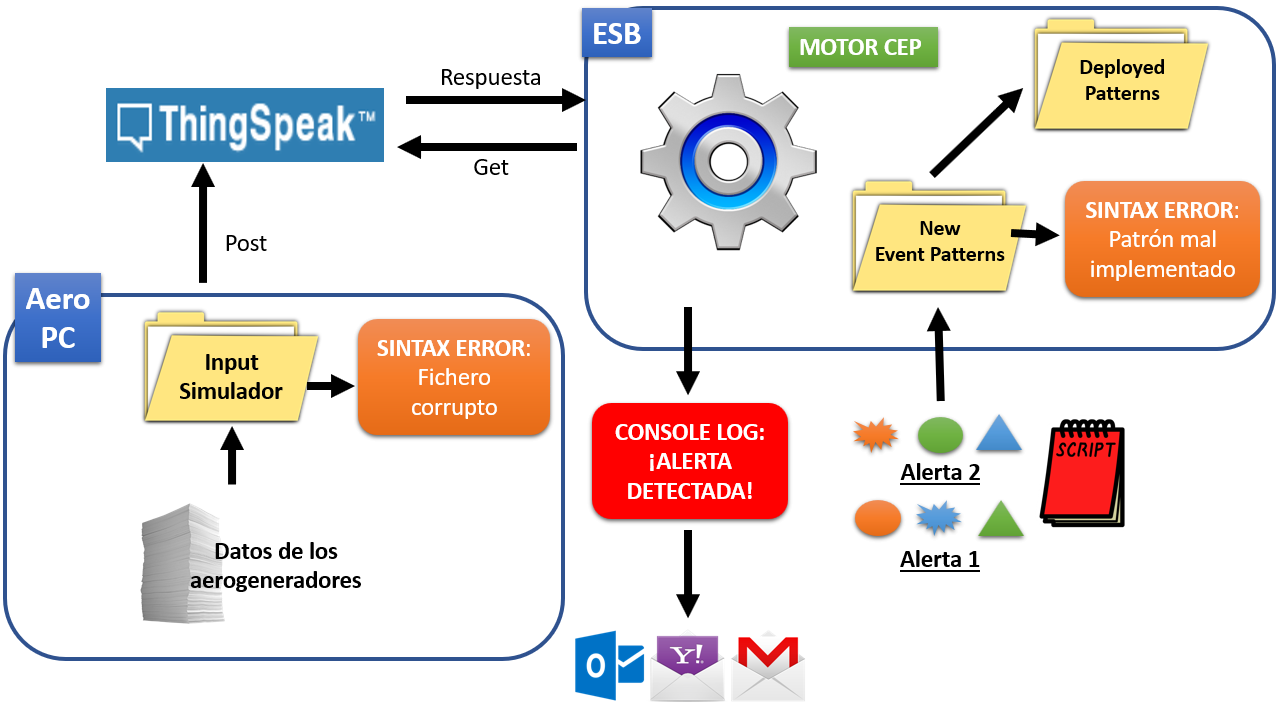


Ilustración 16. Esquema prototipo 1.

Los esquemas de los patrones son cargados en tiempo de ejecución, antes de ingresar un patrón deberemos de declarar el esquema de los eventos simples de los que depende. Para ello deberemos de introducir un dato de prueba. Una vez hecho esto podremos ingresar los patrones que hayamos implementado en el directorio “new event patterns” ubicado en el ESB. Si contiene algún error sintáctico o no están definidos los esquemas de los que depende, no se incluirá el patrón, y se mostrará en la consola del ESB el error.

Finalmente, cuando el patrón es detectado, todos los eventos complejos desplegados son mostrados por la consola del ESB. Además, el evento complejo es enviado por correo electrónico desde [pruebacepuclm@gmail.com](mailto:pruebacepuclm@gmail.com) hasta la dirección destino correspondiente.

* + - 1. Sprint Burndown.

Uno de los errores que se produjo en este sprint, es demorar la planificación del proyecto a la segunda semana puesto que desconocía el uso de la entrega de TFS para el desarrollo con metodologías ágiles, y el control de las horas se descolocó totalmente, tal y como podemos ver en la Ilustración 17.

Las horas estimadas en la planificación como vemos en al comienzo del sprint son 112 horas, sin embargo, al no saber como utilizar bien TFS, hubo varios problemas para la configuración del Sprint Burndown y solo se introdujeron 75 horas. Podemos ver esto, en la línea ideal de trabajo y en las subidas continuas a lo largo del sprint. El esfuerzo real de este sprint ha sido de 148 horas aproximadamente, y no 75 como aparece en el comienzo del Sprint Burndown, ni 112 como estimamos al principio, ya que hubo trabajo previo a la planificación del proyecto, problemas en la implementación del simulador y en la obtención de información de los archivos “db”. A todo ello, hemos de sumarle nuestro error a la hora de configurar TFS. Pese a estos errores, se consiguió estabilizar al final del sprint y se aprendió a utilizarlo correctamente.

Sin embargo, podemos decir que no hubo problemas en esta iteración ya que era bastante sencilla. Lo único que debíamos de hacer es recoger los requisitos para nuestro proyecto, amoldar el caso del curso a nuestro modelo de dato y adaptar la infraestructura de ThingSpeak y del ESB a nuestras necesidades.

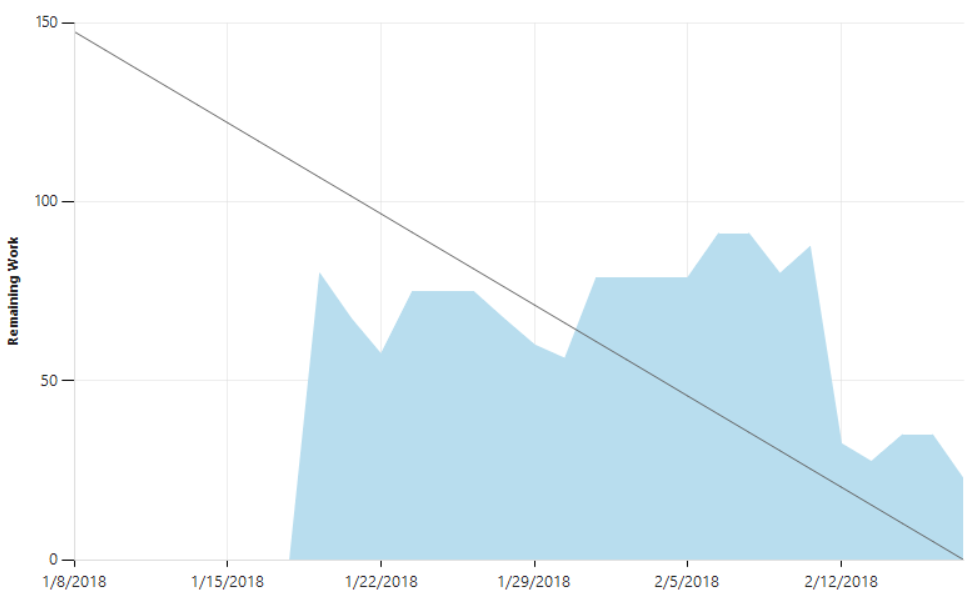


Ilustración 17. Sprint Burndown Inicio del proyecto.

* + 1. Reuniones al finalizar el sprint

Es la reunión para revisar que es lo que se ha realizado a lo largo de esta primera iteración, ver qué hemos hecho correctamente y qué problemas hemos visto, para de esta forma hallar cómo resolverlos para que no vuelvan a afectar negativamente en el proyecto.

* Dirección. Edificio Infante Don Juan Manuel. Avda. de España s/n. Albacete. Despacho 0. B.8.
* Hora y fecha de la reunión. A las 16:30 el día 14 de febrero de 2018.

Los asistentes de la reunión son:

* Gregorio Diaz Descalzo. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález Segovia. Desarrollador y encargado de defender este trabajo de fin de grado.
* Fernando Luján Martínez. Desarrollador de trabajo de fin de grado paralelo, su tutor también es Gregorio.
* Francisco José Polo Sánchez. Jefe de departamento de I+D+i de Ingeteam Service.
  + - 1. Asuntos que tratar

Esta reunión sirve como revisión de lo generado a lo largo de lo que se lleva de proyecto, ver qué aspectos positivos y negativos hemos visto, y en qué podemos mejorar. Los puntos de la reunión son:

* Revisión del prototipo.
* Revisión de la documentación.
* Revisión de como generar los requisitos.
* Como genero los storyboards.
* ¿Cómo insertar la documentación generada en las reuniones?
* Retrospectiva.
  + - 1. Información recogida

En cuanto a la revisión del prototipo. Hemos acordado que hemos cumplido con lo propuesto e incluido algún avance más como el ajuste de los datos, ya que el flujo de datos contiene ruido que influye negativamente en las predicciones. Queda validado, tanto el formato de datos en formato “csv” como la detección de eventos complejos y alarmas. En teoría el formato de entrada iba a ser en formato “.db”, sin embargo, debido a las dificultades que conlleva su transformación se ha optado con utilizar directamente el formato “csv”.

De momento el primer prototipo, recoge todos los datos obtenidos en “.csv”, y registro a registro alimenta al motor CEP. Los registros son diez minutales, cosa que deberemos de tener en cuenta, puesto que no son datos a tiempo real exactos, si no que están ponderados por la media de lo medido en esos diez minutos. En resumen, el prototipo generado en este primer sprint cumple las expectativas impuestas con creces, ya que no solo tenemos un incremento en el que manda información a ThingSpeak y lo retorna al ESB, sino que, además, nos saltamos la dependencia de ThingSpeak y lo alimentamos directamente a través de ficheros “.csv”. De esta forma, minimizamos dependencias y, por si fuera poco, incrementamos la lectura de los datos notablemente. En la revisión del prototipo se detectaron ciertas deficiencias en cuanto al modelo establecido, no obstante, eran simples erratas en el código que daban conflictos sintácticos muy sencillos de solucionar.

Si nos centramos en la documentación generada, se revisaron los siguientes artefactos:

* **Anteproyecto**. En cuanto a lo que el anteproyecto se refiere, he de destacar que había multitud de erratas a lo largo de todo el texto, fruto de mi inexperiencia a la hora de redactar textos formales. Se revisó cuidadosamente la justificación de las competencias, y se pidió expresamente que se incorporara alguna referencia al modelo de calidad actual. El jefe de proyecto hizo hincapié en justificar correctamente, sin aportar más información de la necesaria.

En cuanto a los medios a utilizar, no sabemos con exactitud todo lo que nos hará falta, ya que hay decisiones que no hemos tomado todavía.

* **Storyboards para captura de requisitos de patrones.** Los storyboards preparados para capturar los requisitos de los patrones eran correctos, y útiles a la hora de implementar todos y cada uno de los patrones. Echando un vistazo a las viñetas, teniendo nociones muy básicas de energía eólica, cualquier persona puede entender lo que pretenden detectar los patrones a implementar. Un ejemplo de storyboard puede ser el de la Ilustración 18.

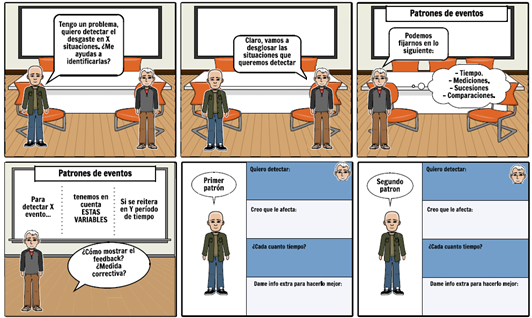
****

Ilustración 18. Plantilla de enunciado de patrones.

* **Actas de reuniones.** Una de las dudas del Team, es donde iban a tener cabida toda la documentación generada en cada una de las reuniones. Debido a que esta documentación es la correspondiente a la generada en una metodología SCRUM, la plasmaremos en el capítulo 4 de esta memoria, donde se verán todas las actividades desarrolladas en cada uno de los Sprints.
* **Arquetipos.** Este artefacto se ha añadido como extra por parte propia. Donde se esquematizan los integrantes que toman parte en el proyecto.
* **Información técnica clave.** Había dudas de cómo plasmar en la memoria del trabajo de fin de grado toda la información técnica acerca de la energía eólica que se recogerían en el segundo sprint. Se decidió que tendría cabida en el capítulo del estado del arte.
  + - 1. Próximos pasos

Se han acordado la incorporación de las siguientes historias de usuario para el siguiente sprint:

* **Estudio de factores externos distintos de la potencia producida y el viento.** En concreto, el estudio de lo que influye la temperatura y la presión, en la productividad y rendimiento.
* **Estudio de la curva de potencia.** Deberemos de hacer un análisis individual a nivel de aerogenerador y otra a nivel de parque eólico. Buscar fuentes de información de otras universidades, o artículos referentes a la energía eólica en busca de posibles comportamientos anómalos en los aerogeneradores. Sacar conclusiones y enunciar con ayuda del Product Owner, el enunciado de los patrones.
* **Implementar patrones.** Implementar una primera versión de los patrones a raíz de las conclusiones sacadas con el análisis.
* **Documentación**. Para adelantar y no dejar para el último momento la redacción de la memoria, se pone como tarea escribir el primer y segundo capítulo. Cuidar el estilo de la memoria, así como incorporar en el capítulo pertinente toda la información correspondiente al primer Sprint.
  + 1. Retrospectiva del Sprint

Una vez hecha la reunión de revisión del Sprint, el Product Owner abandonó la sala, para que el Team, junto con el Scrum Master, tuvieran la retrospectiva. El objetivo de esta reunión fue responder a las siguientes preguntas:

**¿Qué salió bien en la iteración?**

* Se hizo un buen prototipo, que tenía más funcionalidades de las esperadas.
* Dio tiempo a hacer parte de la documentación referente a todas y cada una de las reuniones.
* Se solucionó la coordinación con el cliente, pese al incidente de la ida de Pedro.
* La elección del ESB de MuleSoft, puesto que tiene multitud de funcionalidades.
* Hay muy buena comunicación entre el Team, y el Scrum Master. No hay ningún tipo de problema a la hora de reunirnos.
* La documentación está muy bien detallada.
* Los Storyboards plasman de forma correcta todo.

**¿Qué no salió bien en la iteración?**

* Debido a la mala configuración de TFS, el Sprint Burndown no refleja correctamente lo introducido en el servicio de Microsoft. Suponemos que es debido a que las tareas se redactaron conforme avanzó el Sprint.
* La conversión de los ficheros de “.db” a “.csv”. No se pudo llevar a cabo de forma automática debido a que el único lector de .db que permitía procesos “batch” que los transformaba era “paradox-reader”, sin embargo, no funciona correctamente puesto que la tarea se queda colgada y permite la conversión de dos ficheros seguidos.

**¿Qué mejoras vamos a implementar en la próxima iteración?**

* La implementación de los patrones.
* Amoldamiento de los patrones al ESB.
* Instauración de la base de datos MySQL donde introduciremos las alarmas de interés.
* Si sobra tiempo, integrar de forma automática la conversión de ficheros “.db”.
* Empezar la memoria del trabajo de Fin de Grado.
  1. Sprint 2. Adaptación al caso real

En este sprint analizaremos las conclusiones del análisis de las curvas de potencia y enunciaremos los patrones que queremos implementar. Además, quitaremos las dependencias con ThingSpeak y el ingreso de la información obtenida por los aerogeneradores será ingresada directamente en un servicio FTP del propio ESB. De esta forma, nuestro sistema será más independiente y más eficiente. Los participantes en este sprint son los que aparecen en la Tabla 13.

Tabla 13. Información base del sprint 2.

|  |  |
| --- | --- |
| **Fecha de inicio y fin** | 19 de febrero de 2018 – 29 de marzo de 2018 |
| **Personas involucradas** | * Francisco José Polo Sánchez. * Gregorio Diáz Descalzo. * Antonio Fernández Diez. * Enrique Brazález Segovia. * Fernando Luján Martínez. * Ignacio |

* + 1. Reunión de planificación del sprint

Esta reunión sirve como punto de partida en la adaptación al contexto real donde iría implantado nuestro sistema, además de la implementación de una primera versión de los patrones.

* Dirección: Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A.
* Hora y fecha de la reunión. A las 16:30 el día 19 de febrero de 2018.

Los asistentes de la reunión son:

* Francisco José Polo. Enca Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A. encargado del departamento de I+D+i en el año 2018.
* Ignacio García de Carellan Esteban-Infantes. Doctor ingeniero químico. Jefe R & D Manager de Ingeteam.
* Antonio Fernández Diez. Ingeniero Industrial. Especializado en placas fotovoltaicas.
* Gregorio Diaz Descalzo. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález Segovia. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que tratar

Los puntos de esta reunión fueron los siguientes:

* Qué detectar en concreto con los patrones.
* ¿Qué va a tener nuestro prototipo?
* ¿Qué patrones he detectar? ¿Qué va a ser objeto de estudio?
* ¿En qué plataforma hacemos la aplicación gráfica? ¿Node.js (sin complicación, ya sabía en su día utilizarlo) o usar PHP (un reto, no sabía utilizarlo)?
* ¿Algún problema con el cliente?
* Planificación del sprint realizando las estimaciones de tiempo oportunas.
  + - 1. Información recogida

En la primera parte de la reunión se expuso la necesidad de implantar un sistema que no dependiera de ThingSpeak y así poder establecer menos dependencias con sistemas externos a Ingeteam. Además, pasamos a recoger que fenómenos en los datos sería interesante analizar.

Lo más representativo de esta reunión fue el planteamiento de objetivos de este sprint. Los objetivos que se deberían de cumplir serían los siguientes:

* **El prototipo deberá de desplegar los patrones que se implementen dentro del ESB.** Lo más relevante de este sprint es el desarrollo de estos patrones, deberemos de centrarnos en el estudio de la curva de potencia.
* **Los eventos complejos de interés deberán de insertarse en una base de datos MySQL.** El Scrum Master sugirió utilizar algún software para incorporase tanto PHP como algún sistema gestor de base de datos como es MySQL, en concreto se tomó la decisión de utilizar XAMPP.

Otro punto de la reunión fue el estipular que patrones deberíamos de implementar. En una de las reuniones con el cliente, nos requirió examinar con escrupulosidad la curva de potencia. Se hizo mucho hincapié en estudiar el comportamiento de los aerogeneradores en los distintos bines de viento, es decir, el rendimiento que tienen en intervalos de viento concretos. También es interesante estudiar productividad, factor que enfrenta potencia producida y tiempo. Tendremos que normalizar el patrón, puesto que el factor geográfico es determinante a la hora de que un aerogenerador sea más productivo que otro. También se sugiere contrastar las conclusiones que saquemos con el log de errores, para así comprobar que la alerta que el motor CEP ha sacado es correcta, esto último se llevaría a cabo solo en caso de terminar el proyecto antes de tiempo. Por el momento, solo abarcaremos todo lo que tenga que ver con rendimiento en los diferentes bines de viento, y productividad.

A continuación, se trató el tema de la aplicación gráfica que se iba a implementar. Se discutieron varios tipos de tecnologías, todas y cada una de ellas, servicios web. Sobre todo, hemos de destacar la discusión de usar Node.js y un sistema tradicional a través del uso de PHP. Por un lado Node.js, nos daba la ventaja de que ya lo habíamos utilizado, cosa que no nos suponía ningún problema, sin embargo, el cliente utilizaba PHP en su sistema, y su equipo técnico no tenía ninguna noción acerca de Node.js, cosa que dificultaba mucho su integración en el sistema. Por esta razón, y por mis ganas de aprender con el desarrollo de este trabajo de fin de grado, se decidió utilizar PHP, tal y como el servicio técnico de informáticos del cliente utilizaba. Esta decisión facilitó la integración de nuestro trabajo, con el sistema de “Ingeboards” que el cliente estaba desarrollando. Esta decisión también me ayudó a adquirir más conocimientos acerca de una tecnología que hasta el momento me era desconocida.

Enunciada la problemática y requisitos del cliente, a continuación, el Team, se separará del resto de integrantes de la reunión para planificar el sprint.

* + 1. Planificación del sprint

La organización de este sprint queda recogida en la Tabla 14, enunciando cada una de las historias de usuario, así como sus tareas y estimación de horas de trabajo.

Tabla 14. Resumen de desarrollo sprint 2.

|  |  |  |
| --- | --- | --- |
| Historia de usuario | Tarea | Horas |
| Estudio de factores externos distintos de potencia y velocidad de viento | Estudio de la temperatura y humedad. | 6 |
| Estudio de presión. | 5 |
| Estudio de curva de potencia | Hacer comparación global como parque eólico. | 1 |
| Hacer un análisis individual por aero. | 5 |
| Familiarizarnos con la curva de potencia. | 5 |
| Pintar gráficas a nivel de parque. | 2 |
| Sacar hipótesis a partir del análisis. | 2 |
| Buscar información acerca de posibles detecciones y causas. | 5 |
| Implementar patrones | Estudiar sintáxis EPL. | 3 |
| Implementar patrón de productividad. | 25 |
| Implementar patrón de rendimiento. | 25 |
| Amoldar patrones al ESB y viceversa. | 2 |
| Preparar CSV para probar patrones en el ESB. | 3 |
| Documentación | Documentar reunión 1 | 1 |
| Recoger los patrones que implementemos como Storyboards. | 2 |
| Elaborar primer capítulo de la memoria del TFG. | 5 |
| Elaborar el segundo capítulo de la memoria del TFG. | 10 |
| Organizar estilos del documento del TFG. | 2 |
| Documentar el sprint primero en el capítulo de desarrollo. | 4 |
| Realizar artefacto de problemas encontrados, habilidades aprendidas, etc. | 2 |
| Integrar las actas de las reuniones en la memoria del trabajo de fin de grado. | 1 |
| Realizar acta de la reunión del día 20. | 1 |
| Reescribir capítulo 2. | 1 |
| Reescribir y releer capítulo de desarrollo. | 1 |
| Aprender PHP y familiarización con tecnología web | Hacer tutorial de w3Schools. | 5 |
| Implementar un método index para mostrar RankingProd Table. | 10 |
| Instalar MySQL | 2 |
| Empezar un proyecto en CakePHP en blanco con comunicación con MySQL. | 10 |
| Instalar y amoldar XAMPP a PHPStorm | 3 |

* + 1. Reuniones de Seguimiento

En la Tabla 15 quedan reflejadas las reuniones semanales que tuvieron lugar a lo largo de este sprint.

Tabla 15. Reuniones semanales del sprint 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la reunión | Fecha | Participantes | Asuntos que tratar |
| Reunión del comienzo del segundo | 24/2/2018 | Gregorio Diaz Descalzo  Enrique Brazález Segovia  Fernando Luján Martínez | * ¿Qué va a tener nuestro prototipo? * ¿Qué patrones he detectar? ¿Qué va a ser objeto de estudio? * ¿En qué plataforma hacemos la aplicación gráfica? ¿Node.js o CakePHP? * ¿Algún problema con el cliente? |
| Reunión para explicación de | 22 de febrero de 2018. | Gregorio Diaz Descalzo.  Enrique Brazález Segovia.  Fernando Luján Martínez. | * Fundamentos teóricos y explicación de patrones. * Aceptación o denegación de los patrones presentados. |
| Reunión para concretar y validar los patrones con el cliente. | 23/2/2018 | Francisco José Polo Sánchez.  Enrique Brazález Segovia. | * Fundamentos teóricos y funcionamiento de los patrones. * Aceptación o denegación de los patrones ideados. * Posibles mejoras en los patrones. * Apreciaciones que se podrían hacer extras utilizando el log de errores. |
| Reunión para revisión de código y comunicación de patrones definitivos | 22 de febrero de 2018. | Gregorio Diaz Descalzo.  Enrique Brazález Segovia.  Fernando Luján Martínez. | * Comunicación de los patrones definitivos. * Exposición de dudas en un patrón, puesto que no se sabía en su día como plantearlo. * Coordinación del trabajo de fin de grado. |

* + 1. Desarrollo del sprint

En este punto describiremos qué artefactos se generaron en este sprint, destacando los puntos más importantes de cada una de las historias de usuario nombradas anteriormente.

* + - 1. Patrones en Storyboards.

Partiendo de las plantillas de storyboard elaboradas en el sprint anterior recogimos cuales son los patrones en forma de storyboard. De esta forma tendríamos claros cuales son los factores principales que intervendrían en el análisis para implementar adecuadamente los patrones. Para ello en la reunión 2, se rellenaron estos storyboards.

En primer lugar, se hizo una presentación orientativa para introducir las nociones básicas de CEP, así como varios ejemplos para enseñar al Product Owner el alcance de la sintáxis de EPL. Tal y como hemos visto en el Capítulo 2, nos servimos de 1 y 2 de la Ilustración 19 para hacer esta pequeña introducción.

Basándonos en las nociones de EPL aprendidas en el sprint anterior pudimos dividir el enunciado de los patrones en tres partes como vemos en 3 y 4 de la Ilustración 19.

1. **Nombre del patrón.** ¿Qué queremos detectar? ¿Cómo podríamos representar el evento con un nombre?
2. **Variables para tener en cuenta.** ¿Cuáles son las variables que utilizaremos para detectar este evento? ¿Hay datos calculados?
3. **Período de tiempo.** ¿Cada cuanto tiempo se da? ¿Qué período de tiempo se quiere analizar?

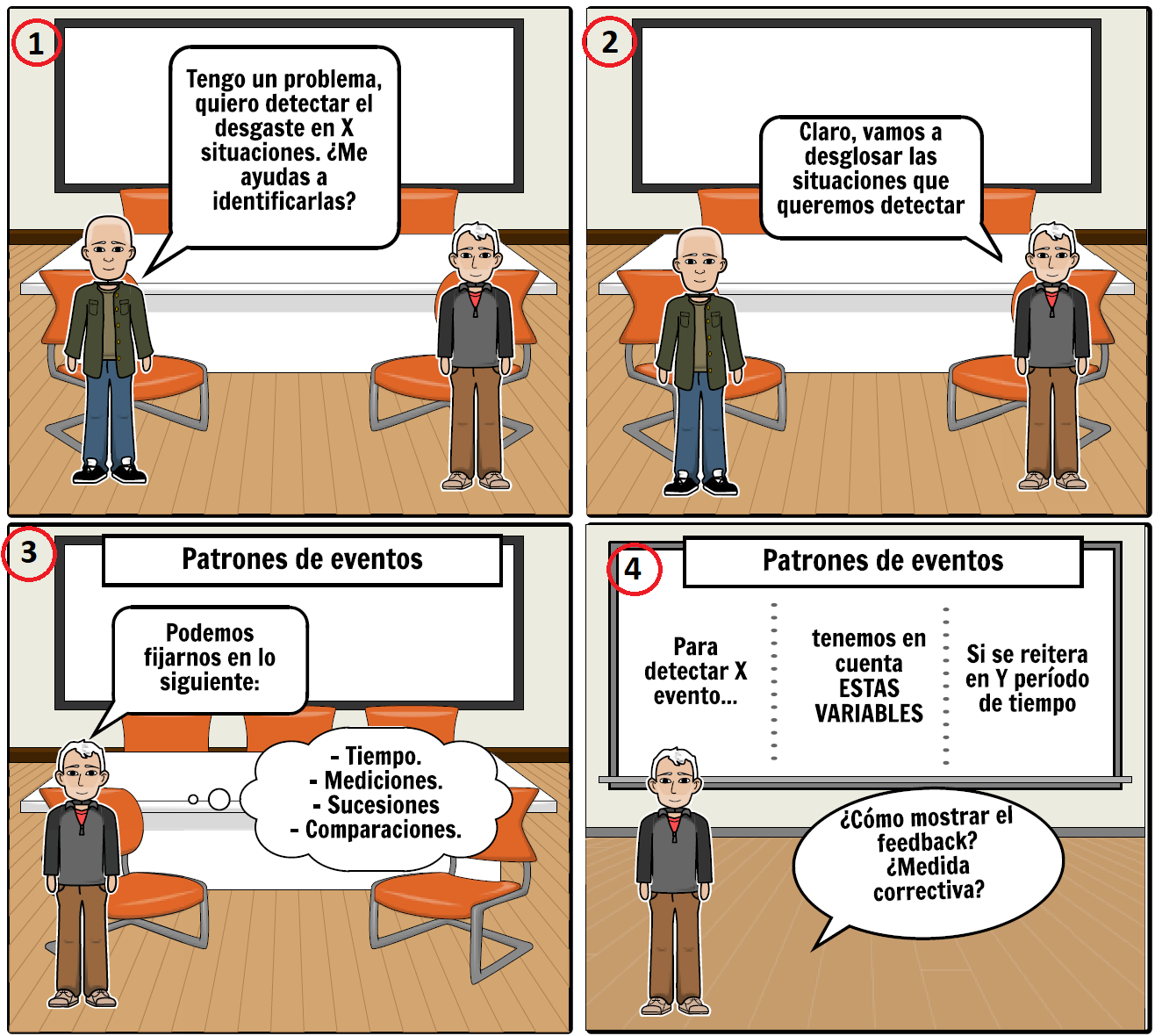


Ilustración 19. Introducción de elaboración de Storyboard.

Tras esta reunión los patrones quedan recogidos en la Ilustración 20. En el siguiente apartado explicaremos uno a uno en qué consisten.

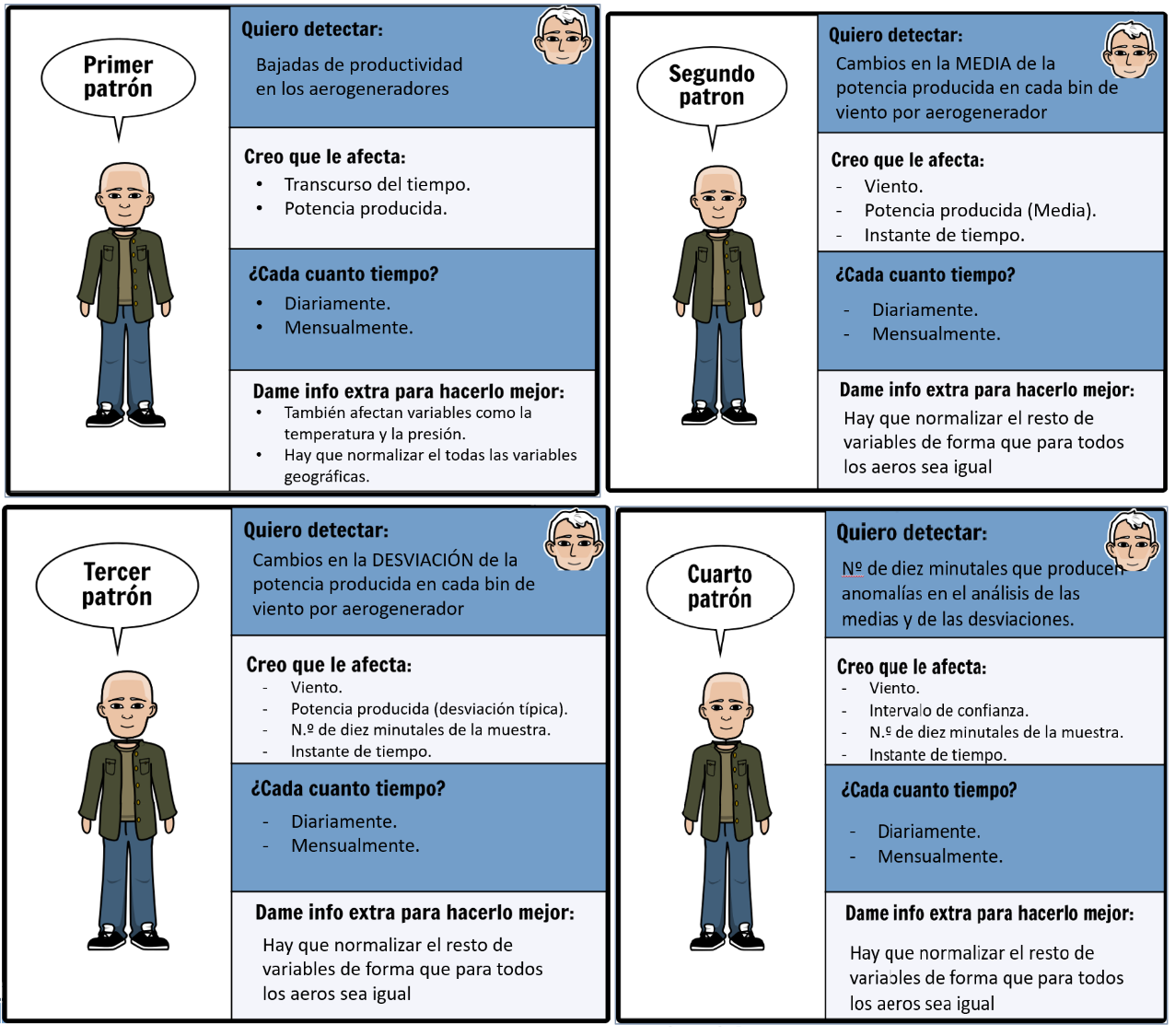


Ilustración 20. Patrones en formato Storyboard.

Finalmente, la reunión terminó de forma informal para poder causar una sensación agradable entre los asistentes con las diapositivas vistas en la Ilustración 21, de esta forma podremos fomentar el buen ambiente entre los stakeholders y mejorar el trabajo de cara al proyecto.

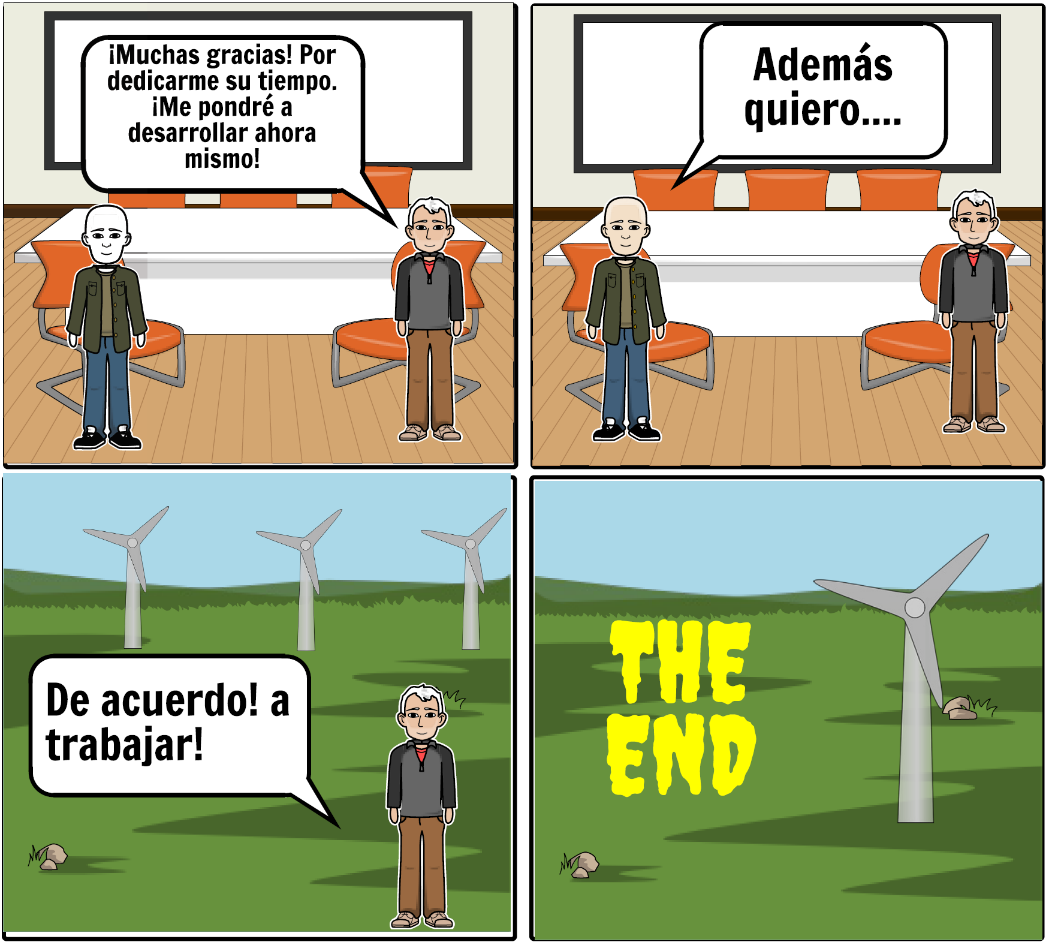


Ilustración 21. Final de la elaboración de los storyboards.

* + - 1. Análisis de la curva de potencia y propuesta de patrones.

Tras unos días estudiando las nociones teóricas de la energía eólica junto con algunos expertos procedentes de la empresa cliente, hemos sacado varias conclusiones y un primer acercamiento a los patrones a implementar. Los patrones miden productividad y rendimiento. La gran diferencia entre ellos la podemos ver en la Tabla 16.

Tabla 16. Factores enfrentados.

|  |  |
| --- | --- |
| Rendimiento | Productividad |
| Viento VS Potencia producida | **Tiempo VS Potencia producida** |

Como podemos ver, por un lado, enfrentamos el viento y la potencia producida para analizar el comportamiento de los aerogeneradores en los distintos bines de viento. Por otro lado, tenemos otro que enfrenta la potencia producida en un intervalo de tiempo concreto, obviando la cantidad de viento que haya. A continuación, veremos la explicación de estos dos patrones.

* **Patrón de Rendimiento**

Para medir el rendimiento nos centraremos en el estudio de la curva de potencia de los aerogeneradores. Sin embargo, hay ciertos problemas a la hora de comparar “el buen rendimiento”, ya que no todos los molinos, reciben en un instante X, una velocidad de viento Y.

El viento en una región tan amplia como un parque eólico es variable, ya bien sea por la disposición de los aerogeneradores (en matriz, en filas irregulares, etc.) o bien por la disposición geógrafica en la que se encuentran (uno encima de un cerro, otro en mitad de una llanura, etc.). Por tanto, este patrón lo que intenta es ver qué aerogeneradores son los mejores en distintos intervalos de viento. Los intervalos de análisis corresponden con los bines de viento (medida normalizada en el ámbito de la energía eólica), es decir, de 0 m/s a 0.5 m/s, de 0.5 m/s a 1 m/s, y así hasta 18 m/s, puesto que a partir de 18 m/s se da la potencia nominal del aerogenerador que estamos estudiando. Por ejemplo, en la Tabla 17, vemos el estudio del rendimiento en los bines de viento 6 y 6.5. Esta división fue posterior al desarrollo de la primera versión de los patrones, ya que al comienzo el Team pensó que sería mejor dividirlo en intervalos unitarios, sin embargo, esto se corrigió tras la revisión del sprint. Sin embargo, el equipo del proyecto hemos considerado más apropiado explicar con esta corrección los patrones que pretenden analizar el rendimiento. Más adelante veremos el código en EPL, donde tomamos intervalos unitarios (de 0 a 1, de 1 a 2 y así hasta 18 m/s).

Una vez hecha esta división, estas mediciones son comparables entre sí, y sí que se podrían analizar por algún tipo de criterio sin tener en cuenta la disposición geográfica. A raíz de ahí, veremos que aerogeneradores son los “peores”. Teoricamente todos los aerogeneradores deberían de producir exactamente lo mismo, sin embargo, si hay una diferencia significativa, puede que haya un fallo que el sistema de monitorización propio de la máquina no haya detectado. Además, gracias a este control podremos prevenir este tipo de deficiencias.

Veremos por un lado el análisis de las medias, y por otro lado el análisis de sus desviaciones típicas. Además, también analizaremos en un intervalo diario de tiempo cuantos diezminutales se salen del intervalo de confianza generado diariamente. De esta forma veremos si a lo largo del tiempo la media se desvía en un bin de viento concreto, o bien si las desviaciones son muy elevadas. Analizando estos fenómenos podremos decir que el aerogenerador toma lecturas muy dispares y puede ser resultado de un funcionamiento incorrecto, origen del desgaste de los componentes electrónicos o mecánicos. En la Tabla 17 podemos ver un ejemplo de las lecturas.

Tabla 17. Mediciones del día 1 y 2 de enero. Rendimiento.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID de aero | Fecha | Intervalo m/s | Media del intervalo | Desviación | Predicción |
| 1 | 1/1/2018 | [6,6.5) | 431 KW | 17 KW | A determinar |
| 2 | 1/1/2018 | [6,6.5) | 446 KW | 27 KW | A determinar |
| 3 | 1/1/2018 | [6,6.5) | 423 KW | 5 KW | A determinar |
| 1 | 1/1/2018 | [6,6.5) | 694 KW | 250 KW | A determinar |
| 2 | 1/1/2018 | [6,6.5) | 425 KW | 19 KW | A determinar |
| 3 | 1/1/2018 | [6,6.5) | 411 KW | 4 KW | A determinar |
| 1 | 2/1/2018 | [6.5,7) | 499 KW | 37 KW | A determinar |
| 2 | 2/1/2018 | [6.5,7) | 548 KW | 29 KW | A determinar |
| 3 | 2/1/2018 | [6.5,7) | 504 KW | 28 KW | A determinar |
| 1 | 2/1/2018 | [6.5,7) | 438 KW | 150 KW | A determinar |
| 2 | 2/1/2018 | [6.5,7) | 584 KW | 64 KW | A determinar |
| 3 | 2/1/2018 | [6.5,7) | 501 KW | 52 KW | A determinar |

Fijándonos en la Tabla 17 podemos sacar las siguientes conclusiones:

* El mejor aerogenerador en el bin 6 es el número 3, puesto que sus desviaciones son muy pequeñas, y no se percibe ninguna anomalía en su sistema.
* El aerogenerador 1 del día 1 al 2 ha ocurrido algo extraño en su comportamiento debido a que la dispersión de un día a otro ha aumentado considerablemente. Puede que sea causa de un fallo mecánico que esté empezando a dar la cara en ese momento,
* La lectura de las medias es normal puesto que se asemejan dentro de lo que cabe a la curva de potencia teórica.
* En el bin 6.5 podemos ver que también hay subida en la desviación del aerogenerador 1 de un día para otro, definitivamente esto es una señal de que algo no está funcionando como debiera ya que en ambos bines se da este fenómeno.

En caso de querer saber la causa de estas mediciones tan poco comúnes en el aero 1, tendríamos que comprobar nuestras conclusiones con el log de errores para ver que fallos mecánicos se dan en un futuro, y sacar alguna relación, por ejemplo, con alguna parada, o con la rotura de algún componente.

* **Patrón de Productividad**

En cuanto a lo que productividad se refiere, enfrentamos dos factores muy importantes que son la potencia producida y el tiempo. Esta vez, como hemos comentado anteriormente, el factor geográfico y la posición de los aerogeneradores es muy relevante para encontrar un criterio que nos permita decir cuál es más productivo que otro. En la Ilustración 22, vemos lo que producen 5 aerogeneradores a lo largo de un día. Como podemos observar sus líneas de tendencia lineales indican cual es más productivo el uno con respecto el otro. Sin embargo, estas mediciones son dependientes del viento. Por tanto, el que reciba más viento, es el que más produce.

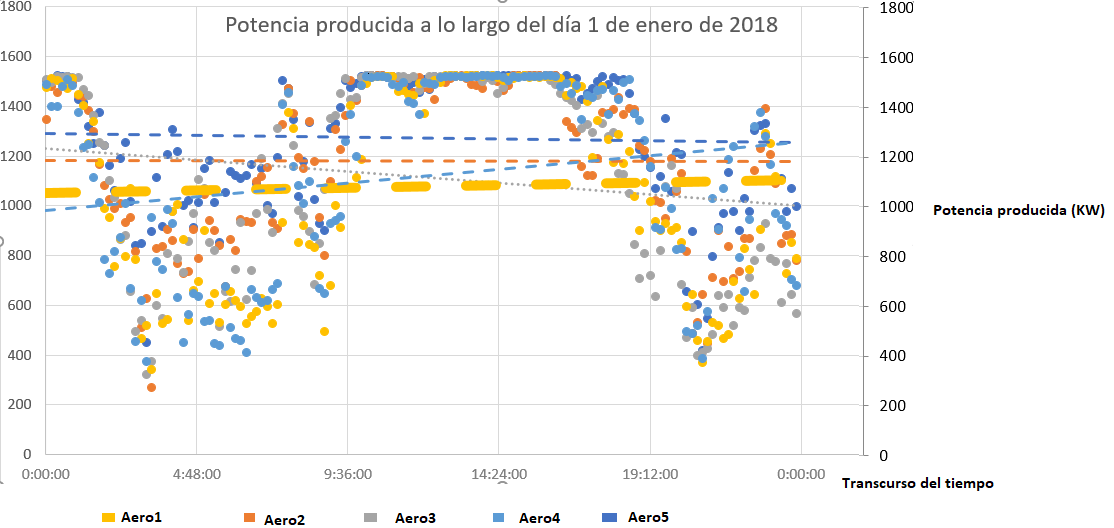


Ilustración 22. Grafíco de potencia producida el día 1 de enero.

Dicho lo anterior, hemos de incorporar el estudio del contexto de cada generador, tanto el factor geográfico, como su lugar dentro de la formación del parque. Debido a estos dos criterios, en condiciones normales, todos y cada uno de los aerogeneradores están destinados a ser unos más productivos que otros. Por lo tanto, no podemos hacer un análisis cuantitativo únicamente fijándonos en los valores numéricos de su producción, obviando la situación dentro del parque del aerogenerador.

Por tanto, lo que haremos es analizar los desplazamientos dentro de ese orden al que están destinados a producir cada uno de los aerogeneradores. Para aclarar este patrón diremos lo siguiente.

Si lo representamos en un evento,

tendría la siguiente forma:

Veremos pues que los elementos del evento quedan ordenados, según los datos obtenidos. Supongamos que el día 2 de enero se obtiene, el siguiente evento:

Como podemos observar dentro de este ranking, de un día para otro, el aero 3 ha bajado una posición. Esa posición a nivel mecánico puede significar algún tipo de desperfecto en la multiplicadora, una subida de temperatura que no debiera de darse, etc. Esto puede ser un evento de interés a ser analizado. La idea es que el algoritmo revele estos desplazamientos dentro del ranking para hacer predicciones a corto-medio plazo.

Otro ejemplo lo podemos ver en la Tabla 18, vemos un ejemplo de las mediciones de los días 1 y 2 de enero.

Tabla 18. Mediciones del día 1 y 2 de enero. Productividad.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID de aero | Día | Potencia producida (en día) | Nº Eventos representativos | Media del aero (KW/h) | Posición ranking |
| 1 | 1 de enero | 4200 KW | 3 | 1400 | 2º |
| 2 | 1 de enero | 3000 KW | 2 | 1500 | 1º |
| 3 | 1 de enero | 3000 KW | 3 | 1000 | 3º |
| 1 | 2 de enero | 4100 KW | 3 | 1366 | 1º |
| 2 | 2 de enero | 3200 KW | 3 | 1066 | 3º |
| 3 | 2 de enero | 4000 KW | 3 | 1333 | 2º |

Supongamos que el orden de producción es el conjunto ordenado descendientemente por productividad. Veremos que el día uno de enero no cumple con el orden ideal, podremos decir que el parque ha producido lo que se esperaba el día 1 de enero, puesto que . Sin embargo, si analizamos el día dos, vemos que . Si analizamos R2 y R3, vemos que la productividad del aero 2 ha bajado muchísimo, esto no debería de pasar, puesto que el aero 2 en condiciones normales, debería de estar de los primeros. Esta bajada, puede ser señal de que hay algún fallo mecánico, alguna reparación, un evento inesperado etc. En base a las traslaciones de esta especie de ranking podremos implantar medidas preventivas para optimizar la producción.

* + - 1. Implementación de los patrones.

El origen de los datos proviene de los eventos simples “WindEvent”. Estos eventos simples tienen 75 campos. Sin embargo, tal y como vemos en la Ilustración 23 solo utilizaremos el campo del viento, la potencia producida, la fecha de la lectura de datos y el System Number que es el identificador de cada aerogenerador.

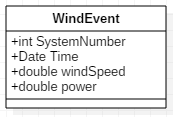


Ilustración 23. Evento simple WindEvent.

Los patrones que hemos descrito anteriormente son el fruto de la combinación de varios scripts. Son los siguientes:

* **Productividad**

Han hecho falta en total 5 patrones para lograr el análisis de lo descrito anteriormente. Se complementan los unos a los otros según el flujo de datos seleccionado en la cláusula “FROM”.

**RankingProd**. Este patrón recoge una ventana temporal de un día completo de los eventos simples Wind Event tal y como vemos en la Ilustración 24. Estos eventos complejos RankingProd son agrupados por identificador, de manera que diariamente desplegaremos un evento RankingProd por cada aerogenerador con las lecturas de la suma de todos sus diez minutales, el número de diez minutales que se han recogido, el identificador del aerogenerador, la marca de tiempo del último evento que ha entrado en la ventana temporal, y la productividad, que es la suma de la potencia producida entre el número de eventos recogidos por cada aerogenerador, es decir la media de la potencia producida en Kilowatio por hora.

Es importante destacar, que al contrario que los patrones de rendimiento, aquí no tenemos en cuenta el viento que sopla ya que normalizamos el resto de las mediciones y solo nos fijamos en las transiciones del ranking. Finalmente decimos que despliegue estos eventos complejos en orden, formando un ranking diario donde el primer evento complejo RankingProd será el mejor, y el último será el peor.

**@Name("RankingProd")**

INSERT INTO **RankingProd**

SELECT SUM(power) as suma,

count(systemNumber) as events,

systemNumber,

MAX(time) as time,

SUM(power)/count(systemNumber) as productividad

FROM WindEvent.win:time\_batch(1 day) t

GROUP BY systemNumber

ORDER BY productividad DESC;

Ilustración 24. Versión 1.0 de RankingProd.

***CambioInicio y CambioSegundo***. Esta parte del patrón para detectar la productividad es una solución poco escalable, ya que no es amoldable para parques con distinto número de aerogeneradores. Debido a falta de tiempo en este sprint, y la dificultad del aprendizaje del lenguaje tuvimos que optar por esta solución, y en un futuro idear un mecanismo para generar un patrón para cada parque. El número de líneas de este patrón varía según el número de aerogeneradores del parque, es decir un parque con 30 máquinas, necesita un patron “CambioInicio” de 100 líneas de código aproximadamente. Podemos ver un ejemplo, suponiendo que el parque a analizar solo tiene 3 aerogeneradores, en la Ilustración 25 y la Ilustración 26.

Como vemos, este patrón captura 3 eventos RankingProd consecutivos con otros 3 tras pasar 1 día.

**@Name("CambioInicio")**

INSERT INTO **CambioInicio**

SELECT a1.systemNumber,

a2.systemNumber,

a3.systemNumber,

a4.systemNumber,

a5.systemNumber,

a6.systemNumber

FROM PATTERN [ every (a1 = RankingProd()->

a2 = RankingProd() ->

a3 = RankingProd() ->timer:interval(1 day) ->

a4 = RankingProd() ->

a5 = RankingProd() ->

a6 = RankingProd())

]

WHERE (a1.systemNumber = a4.systemNumber AND

a2.systemNumber=a5.systemNumber AND

a3.systemNumber=a6.systemNumber)!=TRUE;

Ilustración 25.Versión 1.0 de CambioInicio.

**@Name("CambioSegundo")**

INSERT INTO **CambioInicio**

SELECT a1.systemNumber,

a2.systemNumber,

a3.systemNumber,

a4.systemNumber,

a5.systemNumber,

a6.systemNumber

FROM PATTERN [every (a1 = RankingProd(a1.time.toMillisec()>978314400000)->

a2 = RankingProd() ->

a3 = RankingProd() -> timer:interval(1 day)->

a4 = RankingProd() ->

a5 = RankingProd() ->

a6 = RankingProd())

]

WHERE (a1.systemNumber = a4.systemNumber AND

a2.systemNumber=a5.systemNumber AND

a3.systemNumber=a6.systemNumber)!=TRUE;

Ilustración 26. Versión 1.0 de CambioSegundo.

En otras palabras, cogemos los rankings de dos días consecutivos y comparamos si ha habido algún cambio. Si sólo tenemos CambioInicio funcionando, no se comparará todos los días consecutivos, por ello, es necesario CambioSegundo, para analizar todos y cada uno de ellos. Podemos ver en la Ilustración 27, un ejemplo de los eventos complejos RankingProd que son capturados por cada patrón. Para hacer ello, lo único que deberemos de hacer es declarar en CambioSegundo una condición que evite coger el primer ranking que se da.



Ilustración 27. Ejemplo de utilizanción de CambioInicio y CambioSegundo.

**Transición***.*Tal y como vemos en la Ilustración 28, en este patrón también tenemos el problema que hemos nombrado anteriormente en CambioInicio y CambioSegundo. Lo que hacemos aquí es revisar en que posición se encuentra cada uno de los aeros y ponemos si se ha quedado en la misma posición (0) o si han subido o bajado X número de puestos.

**@Name("Transicion")**

INSERT INTO **Transicion**

SELECT a4.systemNumber as primero,

(case when a4.systemNumber= a1.systemNumber then 0

when a4.systemNumber= a2.systemNumber then 1

else 2 end) as subida1,

a5.systemNumber as segundo,

(case when a5.systemNumber= a1.systemNumber then -1

when a5.systemNumber= a2.systemNumber then 0

else 1 end) as subida2,

a6.systemNumber as tercero,

(case when a6.systemNumber= a1.systemNumber then -2

when a6.systemNumber= a2.systemNumber then -1

else 0 end) as subida3

FROM CambioInicio;

Ilustración 28. Versión 1.0 de Transición.

**FluctuaProd**. Este patrón sirve para poder determinar si ha habido una bajada crítica dentro del ranking. La máxima bajada que se puede dar en este parque es de dos posiciones, por tanto, el patrón será el correpondiente a la Ilustración 29.

**@Name("fluctuaProd")**

INSERT INTO **fluctuaProd**

SELECT \*

FROM PATTERN[every (a1 = Transicion(a1.subida1=-2 OR a1.subida2=-2 OR a1.subida3=-2))];

Ilustración 29. Versión 1.0 de FluctuaProd.

En la Ilustración 30, podemos ver un diagrama de clases que esquematiza cuales son las dependencias entre los patrones que analizan la producividad de los aerogeneradores. Hemos de añadir, que son relaciones de dependencia, y no de herencia, ni de asociación, ya que no podemos considerar un patrón subtipo de otro, ni contienen reglas semánticas que relacionen el contenido de los eventos complejos desplegados.

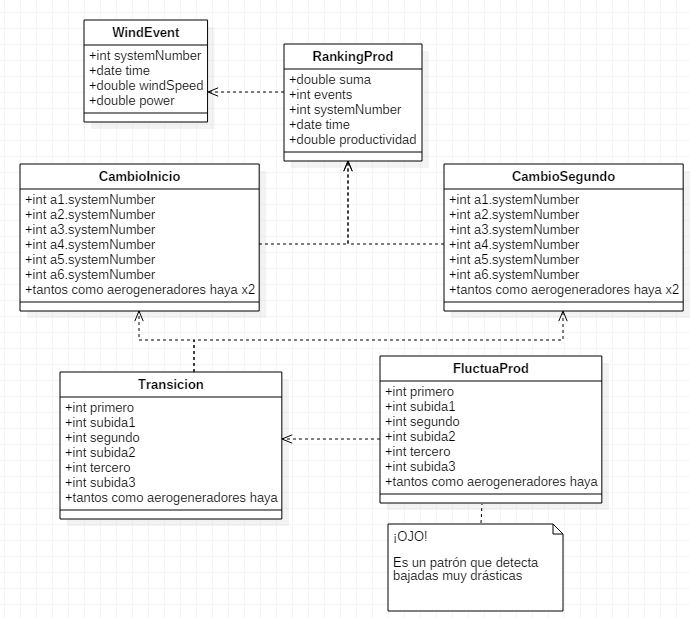


Ilustración 30. Diagrama de clases de los patrones de productividad V.1.0.

* **Rendimiento**

Para llevar a cabo el análisis de rendimiento que hemos descrito en el punto anterior hemos empleado en total 4 patrones. La primera versión de los patrones para analizar el rendimiento de la curva de potencia es la siguiente:

**WindEventT.** Este patrón agrupa por identificador y por intervalo de tiempo unitario todas las mediciones de los aerogeneradores, y diariamente se obtienen la información de la media de la potencia producida, su desviación típica, la marca de tiempo del último evento WindEvent que ha entrado en la ventana temporal diaria, así como el número de diez minutales que se han agrupado. Podemos ver el código de este patrón en la Ilustración 31.

**@Name("WindEventT")**

INSERT INTO WindEventT

SELECT systemNumber ,

stddev(power) as desv,

MAX(time) as time,

AVG(power) as rend,

COUNT(\*) as veces,

Math.floor(windSpeed) as viento

FROM WindEvent.win:time\_batch(1 day) t

GROUP BY systemNumber,Math.floor(windSpeed)

HAVING COUNT(\*)>0 ;

Ilustración 31. Versión 1.0 de WindEventT.

**BajadaDe10.** En este patrón detectaremos dos eventos consecutivos de tipo WindEventT de un mismo aerogenerador de un intervalo de viento concreto (unitario), de forma que el patrón se activará cuando el segundo de ellos tenga una bajada del 10 % con respecto el anterior. Es decir, si de un día para otro ha habido una bajada del 10 % en los Kilowatios hora de ese aerogenerador en un intervalo unitario de viento concreto. Podemos ver el código en la Ilustración 32.

**@Name(BajadaDe10)**

INSERT INTO BajadaDe10

SELECT a1.systemNumber as id1,

a1.rend as rend1,

a1.time as time1,

a1.viento as viento1,

a2.systemNumber as id2,

a2.rend as rend2,

a2.time as time2,

a2.viento as viento2

FROM PATTERN [

every a1=WindEventT() ->

a2=WindEventT(a1.systemNumber =a2.systemNumber AND

a1.viento = a2.viento AND a1.rend\*0.9 > a2.rend)]

order by a1.time;

Ilustración 32. Versión 1.0 de BajadaDe10.

**WindEventR.** Este patrón sirve para ayudar a obtener aquellos puntos que son ubicados fuera de los intervalos de confianza producidos diariamente por cada unidad de viento. Estos puntos son denominados con el nombre “outliers”. En una ventana temporal de 1 día se obtienen las medias y las desviaciones típicas de cada aerogenerador en cada intervalo de viento. Sin embargo, ahora no sólo obtendremos eso, si no que se desplegarán tantos eventos complejos WindEventR como WindEvent haya capturado la ventana diaria. Podríamos decir que se desplegará otra vez esos eventos simples, pero con la medición de la desviación y la media en dos de sus campos. Es decir, si en un día se han capturado 4464 eventos simples WindEvent, se desplegarán a medianoche 4464 eventos complejos WindEventR. En la Ilustración 33, podemos ver el código para su implementación.

**@Name("WindEventR")**

INSERT INTO **WindEventR**

SELECT power,

systemNumber,

AVG(power) as media,

STDDEV(power) as desviacion,

windSpeed,

viento,

count(systemNumber) as veces

FROM WindEventTr.win:time\_batch(1 day) ventana

GROUP BY systemNumber, viento;

Ilustración 33. Versión 1.0 de WindEventR.

**Fuera.** Tal y como vemos en la Ilustración 34 estos eventos complejos WindEventR serán capturados por el patrón Fuera. Este patrón captura todos los eventos que se despliegan a la misma hora con una ventana temporal de un 1 segundo, agrupándolos por aerogenerador y por intervalo de viento. De esta forma contaremos con “vecesFuera” los outliers detectados, es decir, los puntos que se han quedado fuera del intervalo de confianza generado por su media y su desviación típica.

**@Name("Fuera")**

INSERT INTO **Fuera**

SELECT systemNumber,

count(systemNumber) as vecesFuera,

viento

FROM WindEventR.win:time\_batch(1 sec) ventana

WHERE power<media-desviacion OR power>media+desviacion

GROUP BY systemNumber,viento

HAVING count(systemNumber)>0

order by viento;

Ilustración 34. Versión 1.0 de Fuera.

Al igual que en el caso del análisis de la productividad, podemos ver en la Ilustración 35 el diagrama de clases que esquematiza las dependencias entre los patrones que analizan el rendimiento de los aerogeneradores.

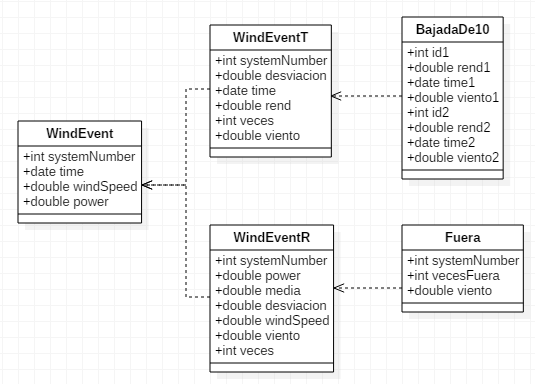


Ilustración 35. Diagrama de clases de los patrones de rendimiento V.1.0.

Para comprobar el correcto funcionamiento de estos patrones se han comprobado utilizando el simulador que ofrece Esper en su página web.

* + - 1. Implementación del ESB

Tras el feedback de la reunión al finalizar el sprint primero, llegamos a la conclusión de que debíamos reducir las dependencias con ThingSpeak. Por ello redifinimos los flujos implementados en el ESB de Mule. El resultado es el siguiente.

**Flujo del input.** En este flujo de datos vemos como el programa recoge los datos a través de un servicio FTP en la propia máquina del ESB. A través de este servicio, podemos introducir un fichero csv en ese directorio (/resources/input) para que después pasé a formato String y sea dividido en registros apilados. Estos registros son pasados a un transformador de eventos. Pasaría de un formato String a un Hashmap con 75 claves asemejándose con el modelo del aerogenerador. A continuación, mostraremos su contenido por consola a través del logger y lo enviaremos al motor CEP a través de “Send Wind Event to Esper Engine”. Podemos ver este flujo en la Ilustración 36.

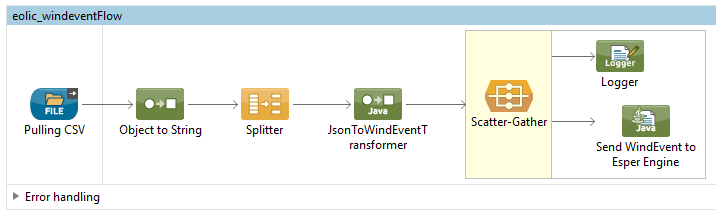


Ilustración 36. Flujo del input V 1.0.

En este flujo de Mule, se implementa el filtrado de datos para poder determinar qué diez minutales son considerados ruido. Llamamos ruido, a aquellos diez minutales que tienen mediciones que contaminan nuestro análisis, ya que contienen valores de potencia producida negativa o bien, reflejan unos kilowatios hora que no se ajustan con lo que se debe de producir en esos bines de viento en condiciones normales. Por tanto, consideraremos como ruido, situaciones de parada, productividad negativa, y mediciones de producción poco estables. Para ello, realizaremos una criba estadística de diez minutales. Se ha entrenado una curva de potencia con los datos de todo un año, para obtener la media y la desviación típica de los Kilowatios hora producidos en cada bin de viento.

Si analizamos la población de todos los diez minutales en términos estadísticos, podríamos decir que se trata de una distribución normal. Obtenidas la media y la desviación típica de la potencia producida para cada bin de viento, podremos establecer un intervalo de confianza para cada uno. De forma orientativa, si analizamos la Ilustración 37, vemos que el 68% de los diez minutales que se generan en condiciones normales, son aquellos que están más próximos a la media.



Ilustración 37. Campana de Gauss de la potencia producida en un aerogenerador

Por tanto, aquellos que más se alejan de la media de este bin, son diez minutales anómalos, y podemos considerarlos como “Ruido”, ya que no se ajustan a la media de potencia producida para ese bin de viento. El filtrado de datos considerará como ruido aquellos diez minutales cuya potencia producida no esté contenida en el intervalo para el bin de viento correspondiente. Así solo descartaremos aquellos datos que sean especialmente anómalos.

**Adición de patrones.** Tal y como vemos en la Ilustración 37, el objetivo de este flujo es desplegar los patrones en el motor CEP. En este flujo podemos ver al igual que el anterior, cómo se comienza con un servicio FTP donde introduciremos un archivo con extensión “.epl” los patrones. Estos patrones se pasarán a formato String, para que el motor compruebe su sintáxis, en caso de estar correctamente implementados, se desplegarán al motor, en caso contrario, mostraremos un aviso por la consola del ESB indicando el error.

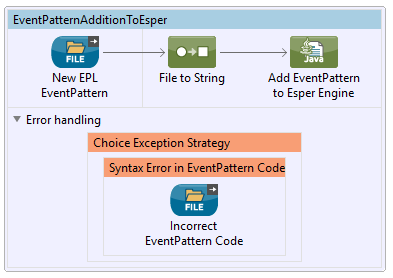


Ilustración 38. Flujo de adición de patrones V 1.0.

**Despliegue de eventos complejos**. El objetivo de este patrón como podemos ver en la Ilustración 38, es mostrar el output de los patrones. Es decir, despliega los eventos complejos correspondientes y se producen las siguientes acciones:

* Son mostrados por el logger de consola del ESB.
* Son mostrados por el logger de Java del ESB que se ha implementado.
* Son enviados a [enriquebs96@gmail.com](mailto:enriquebs96@gmail.com) por correo electrónico.
* Los datos obtenidos por los eventos complejos serán insertados en la base de datos. Solo se insertarán aquellos eventos complejos de tipo RankingProd, CambioInicio, CambioSegundo, Transicion, y FluctuaProd. Aquellos que no sean de este tipo, se volverán a mostrar a través del logger del ESB con el aviso ¡WARNING!

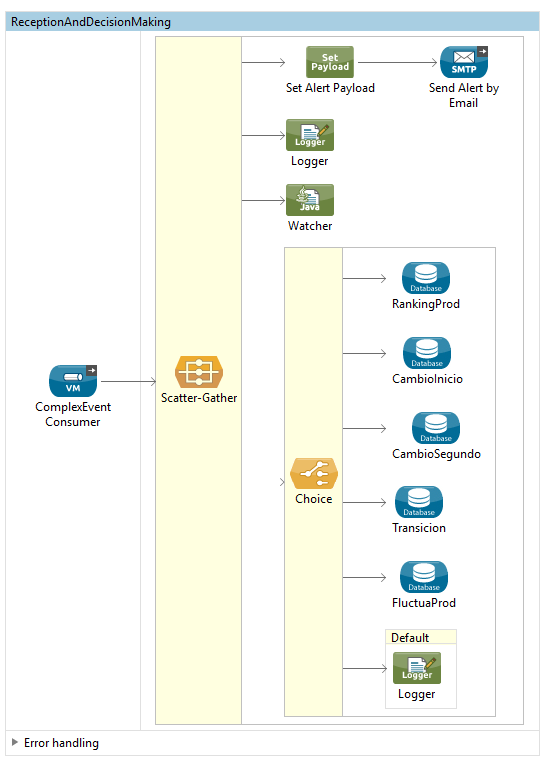


Ilustración 39. Flujo del despliegue de eventos complejos V 1.0.

* + - 1. Sprint Burndown

Como podemos ver en la Ilustración 39, queda reflejado el Sprint Burndown de esta iteración. La estimación de horas se ajusta aproximadamente a la real, ya que las estimadas erán 149, y las reales tal y como vemos en la pendiente ideal del Burndown es de 160 horas.

En esta iteración ha habido una desviación en la estimación debido a pequeños problemas de calidad en la documentación, y porque se intentó depurar el patrón de productividad, aunque no se consiguió con éxito.



Ilustración 40. Sprint Burndown Adaptación al caso real.

* + 1. Reuniones al finalizar el sprint

Esta reunión sirvió para dar fin a la segunda iteración para revisar los prototipos y la documentación correspondiente. Además, se llevó a cabo la retrospectiva para ver qué hemos hecho bien y qué se puede mejorar de cara al proyecto.

* Dirección: Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A.
* Hora y fecha de la reunión. A las 16:30 el día 29 de marzo de 2018.

Los participantes en esta reunión fueron:

* Francisco José Polo. Enca Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A. encargado del departamento de I+D+i en el año 2018.
* Antonio Fernández Diez. Ingeniero Industrial. Especializado en placas fotovoltaicas.
* Ignacio García de Carellan Esteban-Infantes. Doctor ingeniero químico. Jefe R & D Manager de Ingeteam.
* Gregorio Diaz Descalzo. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález Segovia. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que tratar

Los puntos de la reunión son:

* Revisión del prototipo.
* Revisión de la documentación.
* Aprobación de los patrones.
  + - 1. Información recogida

En primer lugar, Enrique expusó todos los puntos que definen los patrones ideados, que intentan medir productividad y rendimiento. Una vez hecho esto, Francisco José dio el visto bueno a los patrones. Sin embargo, corregió algunos aspectos que eran incorrectos, mejorando así el funcionamiento del algoritmo CEP.

Dio la aprobación completa al patrón para esquematizar la productividad, le pareció interesante. Destacó que analizar cualititativamente los aerogeneradores como si fueran corredores en una carrera, normalizando las condiciones geográficas, mediante los desplazamientos en un ranking, era una buena idea.

Una de las apreciaciones más importantes a la hora de llevar a cabo el análisis, es que a la hora de estudiar el rendimiento que los intervalos de viento no fueran unitarios, si no que fueran por bines de viento. Es decir, que se analicen los intervalos que se ajustan a los bines de viento, tal que si un evento era:

,

Después de la intervención del cliente estipulamos que un evento queda representado como:

Las variaciones de potencia producida en los intervalos centrales de la curva son demasiado grandes, de tal forma que, al minimizar el intervalo por bines de viento, obtendremos más precisión en el análisis, ya que reducimos las desviaciones típicas de la potencia producida en los diferentes dominios de viento.

Además, no estaba bien estructurado el patrón para detectar la productividad ya que no era nada dinámico para diferentes parques. Por ello, se insistió en volver a implementarlo de forma que fuera el mismo patrón para todos los parques, independientemente del número de aerogeneradores. A pesar de estos dos detalles, el prototipo cumplía con las expectativas y podríamos pasar al desarrollo de la plataforma gráfica.

La documentación estaba completa pero no estaba bien escrita, de forma, que Scrum Master y Product Owner exijieron que se revisará la redacción cuidadosamente para que quede claro que se quiere explicar en cada momento.

* + - 1. Próximos pasos

Se han acordado la incorporación de las siguientes historias de usuario para el siguiente sprint:

* **Aprendizaje de CakePHP**. En esta historia de usuario elaboraremos varios tutoriales de CakePHP para aprender a utilizar el framework de manera más correcta que como hemos hecho en el sprint anterior. Deberemos de elaborar una primera app con un formulario para poder aprender la interacción con la base de datos. Además, deberemos de aprender a usar el comando “cake bake all”, que genera de forma automática el esqueleto de la aplicación a partir del diseño impuesto en MySQL.
* **Diseño y arquitectura**. Deberemos de elaborar el diseño de la aplicación, así como un boceto de las primeras interfaces para que sean validadas, y varios diagramas UML para poder esquematizar tanto la estructuración de los patrones implementados, como de la base de datos generada.
* **Implementación de la aplicación gráfica**. Tendremos que implementar la aplicación gráfica utilizando CakePHP, deberá de abarcar tanto la pantalla de inicio, como las correspondientes para ver las salidas de los patrones de productividad y rendimiento. La idea es que la aplicación, sea un portal web amoldable en Ingeboards, de forma que no tendremos que implementar ni la gestión de usuarios ni los sistemas de seguridad.
* **Documentación**. Deberemos de generar toda la documentación correspondiente con el siguiente sprint. También deberemos de repasar concisamente la redacción de todo lo que llevamos escrito, y corregirlo para que se entienda adecuadamente.
* **Retoque de patrones**. Corregir los patrones según lo dicho en esta reunión.
  + 1. Retrospectiva del Sprint

Una vez hecha la reunión de revisión del Sprint, el Product Owner abandonó la sala, para que el Team, junto con el Scrum Master, tuvieran la retrospectiva. El objetivo de esta reunión fue responder a las siguientes preguntas:

**¿Qué salió bien en la iteración?**

* Se logró enunciar varios patrones útiles a pesar de no tener identificado como tal los patrones desde un primer momento.
* El prototipo cumplió con las expectativas, pese a las deficiencias de los patrones, ya que lograba mostrar una salida correcta.
* Pudimos llevar a cabo un pequeño avance de la aplicación gráfica, de manera que ahorraremos tiempo y podremos preguntar dudas al equipo de Procesos de Ingeteam, en caso de que nos atasquemos en algún momento.
* Ha habido muy buena comunicación entre los Stakeholders del proyecto.
* Hemos trabajado de forma eficiente y cumplido con todos los objetivos.

**¿Qué no salió bien en la iteración?**

* La elaboración del patrón que evalúa la productividad no es nada eficiente, tiene un problema bastante grande de escalabilidad.
* Los datos fuente de la aplicación no son “.db” tal y como se dijo en un primer momento. Siguen siendo archivos “csv” los ficheros que introducimos para transmitirlos al motor CEP.
* Tenemos problemas al entender cómo funciona CakePHP.
* No nos decidimos qué pruebas vamos a llevar a cabo.

**¿Qué mejoras vamos a implementar en la próxima iteración?**

* Intentaremos respetar de mejor forma las reuniones que organicemos, ya que no son cada dos semanas, tal y como se planifico en un principio.
* Se mejorará aún más la coordinación con el grupo de Procesos de Ingeteam, ya que nos hará falta su ayuda en caso de duda con el desarrollo de la aplicación gráfica.
* Se mejorará la escalabilidad de los patrones y la eficiencia del código.
  1. Sprint 3. Desarrollo de la plataforma

En este sprint desarrollaremos la aplicación gráfica utilizando el framework de CakePHP. Además, intentaremos que los datos que se muestren en el servicio web sean de un parque real. Para llevar a cabo esto, deberemos de depurar los patrones según lo añadido en la reunión al finalizar el Sprint 2. En la Tabla 19 vemos cuanto duró el sprint y quienes fueron los participantes.

Tabla 19. Información base del sprint 3.

|  |  |
| --- | --- |
| **Fecha de inicio y fin** | 2 de abril de 2018 – 16 de mayo de 2018 |
| **Personas involucradas** | * Ignacio García de Carellán. * Gregorio Diáz Descalzo. * Antonio Fernández Diez. * Enrique Brazález Segovia. * Fernando Luján Martínez |

* + 1. Reunión de planificación del sprint

Esta reunión sirve como punto de partida para el desarrollo de la aplicación gráfica, y se repiten una vez más los objetivos de este sprint, ya pactados al terminar el segundo.

* Dirección: Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A.
* Hora y fecha de la reunión. A las 16:30, el día 6 de mayo de 2018.

Los asistentes de la reunión son:

* Ignacio García de Carellan Esteban-Infantes. Doctor ingeniero químico. Jefe R & D Manager de Ingeteam.
* Antonio Fernández Diez. Ingeniero Industrial. Especializado en placas fotovoltaicas.
* Gregorio Diaz Descalzo. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález Segovia. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que tratar

Los asuntos que se trataron en esta reunión fueron los siguientes:

* Repetición de objetivos.
* Intereses.
* Petición de interfaces candidatas.
* Planificación del sprint.
  + - 1. Información recogida

En esta reunión se pusieron de manifiesto una vez más los objetivos que se plantearon al finalizar el sprint anterior. El Product Owner sugirió por parte de su grupo de procesos llevar a cabo uno de los tutoriales de w3schools y otro de la página oficial de CakePHP. De esta forma, se aprenderá a utilizar el framework correctamente, o al menos como funciona.

Se enseñó el portal de Ingeboards para coger ideas de lo que actualmente tienen implantado. Esto sirvió para saber que tipo de estilo debería de tener la aplicación, para que a posteriori les sea más fácil integrar Eolic Event Consumer a su portal web. Scrum Master propuso que se realizaran dos interfaces a nivel de boceto que pudieran esquematizar cual sería la primera visualización tanto de los patrones encargados de analizar productividad, como los de ver el rendimiento dentro del parque.

En cuanto al contenido de la aplicación gráfica, para que se considere como completo, la aplicación deberá de contener un apartado para cada patrón implementado, es decir, que contenga los siguientes hitos:

* Análisis de la productividad en base a las traslaciones de un ranking. Que permita el análisis unicamente con las bajadas y subidas del ranking.
* Análisis de la productividad con una gráfica que permita la visualización del ranking junto con la media de la potencia producida en KW/h de cada aerogenerador.
* Análisis del rendimiento en base a la media y las desviaciones típicas de la potencia producida en KW/h en cada bin de viento.
* Análisis del rendimiento en base a los diez minutales dibujados fuera del intervalo de confianza generado por la potencia producida en KW/h en cada bin de viento.

Otra parte importante de la reunión fue la de discutir cuales iban a ser los trabajos futuros de investigación que se podrían llevar a cabo una vez terminado el proyecto. Estas líneas de trabajo futuras están enumeradas en el Capítulo 7.

Enunciados los objetivos, a continuación, el Team, se separará del resto de integrantes de la reunión para planificar el sprint.

* + 1. Planificación del sprint

La organización de este sprint queda recogida en la Tabla 20, enunciando cada una de las historias de usuario, así como sus tareas y estimación de horas de trabajo.

Tabla 20. Resumen de desarrollo sprint 3.

|  |  |  |
| --- | --- | --- |
| Historia de usuario | Tarea | Horas |
| Aprendizaje de CakePHP | Primera app modelo | 13 |
| Acoplar MySQL (abierto a MongoDB por si acaso). | 3 |
| Integrar HighCharts. | 4 |
| Plantear MongoDB o MySQL. | 4 |
| Integrar proyecto con Netbeans. | 2 |
| Retoque de patrones | Acoplar al caso real para que funcione en el ESB. | 4 |
| Afinar el tema de los timestamps. | 5 |
| Hacer dinámico y conseguir la unidad parra hacer el módulo. | 8 |
| Diseño y arquitectura | Diseñar modelo de clases UML. | 2 |
| Diagrama de paquetes de la arquitectura. | 2 |
| Preparar la arquitectura con la consola de cake. | 3 |
| Buscar plantillas y decisión de estilos. | 5 |
| Diseñar las interfaces. | 2 |
| Diagrama de patrones finales. | 2 |
| Diagramas alternativos. | 4 |
| Implementación | Ajustamos la tabla de las transiciones y completamos. | 3 |
| Generación completa de base de datos definitiva con nombre: “EolicEventConsumer” | 5 |
| Primera visión de las datetables del rendimiento | 5 |
| Primera versión de gráfica de rangos para crear curvas de potencia diarias. | 5 |
| Creación de index, add, delete y view. | 3 |
| Modificar el menú. | 2 |
| Organizar y pactar el Grid de producción. | 2 |
| Hacer formulario de introducción. | 5 |
| Asignar seguimiento de colores con javascript | 2 |
| Inyección de tablas ranking diarias. | 2 |
| Aprender a utilizar el Helper de CakePHP para hacer consultas. | 4 |
| Añadir una columna con la fecha sin el momento exacto, únicamente con el día. | 4 |
| Post que actualiza los 4 ranking. | 2 |
| Información básica del aero seleccionado. | 2 |
| Formulario y front&back de la gráfica de la productividad. | 3 |
| Organizar el seguimiento temporal de la producción con javascript. | 2 |
| Organizar el código. | 1 |
| Refinar la interfaz | 2 |
| Documentación | Planificar la iteración y crear historias y tareas. | 2 |
| Documentar y retoques posibles del capítulo 2. | 2 |
| Documentar reuniones del sprint 3. | 2 |
| Terminar arreglos de sprint 2. | 1 |
| Documentar sprint 3. | 3 |
| Amoldar diagramas con memoria. | 1 |
| ¿Qué es CakePHP? | 1 |
| Dejar zanjados capítulos del 1 al 5. | 2,5 |
| Definir tareas del sprint 2 y 3. | 2 |
| Reunión de planificación del sprint 0. | 3 |
| Refactoring del desarrollo. | 1,5 |

* + 1. Reuniones de Seguimiento

En la Tabla 21, quedan reflejadas las reuniones semanales que tuvieron lugar a lo largo de este sprint.

Tabla 21. Reuniones del sprint 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la reunión | Fecha | Participantes | Asuntos que tratar |
| Bocetos de las interfaces del servicio | 9/4/2018 | Gregorio Diaz Descalzo  Enrique Brazález Segovia  Antonio Fernández | * Validar bocetos * Aportar ideas de lo que deberían aparecer en las gráficas. * Sugerencias extras. |
| Revisión de lo hecho y ampliación del sprint. | 24/4/2018 | Gregorio Diaz Descalzo.  Enrique Brazález Segovia.  Fernando Luján Martínez.  Ignación García de Carellán.  Antonio Fernández. | * Revisión del prototipo de la aplicación. * Revisión de la documentación. * ¿Qué nos queda para terminar? * Pedir ampliación del sprint. |
| Reunión definitiva para terminar el sprint | 16/5/2018 | Gregorio Diaz Descalzo  Enrique Brazález Segovia  Antonio Fernández | * Revisión del prototipo final. * Revisión de la documentación. * Validación de la utilidad de las pantallas. * ¿Cómo justificamos finalmente las pruebas de nuestra aplicación inteligente y gráfica? |

* + 1. Desarrollo del sprint

En este punto describiremos qué artefactos se generaron en este sprint, destacando los puntos más importantes de las historias de usuario nombradas anteriormente.

* + - 1. Interfaces candidatas.

Se llevaron a cabo dos interfaces para poder orientar al Product Owner y enseñar cual era la idea del Team a implementar. Además, esto sirvió como criterio de aceptación. Las interfaces, reflejan por un lado el apartado de rendimiento, y por otro, el de la productividad.

**Productividad**. Esta interfaz queda reflejada en la Ilustración 40. Las aprecicaciones por parte del Product Owner fueron las siguientes:

* 1. El menú lateral es correcto, aunque estaría bien que, aunque no tuviera gestión de usuarios, sí que simuláramos quién es el que ha iniciado sesión en el portal.
  2. Permitir el cambio de los días y a su vez el cambio dinámico de los rankings.
  3. Permitir el seguimiento de un aerogenerador en el ranking, señalándolo en rojo o de algún color llamativo que permita ver adecuadamente las traslaciones diarias.
  4. La información del arerogenerador que estamos siguiendo deberá de reflejar: id de la base de datos, el id que lo distingue del resto a nivel de parque, el nombre que tiene en Ingeboards y la posición en la formación del parque en formato “x||y”, siendo “x” la fila, e “y” la columna.
  5. La gráfica no está bien situada ya que los parques suelen tener 30 activos normalmente, con lo que los rankings ocuparán la mayor parte de la pantalla y no se verá la gráfica. Sería mejor tener dos pestañas, una con el análisis cuantitativo a través de la gráfica, y otro cualitativo, con la disposición de los rankings.



Ilustración 41. Interfaz candidata de la Productividad.

**Rendimiento**. Corresponde con la Ilustración 41, y las aprecicaciones por parte del Product Owner fueron las siguientes:

1. No gusto la idea del cambio de parque dinámico ubicado en el menú lateral de la interfaz, ya que podríamos confundirnos en algún momento de la interación con el sistema y no saber qué parque estamos analizando. Propuso poner una interfaz inicial donde elegiríamos que día y qué parque queremos analizar y ya a partir de ahí, movernos en la lectura diaria de los aerogeneradores.
2. Propuso no tener todo en la misma pantalla, y separar cada uno de los patrones que examinan el rendimiento en varias pestañas o pantallas.
3. Alternar gráficas para representar el output de los patrones. Por ejemplo, gráficos de barras, gráficas de dispersión, gráficas de sectores, etc.
4. Acompañar siempre las gráficas con tablas con los datos recogidos de forma clara y concisa.
5. No gustó en absoluto, que se pudiera seguir el rendimiento de un aerogenerador. Pidió quitar esta opción en el apartado de rendimiento.

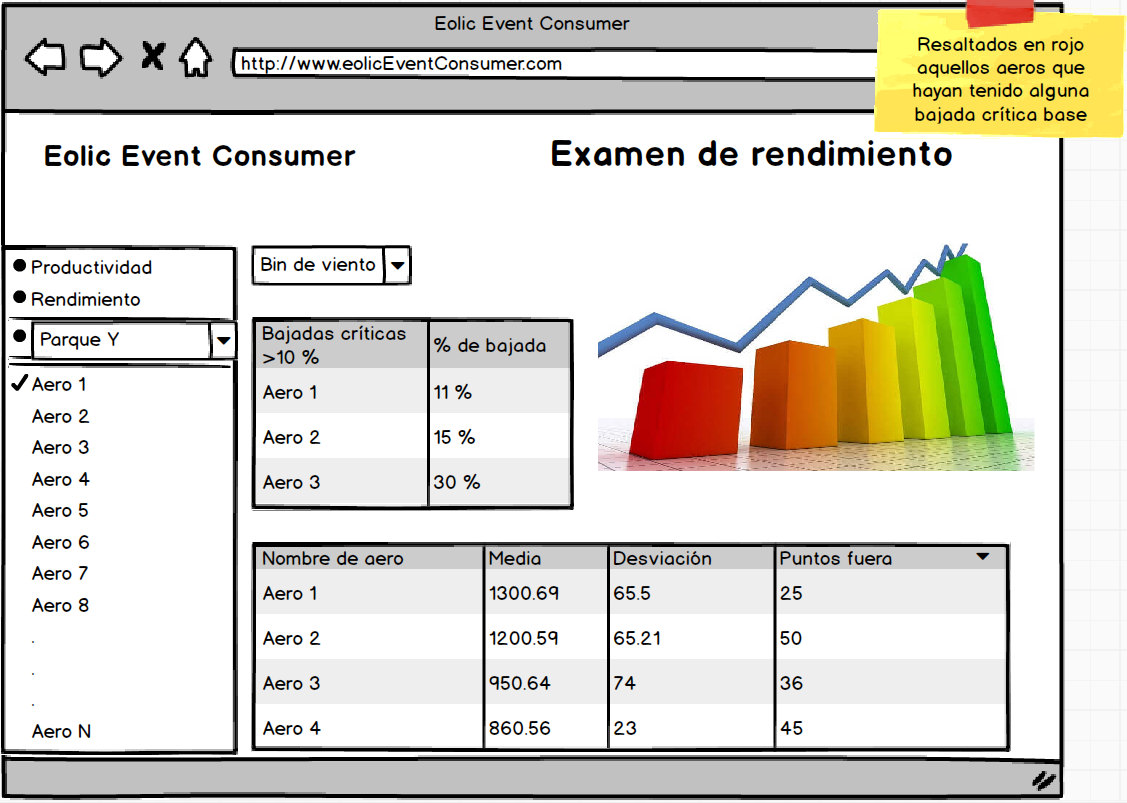


Ilustración 42. Interfaz candidata del Rendimiento.

Para concluir, se pidió expresamente que además de estos dos apartados de rendimiento y productividad, se añadiera en el menú lateral accesos a los métodos “index” que el framework de CakePHP proporciona para visualizar la información almacenada en crudo en la base de datos, sin cuidar nada la estética de estas pantallas.

* + - 1. Patrones finales.

Tras el feedback de corregir y hacer más dinámicos los patrones que analizan la productividad y unificar los del rendimiento, se han modificado para mejorar su eficiencia. La segunda versión de los patrones es la siguiente:

* **Productividad**

El funcionamiento de estos patrones ha cambiado totalmente en comparación con la versión anterior, ya que a primera vista han reducido su extensión y serán los mismos patrones para cualquier parque.

**WindEventP**. Este patrón tal y como se aprecia en la Ilustración 42, recoge todos los eventos de los aerogeneradores del parque durante un día y al igual que WindEventR (comentado en el sprint anterior) añade un campo mas a cada WindEvent con el número total de aerogeneradores.

**@Name("WindEventP")**

INSERT INTO **WindEventP**

SELECT COUNT(DISTINCT systemNumber) as numAeros,\*

FROM WindEvent.win:time\_batch(1 day) t

Ilustración 43. Patrón WindEventP. V 2.0.

**RankingProd**. A continuación, como vemos en la Ilustración 43, RankingProd genera una ventana temporal de un segundo que captura todos los eventos WindEventP que se generan diariamente. Recordemos que WindEventP, desplega los eventos complejos transcurrida la duración de la ventana temporal diaria. Es decir, toda la información que se ha desplegado cada diez minutos durante un día se vuelve a lanzar a la vez, al acabar la ventana. Finalmente, RankingProd recoge toda esa información y genera los rankings de la misma forma que en la iteración anterior. Cabe destacar dos campos nuevos que tendrá este evento complejo:

1. **Fecha**. Que es el día al que pertenece el ranking.
2. **NumAeros**. Que es el número de aerogeneradores que componen el ranking de ese día concreto.

**@Name("RankingProd")**

INSERT INTO **RankingProd**

SELECT SUM(power) as suma,

count(systemNumber) as events,

systemNumber,

MAX(time) as time,

SUM(power)/count(systemNumber) as productividad,

MIN(time.roundFloor('day')) as fecha,

numAeros

FROM WindEventP.win:time\_batch(1 sec) t

GROUP BY systemNumber,numAeros

HAVING count(systemNumber)>0

ORDER BY productividad DESC

Ilustración 44. Patrón RankinProd V 2.0.

**Escala**. Este patrón recoge cada evento RankingProd por separado y saca en que posición se ha quedado con respecto el resto, es decir, su posición en el ranking diario. Lo podemos ver en la Ilustración 44.

**@Name("Escala")**

INSERT INTO **Escala**

SELECT systemNumber as systemNumber,

count(suma) % numAeros+1 as posicion,

fecha

FROM RankingProd

Ilustración 45. Patrón Escala V 2.0.

**Transición**. Plasmado en la Ilustración 45. En el patrón Transicion, vemos como se examinan las escalas de los aerogeneradores, y se compara la posición en la que ha quedado un mismo aerogenerador en dos rankings consecutivos (2 días consecutivos), de esta forma, con la resta de sus posiciones veremos cuánto ha subido o bajado de un día para otro.

**@Name("Transicion")**

INSERT INTO **Transicion**

SELECT a1.systemNumber as systemNumber,

a1.posicion as posicionInicio,

a2.posicion as posicionFin,

a1.posicion - a2.posicion as variacion,

a1.fecha as inicio,

a2.fecha as fin

FROM PATTERN [every a1 = Escala() ->

a2 = Escala(a1.systemNumber = a2.systemNumber)

]

Ilustración 46. Patrón Transición V 2.0.

Podemos ver el diagrama de dependencias final de los patrones que analizan la productividad en la Ilustración 46.

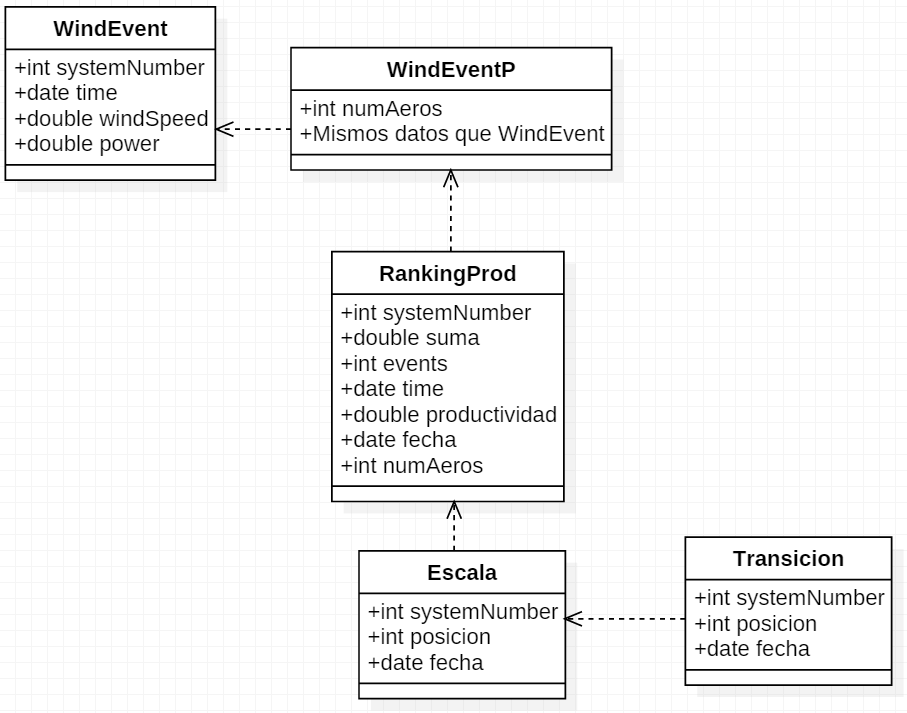


Ilustración 47. Diagrama de clases de los patrones de rendimiento V.2.0.

* **Rendimiento**

El código de los patrones de rendimiento es prácticamente igual que en la iteración anterior. Sin embargo, hay ligeros cambios para establecer la división por bines de viento. De ahí la incorporación de un nuevo patrón WindEventTr, reflejado en la Ilustración 47.

**@Name("WindEventTr")**

INSERT INTO **WindEventTr**

SELECT \*,

(case when ((windSpeed-Math.floor(windSpeed))-0.5)<0 then Math.floor(windSpeed)

else Math.floor(windSpeed)+0.5 end) as viento

FROM WindEvent

Ilustración 48. Patrón WindEventTr V 2.0.

Hay que destacar que al igual que en análisis de la productividad, se ha incorporado el campo “fecha” a los patrones, para tener claro el día del que procede cada evento complejo. Este cambio lo podemos ver en la Ilustración 48, la Ilustración 49, y la Ilustración 50. Cabe destacar que el patrón WindEventT ha sido reescrito y nombrado como EstadisticoDiario.

**@Name("WindEventR")**

INSERT INTO **WindEventR**

SELECT power,

systemNumber,

AVG(power) as media,

STDDEV(power) as desviacion,

windSpeed,

viento,

count(systemNumber) as veces,

MIN(time.roundFloor('day')) as fecha

FROM WindEventTr.win:time\_batch(1 day) ventana

GROUP BY systemNumber, viento

Ilustración 49. Patrón WindEventR V 2.0.

**@Name("EstadisticoDiario")**

INSERT INTO **EstadisticoDiario**

SELECT systemNumber,

AVG(power) as media,

STDDEV(power) as desviacion,

viento,

count(systemNumber) as veces,

MIN(time.roundFloor('day')) as fecha

FROM WindEventTr.win:time\_batch(1 day) ventana

GROUP BY systemNumber, viento

HAVING STDDEV(power)>0

Ilustración 50. Patrón EstadisticoDiario V 2.0.

**@Name("Fuera")**

INSERT INTO **Fuera**

SELECT DISTINCT systemNumber,

media,

desviacion,

count(systemNumber) as vecesFuera,

viento,

MAX(fecha) as fecha

FROM WindEventR.win:time\_batch(1 sec) ventana

WHERE power<media-desviacion OR power>media+desviacion

GROUP BY systemNumber,viento

HAVING count(systemNumber)>0

order by viento

Ilustración 51. Patrón Fuera V 2.0.

**BajadaDeRendimiento**. Finalmente “BajadaDe10”, tal y como podemos ver en la Ilustración 51, se ha reescrito para que no haya problemas con los caracteres de su nombre a la hora de introducirlo ni en el modelo de CakePHP ni en el ESB.

**@Name("BajadaDeRendimiento")**

INSERT INTO **BajadaDeRendimiento**

SELECT a1.systemNumber as systemNumber1,

a1.media as media1,

a1.fecha as fecha1,

a1.viento as viento1,

a2.systemNumber as systemNumber2,

a2.media as media2,

a2.fecha as fecha2,

a2.viento as viento2

FROM PATTERN[

every a1=EstadisticoDiario() -> a2=EstadisticoDiario(a1.systemNumber=a2.systemNumber AND a1.viento = a2.viento AND a1.media\*0.9 > a2.media)

]

order by a1.fecha

Ilustración 52. Patrón BajadaDeRendimiento V 2.0.

Podemos ver el diagrama de dependencias final de los patrones que analizan el rendimiento en la Ilustración 53.

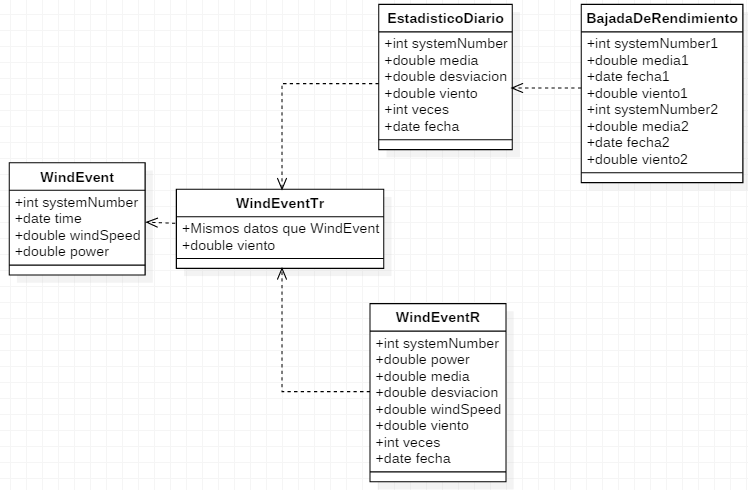


Ilustración 53. Diagrama de clases de los patrones de Productividad V.2.0

* + - 1. Diseño y arquitectura.

Eolic Event Consumer se divide en dos partes como vemos en el diagrama de paquetes de la Ilustración 53. Por un lado, una aplicación de la tecnología CEP y por otro, tenemos un portal web que despliega la información obtenida

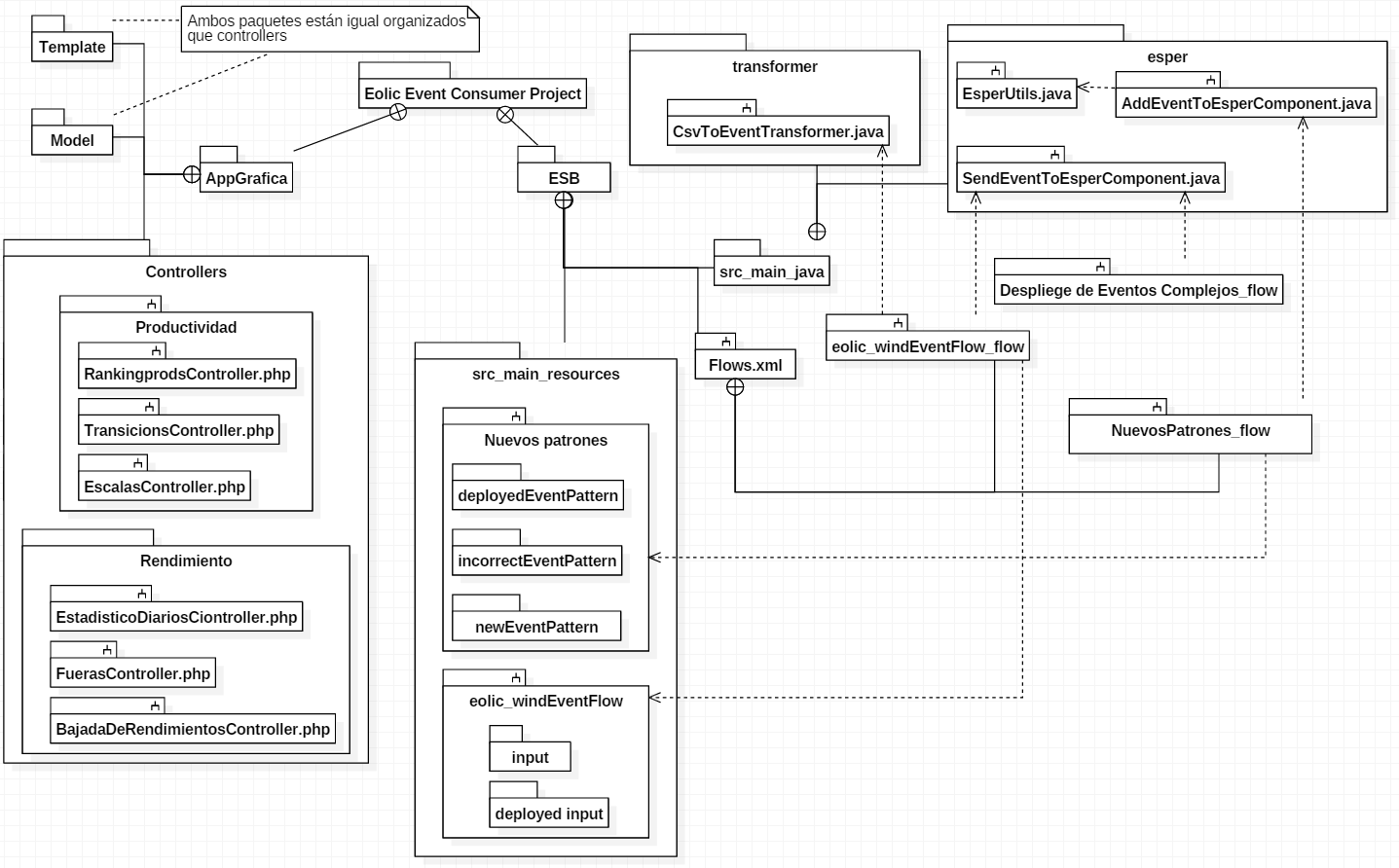


Ilustración 54. Diagrama de paquetes de EolicEventConsumer.

* **Aplicación inteligente integrada en el ESB**

La estructura del ESB se divide en tres partes. Tal y como vemos en el diagrama de paquetes de la Ilustración 53, estas partes son los flujos, las clases de java que actúan en esos flujos, y por otro lado, tenemos los directorios de entrada de información.

**Flujos finales**

Al igual que en el sprint anterior, los flujos siguen siendo tres, y el propósito de cada uno es el mismo, sin embargo, ha habido algunos cambios.

**eolicWindEventFlow***.* La funcionalidad es la misma que en el sprint anterior. Sin embargo, se ha cuidado la coherencia código-funcionalidad para mejorar su comprensión y se han hecho modificaciones en las clases de JAVA involucradas.

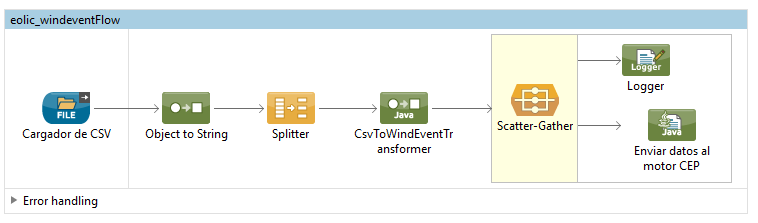


Ilustración 55. Flujo del input V 2.0

Las clases involucradas en este flujo son:

* **Clase Zona**

Esta clase representa un intervalo de viento, en esta problemática, los intervalos de viento corresponden con los bines de viento. Esta clase tiene los siguientes atributos:

* **double zone.** Corresponde con el número del bin. Es decir, si por ejemplo el valor de zone es 5, se corresponde con el bin 5, que captura los diez minutales del intervalo de viento [5,5.5).
* **double media.** Representa la media del modelo entrenado anteriormente en ese bin de viento concreto.
* **double desv**. Representa la desviación típica del modelo entrenado anteriormente en ese bin de viento concreto.
* **int N.** Representa el número de diez minutales capturados en ese bin de viento.
* **Clase CsvToWindEventTransformer**

Esta clase tiene un esqueleto interno que sigue las directrices de los transformadores Java de Mule. No contiene atributos y los métodos son los siguientes:

* + **Object transformMessage (MuleMessage message, String outputEncoding).** Este método se asemeja al método “main” de una clase tradicional. Por parámetro introducimos un mensaje de Mule, que será un String con todos los elementos de un registro “csv”. El objetivo de este método es transformarlo en un HashMap con todos los datos del registro. Además, para identificar el HashMap añadiremos el atributo “eventTypeName”, que corresponde con el nombre del evento.

La otra parte de este método es realizar un filtrado de datos, es decir, clasifica si ese evento simple es considerado ruido o no es así. Podemos ver más acerca de este criterio en el punto 5.3.4.4 . En caso de ser ruido, el “eventTypeName” será “Ruido” y al llegar al componente que envía los eventos al motor, será descartado para no influir de forma negativa en nuestro análisis. Esta clase utiliza una variable global ubicada en “mule-app.properties” denominada, “key.rules”. Esta variable contiene las medias y desviaciones típicas teóricas de lo que debería de producirse en cada bin de viento.

* + **Map<String, Object> clasificador (Map<String, Object> event, String [] registro, int i).** Clasifica cada elemento del “registro” en el diccionario de datos “event”, según el índice “i” que se obtenga. Recordemos que cada registro tiene 75 columnas, y cada una de ellas tiene un significado u otro dentro del modelo del aerogenerador.
  + **ArrayList<Zona> getListProperty (MuleMessage message).** Este método junto con el siguiente, ayuda al filtrado de datos. Se introduce por parámetro el mensaje de mule para sacar su contexto. Es decir, su configuración y propiedades del entorno. En esas propiedades, en formato String están las reglas que dictaminan si los eventos simples son considerados como ruido (key.rules). La labor de este método es obtener a través de esa variable, un ArrayList de Zonas para utilizar ese criterio.
  + **String resetProperty(Arraylist<Zona> reglas).** Este método tiene como objetivo actualizar “key.rules” en tiempo de ejecución, y llevar un conteo de cuantos diez minutales se han dado en cada bin de viento.
* **Clase SendEventToEsperComponent**

Esta clase tiene un esqueleto interno que sigue las directrices de los componentes Java de Mule. Es importante, saber que a este método se deberá de acceder en exclusión mutua para respetar la coherencia en los eventos enviados. No contiene atributos y los métodos son los siguientes:

* + **void sendEventToEsper (Map<String, Object> eventMap**). La principal función que tiene este método es mapear los eventos simples, obteniendo su “schema” y mandarlos al motor CEP. En primer lugar, comprobaremos si este evento simple ha sido considerado como “Ruido”, si no está considerado como ruido procederá a desplegarlo en la instancia del motor CEP. Si es “Ruido”, no cargará el schema, ni mandará su contenido para ser analizado. Para ver estas acciones con más detalle podemos observar el diagrama de secuencia de la Ilustración 55.
  + **Map <String, Object> getEventType (Map<String, Object> eventMap).** Este método es el encargado de mapear el evento simple introducido por parámetro para obtener su “schema” y el tipo.

Tal y como hemos explicado anteriormente, el diagrama de secuencia de la Ilustración 55, refleja la interacción entre las diferentes componentes del ESB para la inyección de información a Eolic Event Consumer.

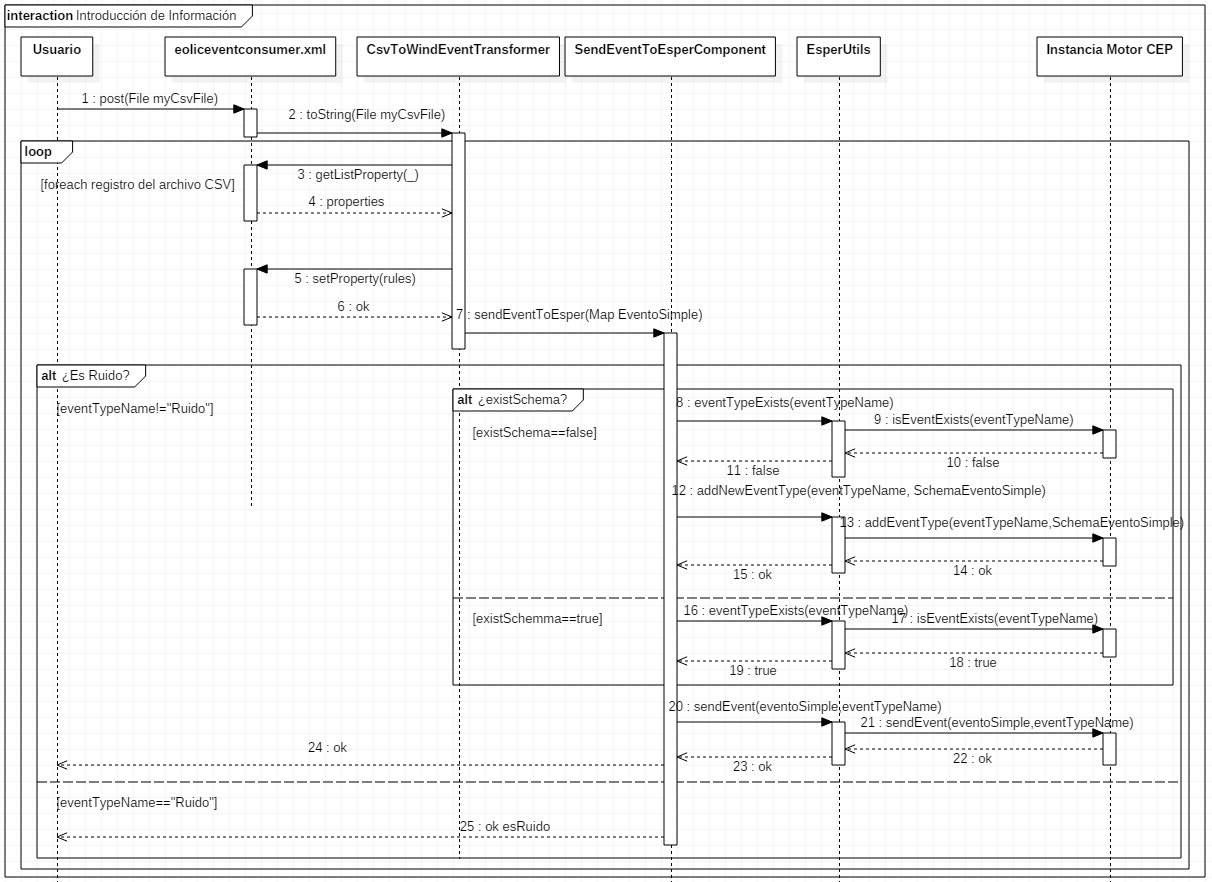


Ilustración 56. Diagrama de secuencia "Introducción de Información"

**NuevosPatronesFlow***.* La funcionalidad es la misma que en el sprint anterior. Sin embargo, se ha cuidado la coherencia código-funcionalidad para mejorar su comprensión y se han hecho modificaciones en las clases de JAVA involucradas. La versión final de este flujo es el representado en la Ilustración 56.

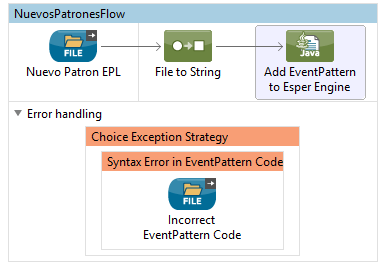


Ilustración 57. Flujo de adición de patrones V 2.0

Las clases involucradas en este flujo son:

* **Clase AddEventPatternToEsperComponent**

Sigue la estructura de un componente Java de Mule. Esta clase tiene como objetivo dar de alta nuevos patrones en el registro de la instancia del motor CEP, y asignar lo que deberá de hacer el patrón en caso de haber sido detectado. Todos los patrones deberán de encolar todos los eventos complejos detectados y enviarlos al flujo de despliegue. Los atributos son los siguientes:

* **Object payload.** Se trata del mensaje de entrada de este componente. Seria un String con el contenido del patrón.
* **EPStatement pattern.** Se trata de una instancia patrón de la librería de Esper a raíz del payload obtenido.
* **StatementAwareUpdateListener genericListener**. Cada patron tiene ligado un listener. Este listener al activarse, ejecuta su método update que tiene la siguiente forma:
  + **void update (EventBean[] newComplexEvents, EventBean[] oldComplexEvents, EPStatement detectedEventPattern, EPServiceProvider serviceProvider).** Este método se activará cada vez que los patrones detecten algún evento complejo. La finalidad de este método es que todos los eventos complejos detectados sean transformados en mensajes de Mule (con un un HashMap) y enviados a el flujo de despliegue.
* **Clase EsperUtils**

El objetivo que tiene esta clase es controlar el registro de patrones y eventos simples en el motor CEP, ayudando a AddEventPattern a gestionar el alta de patrones. Tiene los siguientes métodos:

* + **boolean eventTypeExists (String eventTypeName)**. Devuelve verdadero en caso de que exista un schema registrado en el motor CEP con el nombre igual al contenido de eventTypeName. En caso contrario, devuelve falso.
  + **void addNewEventType (String eventTypeName, Map<String, Object> eventTypeMap)**. Damos de alta en el motor CEP los eventos con el nombre “eventTypeName” correspondiente, y con el schema contenido en el “eventTypeMap”.
  + **EPStatement addNewEventPattern (String eventPattern)**. Crear la instancia de un patrón de Esper por medio de su código fuente contenido en “eventPattern”.
  + **void addListener (StatementAwareUpdateListener listener, EPStatement eventPattern)**. Este método incorpora el listener con su método “update” correspondiente, a la instancia del patrón de Esper “eventPattern”.
  + **void sendEvent (Map <String,Object> eventMap, String eventTypeName)**. Envía el schema, y el nombre de evento que se quiere dar de alta.

Podemos ver en el diagrama de secuencia de la Ilustración 57, como es la interacción con el sistema para añadir nuevos patrones. Podemos ver como se introduce el fichero, para después examinar si tiene errores sintácticos en su definición. En caso de no tener errores sintácticos, se dará de alta en el motor CEP. Sin embargo, si tiene errores, se mostrarán al usuario por consola.

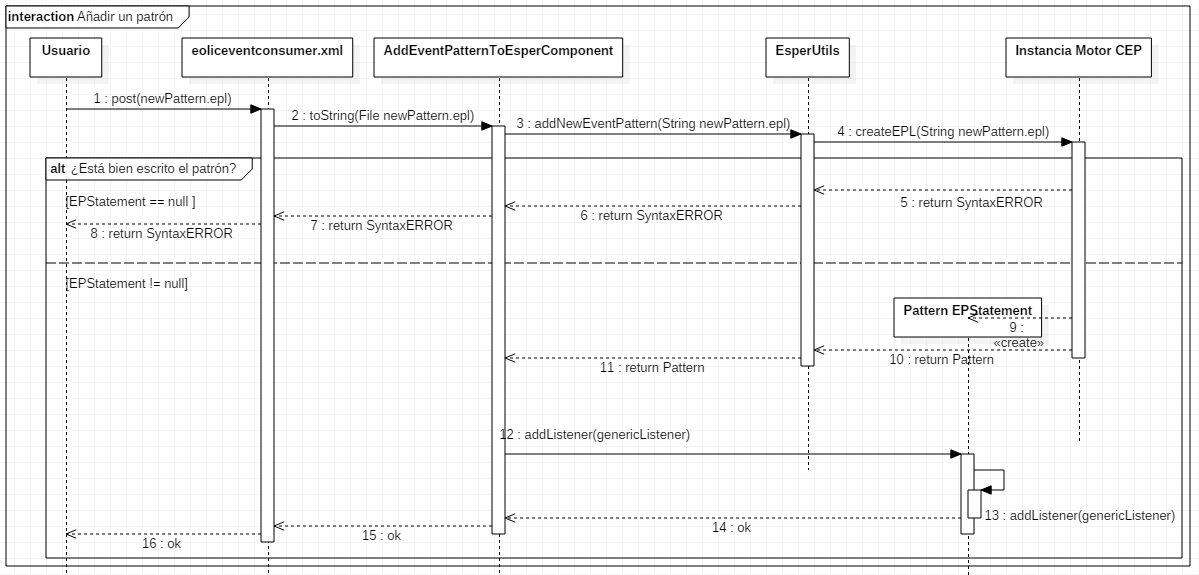


Ilustración 58. Diagrama de secuencia "Añadir un patrón"

**DespliegueEventosComplejosFlow***.* En comparación con el resto de los flujos, este si ha cambiado considerablemente con respecto el del sprint anterior. Tal y como podemos ver en la Ilustración 58, hemos quitado la funcionalidad de mandar un correo por cada evento complejo detectado ya que se van a desplegar una cantidad de ingente de eventos complejos. También el logger del Scatter Gather, para tener un feedback de la consola más limpio, y permita mejorar la trazabilidad de las acciones llevadas a cabo. Al reestructurar la definición de los patrones implementados, se ha tenido que redefinir el ingreso de información en las tablas:

* RankingProds
* Transicions.
* Escalas.
* EstadisticoDiarios.
* Fueras.
* BajadaDeRendimientos.

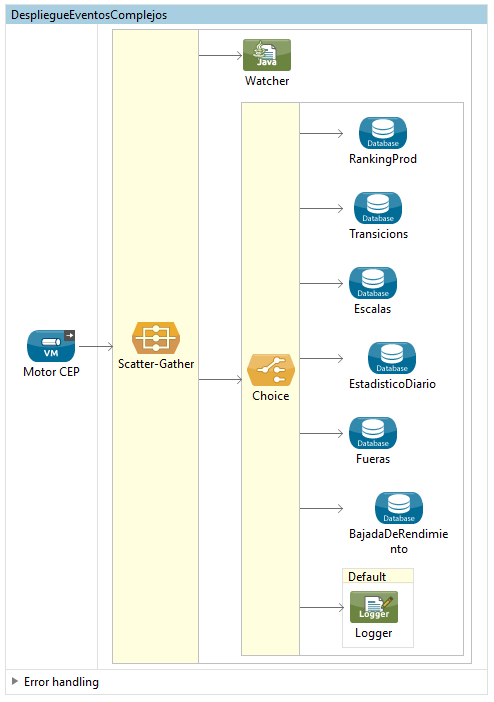


Ilustración 59. Flujo del despliegue de eventos complejos V 2.0

Los listeners creados a partir de la introducción de los patrones, son esenciales para el despliegue de los eventos complejos. Podemos ver en el diagrama de secuencia de la Ilustración 59, como se generan las instancias de los patrones, asociando un listener que especificará cómo debe de actuar al detectar un evento complejo.

Cuando se cumplen las condiciones de un patrón, los eventos complejos detectados por él son enviados uno a uno, al flujo de despliegue, a través de mensajes de Mule. Una vez en el flujo de despliegue, dependiendo del tipo de evento complejo que sea, se mostrarán a través del logger del ESB, o bien, se introducirán en la base de datos.

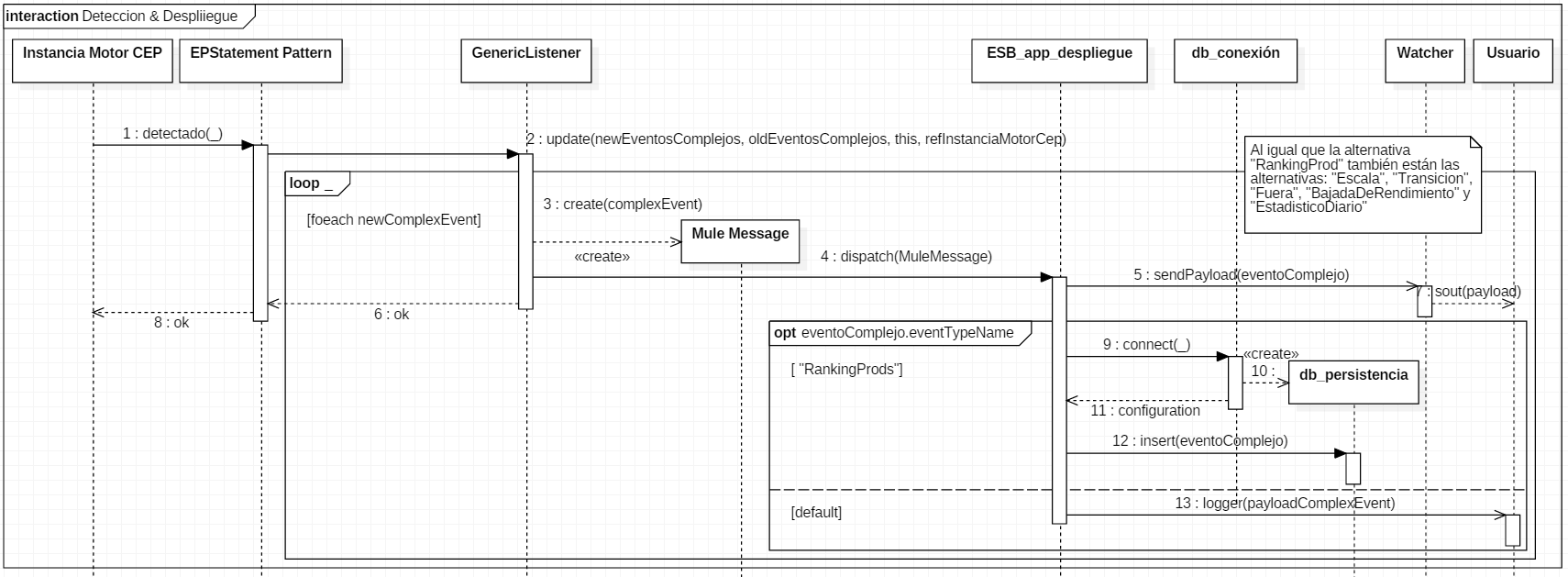


Ilustración 60. Diagrama de secuencia "Detección & Despliegue".

* **Directorios de entrada de información.**

Tal y como hemos visto en la Ilustración 53. Separamos la entrada de datos en dos grupos de directorios:

* **Input de Eventos Simples.** Este grupo de directorios está ligado al flujo de datos“eolicWindEventFlow”. La descripción de la función de los directorios es la siguiente:
  + **Input**. En este directorio se introducen los archivos “csv” con los diezminutales. En caso de ser del modelo esperado irá al directorio “deployed-input”, y en caso negativo a “incorrect-input”.
  + **Deployed-input**. Es el directorio donde se irán direccionando todos los ficheros csv que han sido introducidos correctamente en el sistema.
  + **Incorrect-input**. Es el directorio donde se irán direccionando todos los ficheros csv que no tienen el formato esperado.
* **Input de patrones***.* Este grupo de directorios es el que utiliza el flujo de la adición de patrones del sistema. La función de los directorios es la siguiente:
  + **New-EventPattern.** En este directorio se introducen los archivos con extensión “. epl”, donde quedan definidos los patrones en Esper EPL. En caso de que su sintáxis sea la correcta, se introducirán los patrones en el directorio “deployed-EventPattern”. En caso contrario, se introducirán en “Incorrect-EventPattern”.
  + **Incorrect-EventPattern.** Es el directorio donde se irán direccionando todos los ficheros introducidos en el input cuya sintáxis sea errónea.
  + **Deployed-EventPattern.** Los patrones desplegados con éxito por el motor CEP serán ubicados en este directorio.
* **Aplicación gráfica de CakePHP**

Se divide en tres paquetes principales en “src”. Sigue el clásico patrón de diseño MVC. La descripción de los paquetes es la siguiente:

* **Model**. Este es el esqueleto interno de la aplicación implementado en PHP, se genera automáticamente por consola una vez diseñada la base de datos en “phpAdmin”. Se utiliza el comando del framework “/bin cake bake all #NombreTabla” para su modelado.
* **Template**. Se trata de todo el contenido de las vistas. Tiene una extensión poco común, característica de CakePHP, “ctp”. Están implementadas a través de JavaScript, HTML, Bootstrap y PHP.
* **Controller**. Este paquete contiene el código que mueve toda la lógica de interacción con el servidor. La mayor parte de la lógica de uso está en los controladores, “RankingProdsController.php” y “EstadisticoDiariosController.php”.

El modelo de la aplicación gráfica es el reflejado en el diagrama de clases de la Ilustración 60. Podemos ver cuales son sus tablas, campos y cuales son los métodos implementados en sus controladores.

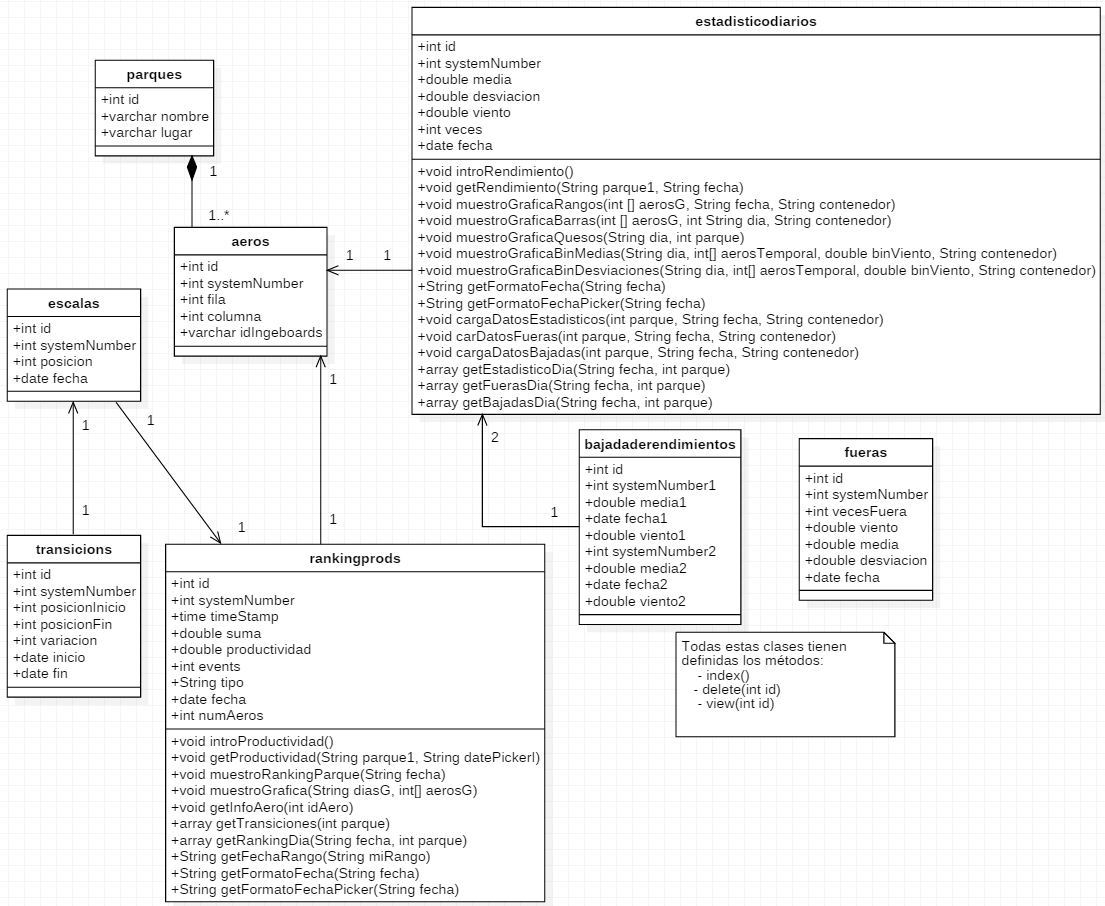


Ilustración 61. Diagrama de clases de la aplicación gráfica

La descripción de cada entidad es la siguiente:

* **Parques**. Recoge todos los parques almacenados en el sistema. Los campos son los siguientes:
  + **int id**. Id del parque de la base de datos.
  + **varchar nombre**. Nombre del parque.
  + **varchar lugar**. Lugar donde se encuentra.
* **Aeros**. Recoge todos los aerogeneradores almacenados en el sistema. Los campos son los siguientes:
  + **int id**. Id del aergogenerador de la base de datos.
  + **int systemNumber**. Id general del aerogenerador.
  + **int fila**. Suponiendo que el parque forme una matriz, este campo es la componente fila en la que se encuentra el activo.
  + **int columna**. Suponiendo que el parque forme una matriz, este campo es la componente columna en la que se encuentra el activo.
  + **varchar idIngeboards**. Se trata del nombre que tendría el aerogenerador en Ingeboards.
  + **int id\_parque**. Es la clave foránea al parque que corresponde ese aerogenerador.
* **Rankingprods**. Aquí se almacenan los eventos complejos ranking detectados por el patrón “Rankingprods”. La descripción de sus campos es:
  + **int id.** Id de ranking de la base de datos.
  + **int systemNumber**. Id general del aerogenerador.
  + **timestamp time**. Instante en el que se capturó el último evento simple “WindEvent” de la ventana temporal.
  + **doule suma**. Se trata de la suma de toda la potencia producida en ese día.
  + **double productividad**. Es la media de la potencia producida a lo largo del día por ese aerogenerador entre el número de diez minutales que se han capturado. Podemos decir que se trata de los Kilowatios por hora generados diariamente.
  + **int events**. Número de diez minutales capturados ese día.
  + **string tipo**. Nombre del patrón “RankingProd”. Esto servirá como output de que todo esta saliendo como se esperaba.
  + **date fecha**. Es el día en el que se capturaron los eventos simples “WindEvent”.
  + **int numAeros**. Número de aerogeneradores que componen el ranking de ese día.
* **Escalas**. Se trata de la tabla que almacena los eventos complejos detectados por el patrón “Escala”. Sus campos son:
  + **int id**. Id propio de la base de datos.
  + **int systemNumber**. Id general del aerogenerador
  + **int posición**. Posición dentro del ranking de ese día para el aerogenerador analizado.
  + **date fecha**. Día del año en que se capturó el evento complejo Escala que este registro representa.
* **Transicions**. Se trata de una tabla que almacena los eventos complejos que reflejan las transiciones de los aerogeneradores de un día para otro en el ranking. Los campos son los siguientes:
  + **int id**. Id propio de la base de datos.
  + **int systemNumber**. Id general del aerogenerador
  + **int posiciónInicio**. Se trata de la posición en la que se quedó el día “inicio” ese aerogenerador.
  + **int posiciónFin**. Se trata de la posición en la que se quedó el día “final” ese aerogenerador.
  + **int variación**. Aquí tenemos cuantas posiciones se ha movido el aerogenerador de un día para otro. Por ejemplo, si el valor es 0, es que mantuvo la posición. Si es 5, es que subió 5 puestos en el ranking, y si es -3, bajó 3 posiciones.
  + **date inicio**. Día del año inicio.
  + **date fin**. Día del año final (inicio y final, son dos días consecutivos en todos los casos).
* **EstadisticoDiarios**. Se trata de la tabla que almacena los datos producidos por el patron EstadisticoDiario. Los campos son los siguientes:
  + **int id**. Id propio de la base de datos.
  + **int systemNumber**. Id general del aerogenerador
  + **double media**. Media de la potencia producida entre el número de eventos recogidos en total en ese bin de viento de un aerogenerador concreto.
  + **double desviacion**. Desviación típica de la potencia producida en ese bin de viento de un aerogenerador en concreto.
  + **double viento**. Bin de viento.
  + **int veces**. Número de eventos recogidos en total en ese bin de viento de un aerogenerador en concreto.
  + **date fecha**. Día del año en que se capturó el evento complejo EstadisticoDiario que ese registro representa.
* **BajadaDeRendimientos**. Es la tabla que almacena los eventos complejos detectados por el patrón “BajadaDeRendimiento”. Supuestamente los valores de los campos deben de ser iguales exceptuando los valores de las medias y las fechas. Sus campos son:
  + **int id**. Id propio de la base de datos.
  + **int systemNumber1**. Id general del primer aerogenerador.
  + **double media1**. Media del EstadísticoDiario primero capturado.
  + **date fecha1**. Fecha del EstadisticoDiario primero capturado.
  + **double viento1.** Bin de viento del EstadisticoDiario primero capturado.
  + **int systemNumber2.** id general del segundo aerogenerador.
  + **double media2.** Media del EstadísticoDiario segundo capturado.
  + **date fecha2.** Fecha del EstadisticoDiario segundo capturado.
  + **double viento2.** Bin de viento del EstadisticoDiario segundo capturado.
* **Fueras**. Corresponde a los eventos complejos detectados por el patrón “Fuera”. Sus campos son:
  + **int id.** Id propio de la base de datos.
  + **int systemNumber**. Id general del aerogenerador.
  + **int vecesFuera.** Número diez minutales de un aerogenerador capturados fuera del intervalo de confianza creado diariamente para un bin concreto de viento. Es decir, el número de outliers detectados.
  + **double viento.** Bin de viento.
  + **double media.** Media de la potencia producida entre el número de eventos recogidos en total en ese bin de viento de un aerogenerador concreto (KW/h).
  + **double desviacion.** Desviación típica de la potencia producida en ese bin de viento de un aerogenerador en concreto (KW/h).
  + **date fecha**. Día del año en que se capturó el evento complejo Fuera que este registro representa.

A continuación, vamos a describir cuales son los controladores que contienen la mayor parte de la lógica de interacción con el servidor.

* **Controlador RankingProds**. Este contolador gestiona toda la información que se obtiene del análisis de la productividad. Los métodos más importantes son los siguientes:
  + **void introProductividad ().** Despliega la página de introducción al apartado de productividad. Contiene un formulario para ingresar una fecha y el nombre del parque a analizar.
  + **void getProductividad (String parque1, String datePickerI).** A través del formulario de la introducción, iniciaremos el panel de análisis de productividad, según el parque que deseemos. Se introduce el día para tener unos datos ya desplegados, sin embargo, los días en el análisis se pueden cambiar a lo largo de la interacción con el sistema.
  + **void muestroRankingParque (String fecha).** Devuelve una vista de la tabla del ranking introducido por parámetro.
  + **void muestroGrafica (String diasG, int[] aerosG).** Muestra el seguimiento del ranking en un rango de días de los aerogeneradores seleccionados utilizando una gráfica de áreas.
  + **void getInfoAero (int idAero).** Muestra por pantalla el id de la base de datos, el systemNumber, la fila, la columna y el idIngeboards, del aerogenerador introducido por parámetro.
  + **Array getTransiciones (int parque).** Devuelve un array con las transiciones del ranking efectuadas por los aerogeneradores pertenecientes al parque introducido por parámetro.
  + **void getRankingDia (String fecha, int parque).** Devuelve un array con los registros ranking de un día concreto, pertenecientes a los aerogeneradores del parque introducido por parámetro.
* **Controlador EstadisticoDiarios**. Este contolador gestiona toda la información que se obtiene del análisis del rendimiento dentro de Eolic Event Consumer. Los métodos más importantes son los siguientes:
  + **void introRendimiento ()**. Despliega la página de introducción al apartado de rendimiento. Contiene un formulario para ingresar una fecha y el nombre del parque a analizar.
  + **void getRendimiento (String parque1, String fecha)**. A través del formulario de la introducción, iniciaremos el panel de análisis de rendimiento, según el parque que deseemos. Se introduce el día para tener unos datos ya desplegados, sin embargo, los días en el análisis se pueden cambiar a lo largo de la interacción con el sistema.
  + **void muestroGraficaRangos (int [] aerosG, String fecha, String contenedor)**. Genera las curvas de potencia de los aerogeneradores seleccionados, producidas en el día que ha sido introducido por parámetro. Las curvas serán más anchas o menos, según la desviación obtenida en cada bin de viento. Se genera el gráfico en un contenedor html con el id igual al contenido de la variable “contenedor” pasada por parámetro.
  + **void muestroGraficaBarras (int [] aerosG, int String dia, String contenedor)**. Genera un gráfico de barras con las desviaciones de los aerogeneradores producidas en el día que ha sido introducido por parámetro. Se genera el gráfico en un contenedor html con el id igual al contenido de la variable “contenedor” pasada por parámetro.
  + **void muestroGraficaQuesos (String dia, String contenedor, String parque)**. Genera un gráfico de sectores por cada bin de viento, marcando cuantos outliers del intervalo de confianza generado por cada generador, se han dado en el día que ha sido introducido por parámetro. Cada sector representa el número de outliers de un aerogenerador. Se generan los gráficos en un contenedor html, con el id igual al contenido de la variable “contenedor” pasada por parámetro.
  + **void muestroGraficaBinMedias (String dia, int[] aerosTemporal, double binViento, String contenedor)**. Se muestra un gráfico con el seguimiento de las medias diarias de la potencia producida de los aerogeneradores seleccionados. Se mostrarán aquellas registradas en los días introducidos por parámetro. Se genera el gráfico en un contenedor html con el id igual al contenido de la variable “contenedor” pasada por parámetro.
  + **void muestroGraficaBinDesviaciones (String dia, int[] aerosTemporal, double binViento, String contenedor)**. Se muestra un gráfico con el seguimiento de las desviaciones diarias de la potencia producida de los aerogeneradores seleccionados. Se mostrarán aquellas registradas en los días introducidos por parámetro. Se genera el gráfico en un contenedor html con el id igual al contenido de la variable “contenedor” pasada por parámetro.
  + **void cargaDatosEstadisticos (int parque, String fecha, String contenedor)**. Despliega una vista con un date table, con los estadísticos que se han dado en un día concreto en el parque introducido por parámetro.
  + **void cargaDatosFueras (int parque, String fecha, String contenedor)**. Despliega una vista con un date table, con los datos de los outliers que se han dado en un día concreto en el parque introducido por parámetro.
  + **void cargaDatosBajadas (int parque, String fecha, String contenedor)**. Despliega una vista con un date table, con los datos de las bajadas de rendimiento que se han dado en un día concreto en el parque introducido por parámetro.
  + **array getEstadisticoDia (String fecha, int parque)**. Devuelve un array con los estadísticos de un día concreto, pertenecientes a los aerogeneradores del parque introducido por parámetro.
  + **array getFuerasDia (String fecha, int parque)**. Devuelve un array con los outliers que se han dado en un día concreto, pertenecientes a los aerogeneradores del parque introducido por parámetro.
  + **array getBajadasDia (String fecha, int parque)**. Devuelve un array con las bajadas de rendimiento que se han dado en un día concreto, pertenecientes a los aerogeneradores del parque introducido por parámetro.
    - 1. Sprint Burndown

Tal y como podemos ver en la Ilustración 61, la estimación de horas de este sprint coincide con la real, 138 horas en total.

Ha habido muchas dificultades en el aprendizaje CakePHP, ya que el sistema de comunicación con el servidor ha resultado tedioso de aprender. No obstante, conforme se fueron desarrollando las tareas se fue almoldando el trabajo para al menos tener en el prototipo, el apartado de la productividad terminado y el de rendimiento planteado. De ahí ese estancamiento en las horas de las semanas más próximas al final del sprint.

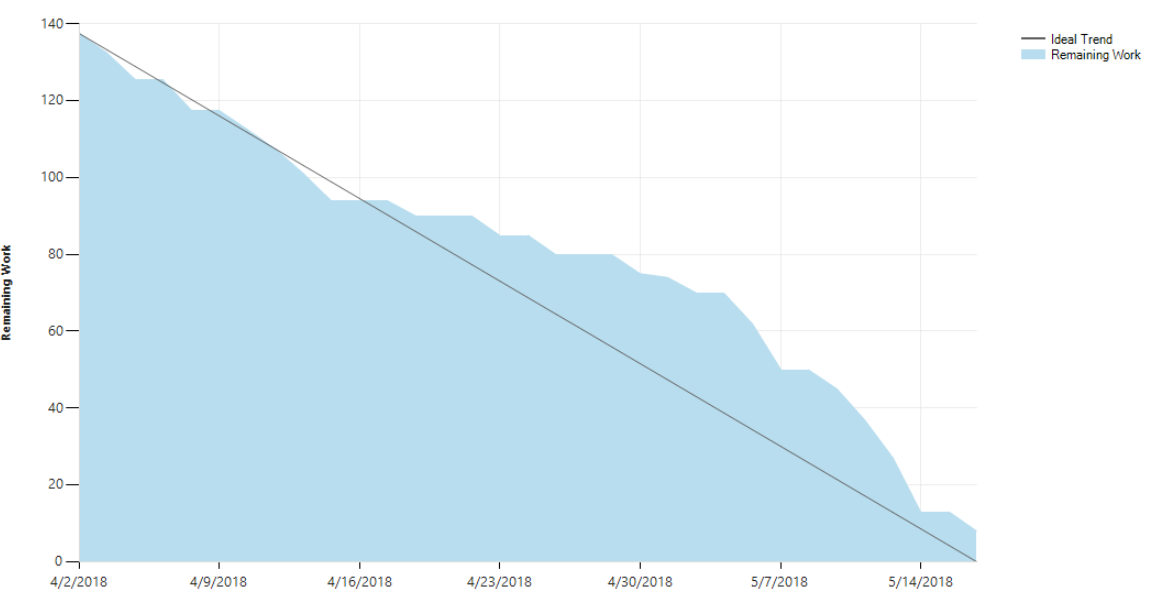


Ilustración 62. Sprint Burndown Desarrollo de la plataforma.

* + 1. Reuniones al finalizar el sprint

Esta reunión sirvió para dar fin a la segunda iteración y así, revisar los prototipos y la documentación correspondiente. Además, se llevó a cabo la retrospectiva para ver qué hemos hecho bien y qué se puede mejorar de cara al proyecto.

* Dirección: Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A.
* Hora y fecha de la reunión. A las 16:30 el día 16 de mayo de 2018.

Los participantes en esta reunión fueron:

* Antonio Fernández Diez. Ingeniero Industrial. Especializado en placas fotovoltaicas.
* Gregorio Diaz Descalzo. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález Segovia. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que tratar

Los puntos de la reunión son:

* Revisión del ESB.
* Revisión de los patrones.
* Revisión de la aplicación gráfica.
* Revisión de la documentación.
  + - 1. Información recogida

Esta reunión sirvió para dar fin al tercer sprint, y determinar cuales eran los hitos y artefactos que faltaban por incluir para dar por teminado este trabajo de fin de grado.

Las funcionalidades implementadas en el Enterprise Service Bus son adecuadas y por el momento no debemos de rectificar ni añadir nada mas. La única defiencia que tenía por entonces la implementación de los fujos del ESB, es que no había ningún comentario en el código. Se pidió expresamente añadir comentarios en todo el código para facilitar su comprensión.

Tras las correcciones dadas en el sprint anterior para hacer más dinámico y mejorar la eficiencia los patrones enunciados, se ha implementado una versión que cumple con los objetivos propuestos y las correcciones.

La funcionalidad de la aplicación era insuficiente ya que únicamente estaba terminado el apartado de productividad, y el de rendimiento no estaba completo debido a las dificultades que se han dado en el aprendizaje al empezar su implementación. El Product Owner pese a que no se cumplieron todos los objetivos, aceptó lo desarrollado, y se hicieron las siguientes apreciaciones.

* **Productividad.** El apartado que realiza el análisis de la productividad cumplía con las correcciones dadas cuando se presentaron las interfaces candidatas, por lo que se consideró como completado y apto.
* **Rendimiento.** El apartado que realiza el análisis del rendimiento no estaba completo. Estaba implementado parcialmente el estudio de desviaciones y faltaba por implementar el análisis temporal y el de los outliers. Se deberá de terminar este apartado, y estará pendiente de aprobación en el siguiente sprint.
* **Guiar al usuario**. Añadir un control de excepciones donde se informe cuando no hay resultados en la búsqueda de información que ha sido introducida, e impedir que el usuario realice una búsqueda con algún campo en los formularios vacío.
* **Estandarizar la interfaz**. Los apartados presentados, así como la pantalla de inicio de Eolic Event Consumer no están estandarizados. Deben de estar regidos por el mismo estilo y debe de cuidarse la coherencia entre las diferentes interfaces.

Finalmente, la documentación era correcta pero debido al gran avance realizado en este sprint se pidió que se cuidara la coherencia entre las diferentes partes del capítulo de desarrollo. Se pidió lo siguiente para el depósito del proyecto:

* Releer, y reescribir otra vez la documentación si fuera necesario.
* Realizar el capítulo 6 donde quedarán especificadas las pruebas tanto del ESB, como de la aplicación gráfica.
* Incluir nuevas líneas de trabajo futuro.
* Escribir las conclusiones.
* Redifinir el estilo de los índices del documento.
  + - 1. Próximos pasos

Se han acordado la incorporación de las siguientes historias de usuario para el siguiente sprint:

* **Refinir diagramas.** Corregir el diseño realizado en este sprint. Las correcciones deberán de efectuarse en los diagramas y añadirse corregidos a la documentación realizada en el sprint 3.
* **Documentación.** Terminar toda la documentación teniendo en cuenta las apreaciones realizadas en esta reunión.
* **Pruebas**. Llevar a cabo la planificación de los diferentes casos de prueba y recoger las evidencias que corroboren su éxito.
* **Implementación.** Terminar el apartado del rendimiento y unificar la interfaz tal y como se ha compentado en las correcciones de esta reunión.
  + 1. Retrospectiva del Sprint

**¿Qué salió bien en la iteración?**

* Se ha fomentado el buen ambiente de trabajo entre los Stakeholders, gracias al enunciado de los patrones a través de la técnica de Storyboards.
* El apartado de la productividad implementado en la interfaz gráfica ha sido aprobado por el Product Owner, por lo que, si enfocamos de la misma forma el de rendimiento, la aplicación será un éxito.
* Los patrones se dan por terminados ya que cumplen con los objetivos.
* La documentación en líneas generales es correcta.
* Se ha logrado adquirir todas las nociones necesarias de CakePHP para obtener un resultado aceptable.

**¿Qué no salió bien en la iteración?**

* No estimamos bien las horas al comienzo de este sprint, y por culpa de ello vamos a tardar más de lo esperado en implementar la aplicación gráfica.
* Debemos de releer la documentación para mejorar la compresión de su contenido.
* Hubo demasiada frustración al comienzo de este sprint al aprender las primeras nociones de CakePHP. Hemos de aprender a superarnos, y afrontar con calma los problemas de aprendizaje.

**¿Qué mejoras vamos a implementar en la próxima iteración?**

* Mejorar la actitud a la hora de aprender más nociones de CakePHP, así como de aprender nuevas tecnologías.
* Aprovechar el buen ambiente de trabajo en este sprint, para reflejarlo de cara al proyecto.
* Tener más paciencia a la hora de leer la documentación para mejorar la comprensión del documento.
  1. Sprint 4. ultimando detalles y pruebas

En este sprint terminaremos de implementar el apartado de rendimiento de la aplicación gráfica, añadiéndole el control de excepciones nombrado por el Product Owner en la revisión del sprint anterior, planificaremos las pruebas que llevaremos a cabo, y dejaremos la documentación lista para hacer el depósito el día 18 de junio. Podemos ver la duración de esta iteración y sus participantes en la Tabla 22.

Tabla 22. Información base del sprint 4.

|  |  |
| --- | --- |
| **Fecha de inicio y fin** | 17 de mayo de 2018 – 17 de junio de 2018 (Depósito el día 18) |
| **Personas involucradas** | * Francisco José Polo Sánchez. * Ignacio García de Carellán. * Gregorio Diáz Descalzo. * Antonio Fernández Diez. * Enrique Brazález Segovia. * Fernando Luján Martínez |

* + 1. Reunión de planificación del sprint

Esta reunión se realizó justo después de la de finalización del sprint anterior. La información de la reunión es la siguiente:

* Dirección: Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A.
* Hora y fecha de la reunión. A las 17:30 el día 16 de mayo de 2018.

Los asistentes de la reunión son:

* Francisco José Polo. Enca Parque Científico y Tecnológico, Paseo de la Innovación, 3, 02006 Albacete. Oficinas de Ingeteam S.A. encargado del departamento de I+D+i en el año 2018.
* Antonio Fernández Diez. Ingeniero Industrial. Especializado en placas fotovoltaicas.
* Gregorio Diaz Descalzo. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález Segovia. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que trataR

Debido a que se hizo seguida a la revisión del sprint los puntos de esta reunión son:

* Realización de pruebas.
* Depósito del trabajo.
* Ley de Protección de Datos.
* Una única reunión final con el Product Owner.
  + - 1. Información recogida

Al comienzo de la reunión se estipuló que las pruebas no debían de ser automatizadas ya que se perdería mucho tiempo en ello. Sin embargo, estaría bien que al menos estén planteados los casos de prueba más representativos con capturas de pantalla que recojan las evidencias de que el software funciona correctamente. Deberemos de comprobar sobre todo que la salida por consola del ESB, es sólida, para proporcionar trazabilidad en los datos que se van introduciendo en el motor de análisis. Según lo acordado en esta reunión, el enunciado de los casos de prueba deberá de recoger: los datos de entrada, la salida esperada, la salida obtenida y la referencia a las evidencias capturadas.

El depósito del trabajo de fin de grado es el día 18 de junio, se hizo hincapié en tener toda la documentación, y el código completamente el día 17, para el día 18 repasar la memoria y entregarlo.

Debido a la Ley de Protección de Datos deberemos de borrar de todo el trabajo de fin de grado todos los nombres y referencias que pudieran identificar el parque real que estamos analizando, ya que los datos con los que hemos estado entrenando la aplicación son reales.

Debido a la gran carga de trabajo del Product Owner, únicamente haremos una reunión final el día 7 de junio, para dar por finalizado el desarrollo de la aplicación gráfica. El resto de las tareas son de carácter académico por lo que no necesitaremos la aprobación del Product Owner.

* + 1. Planificación del sprint

La organización de este sprint queda recogida en la Tabla 22, enunciando cada una de las historias de usuario, así como sus tareas y estimación de horas de trabajo.

Tabla 23. Resumen de desarrollo sprint 4.

|  |  |  |
| --- | --- | --- |
| Historia de usuario | Tarea | Horas |
| Redifinir diagramas | Redifinir diagrama de paquetes. | 1 |
| Redifinir diagrama de clases de app gráfica. | 1 |
| Refinir diagrama de secuencia del input. | 1 |
| Redifinir diagrama de secuencia de adición de patrones. | 1 |
| Redifinir diagrama de secuencia de despliegue de eventos complejos. | 1 |
| Implementación de la página | Implementación de las gráficas de curvas diarias del rendimiento de manera final. | 4 |
| Implementación del diagrama de barras del rendimiento. | 2 |
| Implementación del apartado de outliers del rendimiento. | 5 |
| Acoplación de las tres gráficas y unificarlas. | 1 |
| Pantalla de inicio. | 3 |
| Modificación en la creación de gráficas para poder ingresarlas en un contenedor concreto. | 2 |
| Refinado de todas las vistas y unificación. | 2 |
| Control de excepciones primero y control de las fechas. | 2 |
| Ingreso de aeros correspondientes al parque y bines de viento correctos. | 4 |
| Pestaña para el seguimiento temporal | 4 |
| Segundo control de excepciones más refinado | 5 |
| Corrección en la dimensión de los contenedores | 2,5 |
| Pruebas | Probar mecanismo de notificaciones | 3 |
| Plantear modelo y aceptación del SM. | 1 |
| Planificar alternativas de cómo plantear los casos de prueba. | 2 |
| Planificar pruebas de la aplicación gráfica. | 3.5 |
| Planificar pruebas para ESB. | 3.5 |
| Documentar pruebas para el ESB. | 5 |
| Documentar pruebas para la aplicación gráfica. | 5 |
| Corregir y unificar pruebas. | 5 |
| Documentación | Repasar desarrollo de sprint 1 | 2 |
| Repasar desarrollo de sprint 2 | 2 |
| Repasar desarrollo de sprint 3 | 2 |
| Repasar desarrollo de sprint 4 | 5 |
| Realizar capitulo de pruebas | 5 |
| Realizar capítulo de resultados. | 5 |
| Realizar capítulo de conclusiones. | 3 |

* + 1. Reuniones de Seguimiento

En la Tabla 23 quedan reflejadas las reuniones semanales que tuvieron lugar a lo largo de este sprint.

Tabla 24. Reuniones semanales del sprint 4.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre de la reunión** | **Fecha** | **Participantes** | **Asuntos que tratar** |
| Revisión del prototipo | 6/6/2018 | * Francisco José Polo Sánchez. * Ignacio García de Carellán. * Gregorio Diáz Descalzo. * Antonio Fernández Diez. * Enrique Brazález Segovia. * Fernando Luján Martínez. | * Revisión de la depuración del apartado de productividad. * Revisión de rendimiento. * Revisión de excepciones. |
| Revisión de diseño y planificación de pruebas | 8/6/2018 | * Ignacio García de Carellán. * Gregorio Diáz Descalzo. * Antonio Fernández Diez. * Enrique Brazález Segovia. | * Revisión del enunciado de las pruebas. * Revisión del formato de redacción. * Estado de las pruebas. * ¿Cuánto queda? * Revisión de estructura de la memoria. |

* + 1. Desarrollo del sprint

La redifinición de diagramas de secuencia se ha añadido a la documentación del sprint anterior para respetar la coherencia del documento y no repetir información irrelevante.

En cuanto al apartado de rendimiento, ni la arquitectura ni el diseño ha cambiado puesto que se dejó planteado, pero no implementado. En este sprint, nos hemos dedicado principalmente a terminar de implementarlo.

Cabe destacar que en la última reunión con el Product Owner se hicieron muchas correcciones en la aplicación gráfica, a continuación, los veremos en detalle.

* + 1. 1. Correcciones de Última hora

En la última reunión efectuada con el Product Owner el día 6 de junio en sus oficinas, se hicieron una serie de correcciones que supusieron una carga extra de trabajo que no se preveía en este sprint. Las correcciones que se hicieron fueron las siguientes:

* **Incorporación de notificaciones.** Debido a que ya tenemos los patrones totalmente desarrollados, se solicita la implementación de más patrones que identifiquen los fenómenos más peculiares vistos en el análisis del rendimiento y de la productividad.
* **Lugar de acceso a las notificaciones.** Crear el acceso de las notificaciones en el menú superior del portal, indicando cuantas situaciones de interés se han detectado.
* **Variaciones en tanto por ciento.** Estas notificaciones deberán de estudiar variaciones en tanto por ciento de la productividad y el rendimiento, es decir, estudiar las bajadas de la potencia producida, así como el número de outliers detectados y las transiciones en el ranking.
* **Terminología.** Corrección de la terminología empleada en las vistas. Por ejemplo, sustituir “desviaciones” por “desviaciones típicas”, “puntos fuera” por “outliers”, etc. La utilización de términos erróneos puede dar lugar a confusión durante la utilización de la herramienta.
* **Gráfica de transiciones en el ranking.** Añadir una gráfica en el apartado de productividad que represente la variación de posiciones en el ranking diario. La gráfica ya implementada, representa numéricamente la potencia producida promedio KW/h en cada día, pero no representa con exactitud las subidas y bajadas dentro del ranking. Esta nueva gráfica sí que representaría estas transiciones.

Esta carga extra de trabajo derivó en la inclusión de una historia de usuario de 39 horas aproximadamente. La división de tareas y horas es la aportada en la Tabla 25.

Tabla 25. Historia de usuario “Correcciones del Product Owner”.

|  |  |  |
| --- | --- | --- |
| Historia de usuario | Tarea | Horas |
| Correcciones del Product Owner | Rectificación del filtrado de datos hecho en el ESB. | 2 |
| Control de timestamps. | 1 |
| Redifinición de campo fecha de WindEvent. | 1 |
| Inclusión del gráfico de seguimiento de posiciones en el ranking. | 5 |
| Refactorizar todo el proyecto y renombrar el término “puntos fuera” por “outliers”. | 2 |
| Cambio de nombres en los ejes de los gráficos por KW/h, y no KW. | 2 |
| Creación de notificaciones de distinto tipo a través de patrones. | 5 |
| Crear tabla de notificaciones. | 2 |
| Amoldar ESB al patrón de las notificaciones. | 1 |
| Crear apartado de notificaciones en el nav-top de la aplicación | 5 |
| Comentar el código y aplicar primer principio de Lean-Aux (Eliminar lo que sobre). | 5 |
| Gestión de notificaciones con estados “Pendiente” y “Tramitada”. | 4 |
| Creación de pestaña vinculada a la información de las notificaciones y depurar | 4 |

A continuación, en el punto 5.5.4.2 , veremos como se ha organizado el desarrollo de estas correcciones.

* + - 1. Gestión de notificaciones

Esta modificación en el proyecto supuso un reto, ya que a una semana de entregar se incorporaron nuevos patrones, que implicaron cambios en el flujo de despliegue y la implementación de nuevos de elementos dentro de la aplicación gráfica.

Se incorporaron 4 tipo de notificaciones, una por cada patrón implementado. Se tuvo que modificar una vez más, dos de los patrones ya implementados. El patrón “Escala” y el patrón “Transicion”. Los patrones modificados los podemos ver en la Ilustración 63 y la Ilustración 64. Hemos incorporado los siguientes campos:

**Patrón Escala.** Se incorporó el campo **numAeros**, para llevar la cuenta de cuantos aerogeneradores contaba el ranking capturado, y, por otro lado, **productividad,** para saber numéricamente cuan productivo había sido ese aerogenerador.

**@Name("Escala")**

INSERT INTO **Escala**

SELECT systemNumber as systemNumber,

count(suma) % numAeros+1 as posicion,

fecha,

numAeros,

productividad,

FROM RankingProd

Ilustración 63. Patrón Escala V 3.0.

**Patrón Transición.** Se incorporó el campo **variacionR** para traducir en tanto por ciento cuanto había bajado el aerogenerador de un día para otro. Por ejemplo, si en un parque hay 30 aerogeneradores y este campo tiene un valor -3.33, significaría que solo ha bajado un puesto. Además, se añadió el campo **variacionPR**, que refleja también la variación de “productividad” en tanto por cierto con respecto el día anterior. Por ejemplo, si este campo tiene un valor 15, significa que ha crecido un +15% con respecto el día anterior.

@Name("**Transicion**")

INSERT INTO **Transicion**

SELECT a1.systemNumber as systemNumber,

a1.posicion as posicionInicio,

a2.posicion as posicionFin,

a2.posicion - a1.posicion as variacion,

a1.fecha as inicio,

a2.fecha as fin,

(a2.posicion - a1.posicion)\*(100/a1.numAeros) as variacionR,

(a2.productividad\*100/a1.productividad) - 100 as variacionPR

FROM PATTERN [every a1 = Escala() ->

a2 = Escala(a1.systemNumber = a2.systemNumber)

]

Ilustración 64. Patrón Transición V 3.0

Los patrones, aunque son distintos, todos despliegan el mismo evento complejo “Notificacion”, pero la disposición de sus campos varía según son de un tipo o de otro. Se ha hecho de esta forma, para ganar eficiencia y tener unificadas todas las notificaciones en una misma tabla en la base de datos posteriormente. Los patrones para cada tipo de notificación son los siguientes:

* **Notificación “Fuera”.** Se crea una notificación de tipo “Fuera”, cada vez que se despliega un evento complejo “Fuera” con más de 14 outliers detectados. Cada notificación de este tipo contendrá el número de máquina, la fecha, el número de outliers detectados, el bin de viento, la media y la desviación típica de los kilowatios hora producidos en esa fecha.

@Name("**NotificacionFuera**")

INSERT INTO **Notificacion**

SELECT e.systemNumber as systemNumber,

"**Fuera**" as tipo,

e.fecha as fecha,

e.vecesFuera as campo1,

e.vecesFuera as campo2,

e.viento as campo3,

e.media as campo4,

e.desviacion as campo5

FROM PATTERN[every e = Fuera(e.vecesFuera>14)]

Ilustración 65. Patrón de la notificación "Fuera".

* **Notificación “Transicion”.** Se crea una notificación de tipo “Transicion”, cada vez que se detecta, que un aerogenerador ha bajado un 75 % en el ranking, es decir, que de estar en cabeza de tabla se ha puesto al final de un día para otro. Cada notificación de este tipo contendrá el número de máquina, la fecha, la posición del día en el que se detectó la transición, la variación de puestos que ha habido, y finalmente los campos “variacionR” y “variacionPR” que hemos incorporado al patrón Transición anteriormente.

@Name("**NotificacionTransicion**")

INSERT INTO **Notificacion**

SELECT e.systemNumber as systemNumber,

"**Transicion**" as tipo,

e.inicio as fecha,

e.posicionFin as campo1,

e.variacion as campo2,

e.variacionR as campo3,

e.variacionPR as campo4,

e.variacionPR as campo5

FROM PATTERN[every e = Transicion(e.variacionR < -75 )]

Ilustración 66. Patrón de la notificación "Transición".

* **Notificación “BajadaM”.** Se crea una notificación de tipo “BajadaM”, cada vez que se detecta, que un aerogenerador ha bajado su rendimiento un 75 % con respecto la lectura del día anterior en un mismo bin de viento. Cada notificación de este tipo contendrá el número de máquina, la fecha, el número de diez minutales capturados de los días que estamos comparando, el bin de viento, la media de los kilowatios hora producidos el día en el que se detectó la notificación y su variación en tanto por ciento de un día para otro.

@Name("**NotificacionBajadaM**")

INSERT INTO **Notificacion**

SELECT a1.systemNumber as systemNumber,

"**BajadaM**" as tipo,

a1.fecha as fecha,

a1.veces as campo1,

a2.veces as campo2,

a2.viento as campo3,

a2.media as campo4,

(100-(a2.media\*100/a1.media)) as campo5

FROM PATTERN[

every a1=EstadisticoDiario() -> a2=EstadisticoDiario(a1.systemNumber=a2.systemNumber AND

a1.viento = a2.viento AND (a2.media\*100/a1.media) <= 25)]

Ilustración 67. Patrón de la notificación "BajadaM".

* **Notificación “BajadaD”.** Se crea una notificación de tipo “BajadaD”, cada vez que se detecta un aumento del 200 % (se triplica la desviación típica) en la dispersión de los diezminutales de un aerogenerador de un día para otro, en un bin de viento concreto. Cada notificación de este tipo contendrá el número de máquina, la fecha, el número de diez minutales capturados de los días que estamos comparando, el bin de viento, la desviación típica de los kilowatios hora producidos el día en el que se detectó la notificación y la variación de la desviación típica en tanto por ciento de un día para otro.

@Name("**NotificacionBajadaD**")

INSERT INTO **Notificacion**

SELECT a1.systemNumber as systemNumber,

"**BajadaD**" as tipo,

a1.fecha as fecha,

a1.veces as campo1,

a2.veces as campo2,

a2.viento as campo3,

a2.desviacion as campo4,

(100-(a2.desviacion\*100/a1.desviacion)) as campo5

FROM PATTERN[ every a1=EstadisticoDiario() ->

a2=EstadisticoDiario(a1.systemNumber=a2.systemNumber AND

a1.viento = a2.viento AND

((a2.desviacion\*100/a1.desviacion) >200) AND

a1.desviacion>=50)

]

Ilustración 68. Patrón de la notificación "BajadaD".

La inclusión de estos patrones ha supuesto la adición de un acceso a la base de datos más en el flujo de despliegue del ESB, como vemos en la Ilustración 69.

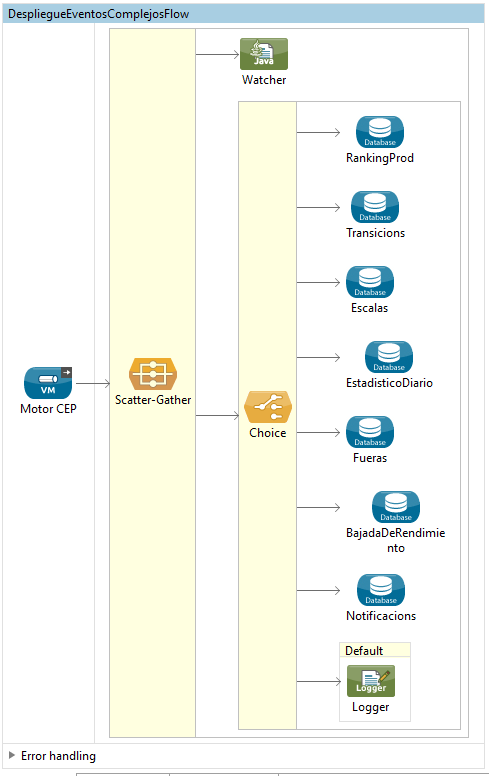


Ilustración 69. Flujo del despliegue de eventos complejos V 3.0

Además, se ha incluido en el modelo, la entidad “Notificacions”. La incorporación de las notificaciones en el modelo de clases de la aplicación gráfica lo podemos ver en el diagrama actualizado de la Ilustración 70.

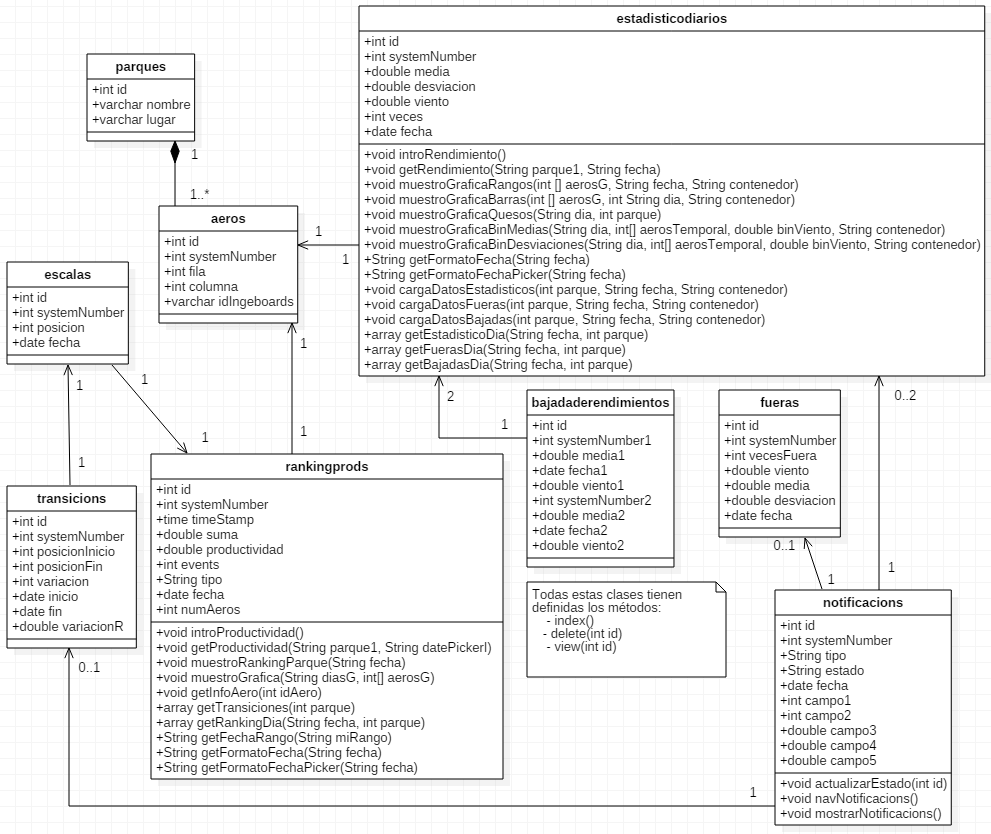


Ilustración 70. Diagrama de clases de la aplicación gráfica V 2.0.

El valor de los atributos de los eventos complejos “Notificacion” tienen un significado u otro, según el patrón que los ha detectado. Podemos ver en la Tabla 26, las relaciones que tienen los campos de los eventos complejos “Notificacion” con respecto su tipo.

Tabla 26. Tipos de notificaciones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tipo | Campo1 | Campo2 | Campo3 | Campo4 | Campo5 |
| Fuera | Nº de Outliers | Nº de Outliers | Bin de viento  (m/s) | Media (KW/h) | Desviación típica (KW/h) |
| Transicion | Pos.Final | Variación (puestos) | VariacionR (%) | VariacionPR % | VariacionPR  % |
| BajadaD | Nº Diez Minutales | Nº Diez Minutales | Bin de viento  (m/s) | Media (KW/h) | Variación de la media % |
| BajadaM | Nº Diez Minutales | Nº Diez Minutales | Bin de viento  (m/s) | Desviación típica (KW/h) | Variación de la desviación típica % |

El campo “estado” en el modelo de la aplicación gráfica puede tener dos valores, “Pendiente” o “Tramitada”. De forma inicial, todas las notificaciones desde que se insertan en la base de datos estarán en estado “Pendiente”. Durante la navegación de la aplicación, el menú superior mostrará sólo las notificaciones que estén en estado “Pendiente”. De esta forma, mostraremos qué alertas se han dado y no han sido tramitadas. Para tramitar y dar por supervisadas las notificaciones, cambiaremos su estado a “Tramitada” a través de la vista que muestra su información con detalle.

El controlador “NotificacionsController” gestiona la actualización de estados, así como su despliegue en el portal web. Sus métodos son los siguientes:

* **void actualizarEstado (int id)**. Es el método encargado de pasar del estado “Pendiente”, a “Tramitada”, de la notifiación pasada por parámetro.
* **void navNotificacions()**. Es el método utilizado para mostrar a través de la nav-bar superior el número de notificaciones pendientes almacenadas.
* **void mostrarNotificacions()**. Se despliega una vista con toda la información en detalle de las notificaciones detectadas. Además, cada notificación tiene la opción de poderse tramitar a través de un botón.
  + - 1. Pruebas

Esta parte del desarrollo de este sprint queda recogido en el Capítulo 6, donde podemos ver la planificación de los casos de prueba, tanto del ESB, como de la aplicación gráfica.

* + - 1. Sprint Burndown

Como podemos ver en la XXXXX, al comienzo de este sprint se estimaron 91 horas y media. Sin embargo, debido a las correcciones aportadas por el Product Owner, se sumaron 39 horas más. Podemos ver este incremento de horas en la primera semana de junio.

AQUÍ IRÍA EL SPRINT BURNDOWN

A pesar de este problema, el sprint terminó con éxito y las horas reales de este sprint fueron 130 y media aproximadamente.

* + 1. Reuniones al finalizar el sprint

Esta reunión sirvió para poner fin a este proyecto. Se revisaron todos los artefactos generados, y el Scrum Master dio una serie de directrices para llevar a cabo la defensa de este TFG.

* Dirección. Edificio Infante Don Juan Manuel. Avda. de España s/n. Albacete. Despacho 0. B.8.
* Hora y fecha de la reunión. A las 16:30 el día 15 de mayo de 2018.

Los participantes en esta reunión fueron:

* Gregorio Diaz Descalzo. Jefe de proyecto de este trabajo de fin de grado. Titular de la Universidad de Castilla-La Mancha.
* Enrique Brazález Segovia. Desarrollador y encargado de defender este trabajo de fin de grado.
  + - 1. Asuntos que tratar

Los puntos de la reunión son:

* Revisión de la implementación de las correcciones hechas por el Product Owner.
* Revisión del prototipo.
* Revisión de la documentación.
* Defensa del proyecto de fin de grado.
  + - 1. Información recogida

En primer lugar, se revisó la incorporación de todas las correcciones propiciadas por el Product Owner. La solución aportada era apta ya que mostraba los fenómenos más extraños dentro de las mediciones, de forma vistosa y sencilla.

El prototipo final cumplía con todas las expectativas puestas en la planificación del proyecto. El único problema era que el sistema no daba ninguna predicción acerca de qué aerogenerador podía fallar y cuál era su fallo. Se puede ver qué aerogenerador tiene un comportamiento anómalo con esta aplicación, sin embargo, no se pueden conocer las causas de forma específica, ya que deberíamos de afinar más los patrones y tener en cuenta aún más variables (temperaturas, presión atmosférica, dirección del viento, etc). Como línea de trabajo futuro podríamos cruzar el log de errores y los diez minutales de las máquinas, para enunciar una predicción. Más adelante en el Capítulo 7 hablaremos de ello.

La estructura de la memoria era correcta, sin embargo, habría que corregir algunas expresiones que dificultaban la lectura del documento. Al final de la reunión, se dieron una serie de pautas a seguir para que la defensa ante el tribunal fuera un éxito. Se pidió expresamente no describir lo que habíamos hecho sprint por sprint, deberíamos de centrarnos en el concepto de la tecnología, explicar la metodología de trabajo utilizada y enunciar de forma clara que fenómenos es capaz de capturar Eolic Event Consumer. Como parte final de la defensa, se sugiere enseñar el portal web al tribunal, sin olvidarnos de mencionar que la información mostrada es fruto del análisis realizado por la tecnología CEP en el ESB.

* + - 1. Próximos pasos

Las únicas tareas que quedan después de haber llevado a cabo esta reunión son las siguientes:

* Releer y corregir expresiones de la memoria.
* Hacer presentación para la defensa de este TFG.
  + 1. Retrospectiva del Sprint

Una vez hecha la reunión de revisión del Sprint, el Product Owner abandonó la sala, para que el Team, junto con el Scrum Master, tuvieran la retrospectiva. El objetivo de esta reunión fue responder a las siguientes preguntas:

**¿Qué salió bien en la iteración?**

* Se ha implementado toda la aplicación que nos habíamos propuesto al comienzo del proyecto.
* Se ha documentado todo el proyecto de forma correcta con una calidad aceptable.
* El Team ha adquirido conocimientos de estadística, tecnología web y computación para futuros trabajos.
* Se ha obtenido una buena relación con el Product Owner de cara al futuro.
* Pese a la carga de trabajo final, se ha conseguido llegar a una solución que cumple con las expectativas propuestas.
* La coordinación y la participación en este sprint ha sido muy buena, ya que los stakeholders estaban interesados en aportar ideas y perfeccionar el resultado ofrecido.

**¿Qué no salió bien en la iteración?**

* Se debieron de haber planteado las correciones de última hora en el sprint 3 y no en el 4, ya que podrían haber ocasionado un gran problema y haber supuesto un retraso en la entrega.
* El Team debería de haber planteado mejor las horas de trabajo en TFS, ya que se ha trabajado más horas de las indicadas en los Sprint Burndown.

**¿Qué podemos mejorar de cara al futuro?**

* Mejorar la comunicación oral de cara a futuras reuniones con otros Product Owner y en concreto, con el tribunal en la defensa del trabajo de fin de grado.
* Mejorar las expresiones utilizadas en la documentación.
* De cara a futuros proyectos, deberemos de tener más al día el control de versiones, en concreto la carga del Sprint Burndown, puesto que es el reflejo de nuestro trabajo diario.
  1. Resumen del desarrollo

En la Tabla 24 observamos el resumen de las horas estimadas a lo largo del proyecto por cada historia de usuario. Hemos de tener en cuenta que el esfuerzo realizado ha sido mayor, ya que se tratan de estimaciones personales del Team.

Tabla 27. Resumen de horas estimadas.

|  |  |  |
| --- | --- | --- |
| Historia de usuario | Nº de Sprint | Horas |
| Introducción a Tecnología CEP | 0 | 22 |
| Primer estudio de la curva de potencia | 1 | 4 |
| Realizar el anteproyecto | 1 | 20 |
| Recogida de requisitos y análisis de contexto | 1 | 20 |
| Acoplamiento e implementación del ESB | 1 | 25 |
| Planificación del trabajo de fin de grado | 1 | 7 |
| Curso de Energía Eólica | 1 | 7 |
| Creación y acoplamiento del simulador con la Nube | 1 | 29 |
| Aprender PHP y familiarización con tecnología web | 2 | 30 |
| Estudio de factores externos distintos de potencia y velocidad de viento | 2 | 11 |
| Estudio de curva de potencia | 2 | 20 |
| Implementar patrones | 2 | 58 |
| Documentación | 2 | 30 |
| Aprendizaje CakePHP | 3 | 26 |
| Diseño y arquitectura | 3 | 20 |
| Implementación | 3 | 54 |
| Documentación | 3 | 21 |
| Retoques de patrones | 3 | 17 |
| Redefinir diagramas | 4 | 5 |
| Pruebas | 4 | 30 |
| Documentación | 4 | 35 |
| Implementación | 4 | 30.5 |
|  | **TOTAL** | **512,5 horas** |

A lo largo del proyecto tal y como podemos ver en los Sprint Burndown las horas estimadas en las reuniones de planificación no han sido suficientes, ya que a lo largo del proyecto ha habido que dedicar más tiempo a algunas tareas. En la Tabla 25, queda definido el resumen de horas estimadas y reales llevadas a cabo en este trabajo de fin grado según las tareas introducidas en el control de trabajo de TFS.

Tabla 28. Resumen de horas estimadas y reales.

|  |  |  |  |
| --- | --- | --- | --- |
| Sprints | | Estimadas | Reales |
| Sprint 0. Introducción a CEP. | | 22 horas | 149 horas |
| Sprint 1. Inicio del proyecto. | | 112 horas | 148 horas |
| Sprint 2. Adaptación al caso real. | | 149 horas | 160 horas |
| Sprint 3. Desarrollo de la plataforma. | | 138 horas | 138 horas |
| Sprint 4. Ultimando detalles y pruebas. | | 91,5 horas | 130,5 horas |
|  | **TOTAL** | **512,5 horas** | **598,5 horas** |

1. Experimentos y resultados

En este apartado metemos todas las pruebas funcionales unitarios y de integración llevadas a cabo en este proyecto. Así como pruebas de calidad.

Planificación de las pruebas.

1. Conclusiones y propuestas
   1. Conclusiones

Dos páginas ricas de conclusiones

* Acabado este proyecto hemos de destacar el objetivo
* Repetir el objetivo.
* Introducción de lo que abarca CEP OTRA VEZ.
* QUÉ HACE CEP en palabras breves.
* QUÉ NO HACE.
* Deficiencias que presenta, ventajas absolutas fruto de su trabajo
* Qué me ha aportado realizar el proyecto como ingeniero informático.
* Qué me ha aportado trabajar con Ingeteam como cliente.
  1. Trabajo futuro

Conforme se ha ido avanzando en el desarrollo de esta aplicación, se han descubierto multitud de mejoras y modificaciones que pueden servir para hacer más versátil y eficiente la aplicación. Como consecuencia de estas deficiencias, se describen a continuación varias ampliaciones y líneas de trabajo futuras para mejorar el sistema:

* **Ingreso de patrones desde la interfaz.** Una de las deficiencias que tiene nuestro diseño es la integración de nuevos patrones en el motor CEP. Por ello uno de los siguientes pasos podría ser que desde la aplicación web se nos permitiera introducir nuevos patrones y se incorporasen directamente en nuestra aplicación CEP. Dotando de comunicación directa entre el ESB y el servidor donde esté ubicado el servicio web. Como hemos visto, el ESB está dotado de conectores que permiten la comunicación de muchas formas con componentes externos, simplemente sería rediseñar otro flujo de interacción en el ESB que permita la entrada de estos patrones a través de AJAX, o bien de una petición POST.
* **Dar de baja los patrones desde la interfaz.** Uno de los problemas que tiene la aplicación es que no podemos dar de baja un patrón una vez lo metemos en nuestro motor CEP. Tal y como lo tenemos planteado hasta el momento, deberemos de resetear el ESB para quitar los patrones. La idea es similar a la del punto anterior, esos patrones una vez dados de alta en el sistema podrían ser incorporados en una base de datos. Y a través de peticiones al servidor ser incorporados en el motor CEP o no. Recordemos que el ESB sólo está explotando su potencial en un porcentaje muy bajo, podemos sacarle muchísimo más partido.
* **Depurar la aplicación web.** El diseño de la aplicación web tiene deficiencias a nivel de seguridad y no explota del todo CakePHP. Una de las líneas de trabajo futuras, sería la depuración del código de la página para que siguiera más fielmente el patrón MVC y conseguir un trabajo más responsive. Hasta el momento la aplicación gráfica abarca las funcionalidades más básicas para que la visualización de alarmas sea efectiva, no obstante, es mejorable.
* **Migración de MySQL a MongoDB.** Uno de los problemas que tiene nuestra aplicación es que el transformador de los datos de entrada es específico para un modelo de aerogenerador concreto, al igual que las tablas MySQL creadas. La migración a MongoDB la hemos visto necesaria conforme hemos ido avanzando a lo largo del proyecto. Sin embargo, el cliente todavía no ha dado el paso para migrar su sistema actual a MongoDB, por ello hemos intentado hacer un sistema totalmente compatible con lo que tienen implantado hasta el momento. No obstante, este cambio es necesario en el futuro, ya que se ganará dinamicidad, versatilidad y se explotará al 100 % el Procesamiento de Eventos Complejos.
* **Conexión real con los PLCs de los aerogeneradores.** Esta aplicación simula el flujo de datos real, haciendo uso de delays, y timestamps ficticios. La idea de CEP, es conectar a tiempo real su motor de procesamiento con los productores de eventos. Sin embargo, esto no es posible debido a la arquitectura de los parques eólicos, o con el que hemos estado trabajando actualmente. Por ello, una línea de trabajo futura puede ser el estudio de implantar una vía de comunicación entre los PLCs de monitorización de los aerogeneradores, con las subestaciones de los parques, para que estas deriven la información sacada a tiempo a real al motor CEP desplegando todo el potencial de Eolic Event Consumer.
* **Exprimir Esper-EPL.**  Los patrones implementados en Esper-EPL solo exprimen el jugo de la sintáxis en un 30 %. Por ello uno de los trabajos futuros, sería trabajar en implementar patrones mucho más potentes haciendo uso de más funcionalidades que el lenguaje ofrece. La documentación y la librería de Esper ofrece multitud de operadores y funciones muy útiles, y que están sin explotar del todo en este trabajo de fin de grado.
* **Depurar los patrones y optimizarlos contando con más factores.** Los patrones que hemos implementado para ver el rendimiento y productividad de los aerogeneradores son básicos, ya que solo hacemos uso de unas pocas variables que el sistema de monitorización provee. Actualmente utilizamos variables como son, potencia producida, transcurso del tiempo, velocidad del viento, el momento en el que se generan los datos en el aerogenerador y el momento en el que la información pasa por el motor CEP. Además de estas variables, otra versión posterior de los patrones puede incluir el uso de la dirección del viento, la presión atmosférica, la temperatura en los distintos componentes del aerogenerador, volumen de aceite, suciedad en los rodamientos de las partes mecánicas del aerogenerador, etc. De esta forma, ajustaremos mucho más los patrones para que saquen alertas más específicas.
* **Contrastar las alertas captadas con el log de errores real.** El propio sistema de monitorización de los aerogeneradores ya tiene incorporado un log de errores donde se pueden ver alertas tipo, como son:
  + Paradas del aerogenerador por velocidad excesiva.
  + Paradas por mantenimiento.
  + Cortocircuito en sistema de control del PITCH.
  + Subida de temperatura en el aceite de la multiplicadora.
  + Etc.

Sería bastante interesante poder contrastar la información que las alertas del sistema CEP genera, con el log de errores de la propia máquina, para de esta forma analizar patrones de eventos en búsqueda de poder predecir posibles desperfectos en los aerogeneradores. De esta forma, podríamos aprovechar los momentos en que sopla poco viento para realizar labores de mantenimiento que impidan el desgaste.

* **Adaptar Medit4CEP para la aplicación web.** La aplicación Medit4CEP está dotada de una interfaz gráfica para que cualquier persona que no tenga conocimientos de programación pueda generar un patrón. Una idea para trabajar en el futuro sería elaborar un sistema que se integre en la aplicación web de Eolic Event Consumer, para poder generar patrones a través de una interfaz gráfica como lo hace Medit4CEP.
* **Automatización de la generación de patrones.** Otra línea de trabajo puede ser hacer uso de otras técnicas de Big Data para generar modelos que se puedan traducir en patrones de eventos complejos. Por ejemplo, podríamos almacenar todos los datos que se han dado en varios años, y que un algoritmo en base a la información producida por los aeros cruzada con las alarmas, genere varios modelos, que enuncien patrones y se despleguen en el motor CEP.
* **Automatización de las pruebas.** Las pruebas como hemos visto en Capítulo 6, se han planificado, y efectuado manualmente, sin embargo, no se han automatizado. Como línea de mejora podemos plantearlas de forma automatizada con JUnit y MUnit.

Bibliografía

En esta sección veremos todas las referencias bibliográficas de las que ha nacido este trabajo de fin de grado a través del estándar IEE.

[1] X. Zhao, H. Fan, H. Zhu, Z. Fu, and H. Fu, “The Design of the Internet of Things Solution for Food Supply Chain,” *Proc. 2015 Int. Conf. Educ. Manag. Inf. Med.*, vol. 49, no. Emim, pp. 314–318, 2015.

[2] M. Antonio and R. Borja Díaz, “Energía eólica,” vol. 1, pp. 54–57, 2013.

[3] IBM, “Business Rules Management System | IBM Digital Business Automation.” [Online]. Available: https://www.ibm.com/cloud/automation-software/business-decision. [Accessed: 29-Jan-2018].

[4] R. E. SIEMENS Gamesa, “Gamesa consolida una tecnología propia para el mantenimiento predictivo de sus aerogeneradores,” *29-10-2008*, 2018. [Online]. Available: http://www.siemensgamesa.com/es/comunicacion/noticias/gamesa-consolida-una-tecnologia-propia-para-el-mantenimiento-predictivo-de-sus-aerogeneradores.html?idCategoria=0&fechaDesde=&especifica=0&texto=&fechaHasta=. [Accessed: 14-Mar-2018].

[5] Microsoft Corporation, “La Arquitectura Orientada a Servicios (SOA) aplicada al mundo real,” 2006.

[6] S. O. Tortosa, A. O. Baíllo, and R. B. Plata, “Arquitectura Orientada a Servicios Para La Distribuidos,” pp. 3–10, 2006.

[7] D. Luckham and R. Schulte, “Event Processing Glossary - Version 2.0,” *Processing*, no. July, pp. 1–19, 2011.

[8] S. Moreno Saiz, “Estudio de Arquitecturas Software para Servicios de Internet de las Cosas,” 2015.

[9] Exforsys Inc, “Arquitectura impulsada por eventos SOA 2.0 | Capacitación y consultoría en TI - Exforsys,” *7-7-2007*, 2007. [Online]. Available: http://www.exforsys.com/tutorials/soa/soa-event-driven-architecture.html. [Accessed: 16-Mar-2018].

[10] Chris Taylor, “El papel del CEP en un entorno de Arquitectura Orientada a Servicios,” *12-10-2017*, 2017. [Online]. Available: https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/el-papel-del-cep-en-un-entorno-de-arquitectura-orientada-a-servicios. [Accessed: 16-Mar-2018].

[11] P. Russom, “Big data analytics,” *TDWI Best Pract. Rep.*, pp. 1–35, 2011.

[12] M. Döhring *et al.*, “The convergence of workflows, business rules and complex events: Defining a reference architecture and approaching realization challenges,” *ICEIS 2010 - Proc. 12th Int. Conf. Enterp. Inf. Syst.*, vol. 3 ISAS, no. January 2010, pp. 338–343, 2010.

[13] J. Boubeta, “Model-Driven Development of Domain-Specific Interfaces for Complex Event Processing in Service-Oriented Architectures.,” p. 223, 2014.

[14] J. B. Puig, G. O. Bellot, and I. M. Bulo, “Procesamiento de Eventos Complejos en Entornos SOA: Caso de Estudio para la Detecci{ó}n Temprana de Epidemias,” *VII Jornadas Cienc. En Ing. Serv.*, pp. 63–76, 2011.

[15] Alberto del Barrio de Pablos, “Introducción al procesamiento de Eventos Complejos I - Decide Soluciones,” *2013-09-15*, 2018. [Online]. Available: http://decidesoluciones.es/introduccion-al-procesamiento-de-eventos-complejos-i/. [Accessed: 15-Mar-2018].

[16] B. B. M. Michelson and E. Links, “Event-Driven Architecture Overview 2011,” *Architecture*, 2011.

[17] EsperTech, “Esper - EsperTech.” [Online]. Available: http://www.espertech.com/esper/. [Accessed: 29-Jan-2018].

[18] Esper, “Esper Reference,” *Esper Core Engine*, vol. 2014, p. 2014, 2012.

[19] Oracle, “Introduction to Oracle CQL,” 2009. [Online]. Available: https://docs.oracle.com/cd/E16764\_01/doc.1111/e12048/intro.htm. [Accessed: 28-Mar-2018].

[20] C. G. V, “SQLstream s-Server,” 2018.

[21] SQLStream, “SQLStream downloads and pricing.” [Online]. Available: http://sqlstream.com/download/. [Accessed: 28-Mar-2018].

[22] SAP Company., “SYBASE Price List as of 2 Feb 2018,” *2/2/2018*, 2018. [Online]. Available: http://www.kernelsoftware.com/products/catalog/sybase.html. [Accessed: 28-Mar-2018].

[23] Chakray, “Comparativa ESB Open Source - Chakray,” *13-1-2017*, 2017. [Online]. Available: https://www.chakray.com/comparativa-esb-open-source/. [Accessed: 12-Apr-2018].

[24] Jess Thompson, “FuseSource Is a New Competitor in the Supported Open-Source ESB Market,” *29-10-2010*, 2010. [Online]. Available: https://www.gartner.com/doc/1459835/fusesource-new-competitor-supported-opensource. [Accessed: 12-Apr-2018].

[25] AdroitLogic, “UltraESB-X | Light-weight And Lean Enterprise Integration | AdroitLogic,” *12-4-2018*, 2018. [Online]. Available: https://www.adroitlogic.com/products/ultraesb/. [Accessed: 12-Apr-2018].

[26] Chris Haddad, “ESB Comparison,” *4-8-2012*, 2012. [Online]. Available: https://dzone.com/articles/esb-comparison. [Accessed: 12-Apr-2018].

[27] Z. Cataldi, “La ingeniería del software,” *Una Metodol. para el diseño, Desarro. y evaluación Softw. Educ.*, pp. 33–57, 2000.

[28] Norberto Figuerola, “Desarrollo de Software – ¿Metodología Tradicional o Agil ? | PMQuality,” 2011. [Online]. Available: https://pmqlinkedin.wordpress.com/about/metodologia-tradicional-o-agil/. [Accessed: 19-Mar-2018].

[29] Maria Eugenia Arevalo Lizardo, “Diferencias entre Metodologías Tradicionales y Ágiles #MetodologiasAgiles,” *15-10-2011*, 2011. [Online]. Available: https://arevalomaria.wordpress.com/2011/11/15/diferencias-entre-metodologias-tradicionales-y-agiles-metodologiasagiles/. [Accessed: 19-Mar-2018].

[30] M. Trigas Gallego and A. C. Domingo Troncho, “Gestión de Proyectos Informáticos. Metodología Scrum.,” *Openaccess.Uoc.Edu*, p. 56, 2012.

[31] Kanbantool, “Metodología Kanban.” [Online]. Available: https://kanbantool.com/es/metodologia-kanban. [Accessed: 27-Mar-2018].

[32] J. Joskowicz, “Reglas y prácticas en eXtreme Programming,” *Univ. Vigo. España*, pp. 1–22, 2008.

[33] M. J. Pérez Pérez, “Guía Comparativa de Metodologías Ágiles,” *Univ. Valladolid*, pp. 3–117, 2012.

[34] Eclipse, “Eclipse - The Eclipse Foundation open source community website.” [Online]. Available: https://www.eclipse.org/. [Accessed: 29-Jan-2018].

[35] Eclipse Egit, “EGit.” [Online]. Available: http://www.eclipse.org/egit/. [Accessed: 29-Jan-2018].

[36] MuleSoft, “Anypoint Studio // Documentación de MuleSoft.” [Online]. Available: https://docs.mulesoft.com/anypoint-studio/v/6/. [Accessed: 29-Jan-2018].

[37] CakePHP Community, “¿Qué es CakePHP? y ¿Por qué usarlo? - 2.x,” 2016. [Online]. Available: https://book.cakephp.org/2.0/es/cakephp-overview/what-is-cakephp-why-use-it.html. [Accessed: 12-Apr-2018].

[38] Apache Friends, “XAMPP Página web oficial,” 2018. [Online]. Available: https://www.apachefriends.org/es/index.html. [Accessed: 13-Jun-2018].

[39] Project Management Institute, *Guía de los fundamentos para la dirección de proyectos (guía del PMBOK®)*. 2013.

Anexos

* + 1. Ejemplo de uso de la herramienta

adflñkajf qelrkj qer lqewrj hqlkrj qlhr lqjr lkqrk ckzfjasdlfh qenrl jqelrkj qleh hnlqwerj qw

lajfa lnfladsjf asdfn ladfj aldfladjf ladjflñaeyrqehrn lqwerj oqewrh nqer

* + 1. Manual de usuario

lkadfjla jdflqjer qertkj qer'ijqtej qoier hnqert

kllñkalñkdfg laesrfj lqñwer hnladf fjowqehtn lkwrjeoyhk lñkjje wlkjwenr ñljwer k