

Universidade  
de Vigo

Escola Superior de  
Enxeñería Informática

## Solicitud de aprobación de anteproxecto

Titulación: Enxeñería Informática

Datos do/a Alumno/a			
Nome: Óscar González Fernández		DNI: 44475167 K	
Enderezo: Rúa Padre Sarmiento nº2 4ºB		Localidade: Ourense	
Provincia: Ourense	C.P.: 32005	Teléfono: 988232810	E-mail: ogfernandez@gmail.com

Datos do Proxecto Fin de Carreira	
Título do PFC: DARE: Distributed aAutomator Runtime Environment	
<input checked="" type="checkbox"/> Marque esta casilla se solicita realizar unha memoria cunha documentación diferente da establecida no art. 12.1 do Regulamento de PFC. <b>IMPORTANTE:</b> deberá xustificarse no apartado OBSERVACIÓNS do Anteproxecto	
Director/a do PFC: Florentino Fernández Riverola DNI: 34997200-D	
Codirector/a do PFC (se procede): Daniel González Peña DNI: 44477165-H	
Área de coñecemento: Linguaxes e Sistemas Informáticos	
Departamento: Informática	

A persoa que asina e cos datos que se indican solicita á Comisión de Docencia da Escola Superior de Enxeñería Informática a aprobación do anteproxecto que se acompaña.

Ourense, 30 de marzo de 2009


O/A alumno/a

Vº e Pr. O/A Director/a do proxecto

Asdo: Óscar González Fernández

Asdo: Florentino Fdez-Riverola

Informe da Comisión de Docencia (art. 5 do Regulamento de PFC)	
Desfavorable: <input type="checkbox"/>	Favorable <input type="checkbox"/>
Motivo:	Código PFC asignado:

Outras consideracións (se procede):  	Ourense,            de            de 2009  O/A Presidente/a da Comisión de Docencia   Asdo:

Universidade  
de Vigo

copia para o alumno solicitante

Escola Superior de  
Enxeñería Informática

## Solicitud de aprobación de anteproxecto

**Titulación:** Enxeñería Técnica en Informática de Xestión

Datos do/a Alumno/a			
Nome: Óscar González Fernández			DNI: 44475167 K
Enderezo: Rúa Padre Sarmiento nº2 4ºB			Localidade: Ourense
Provincia: Ourense	C.P.: 32005	Teléfono: 988232810	E-mail: ogfernandez@gmail.com

Datos do Proxecto Fin de Carreira
Título do PFC: DARE: Distributed aAutomator Runtime Environment
<input checked="" type="checkbox"/> Marque esta casilla se solicita realizar unha memoria cunha documentación diferente da establecida no art. 12.1 do Regulamento de PFC. <b>IMPORTANTE:</b> deberá xustificarse no apartado OBSERVACIÓNS do Anteproxecto
Director/a do PFC: <b>Florentino Fernández Riverola</b> DNI: 34997200-D
Codirector/a do PFC (se procede): <b>Daniel González Peña</b> DNI: 44477165-H
Área de coñecemento: Linguaxes e Sistemas Informáticos
Departamento: Informática

A persoa que asina e cos datos que se indican solicita á Comisión de Docencia da Escola Superior de Enxeñería Informática a aprobación do anteproxecto que se acompaña.

Ourense, 30 de marzo de 2009

O/A alumno/a

Vº e Pr. O/A Director/a do proxecto

Asdo: Óscar González Fernández

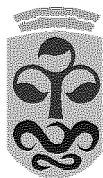
Asdo: Florentino Fdez-Riverola

Informe da Comisión de Docencia (art. 5 do Regulamento de PFC)	
Desfavorable: <input type="checkbox"/>  Motivo:	Favorable <input type="checkbox"/>  Código PFC asignado:

<b>Outras consideracións (se procede):</b>	<p>Ourense,        de        de 2009</p> <p>O/A Presidente/a da Comisión de Docencia</p> <p>Asdo:</p>
--	--

---

copia para o Centro



Universidade  
de Vigo

Escola Superior de  
Enxeñería Informática

## Solicitud de tribunal de proxecto fin de carreira

(Recoméndase ler os artigos 10, 11 e 17 do Regulamento de PFC da ESEI)

☐ Marque esta casilla se solicita a avaliación do PFC por parte do director do Proxecto Fin de Carreira. **IMPORTANTE:** Opción so dispoñible para a titulación de Enxeñeiro Técnico en Informática de Xestión.

**Secretario do Tribunal:** Moisés Cid Deza

**Secretario suplente:** José Manuel Sorribes Fernández

Relación de profesores propostos para Presidente do Tribunal:

1	Reyes Pavón Rial
2	Rosalía Laza Fidalgo
3	Eva Lorenzo Iglesias
4	Lourdes Borrajo Diz
5	José Baltasar García Perez-Schofield

# DESCRIPCIÓN DO ANTEPROXECTO

Adapta-lo tamaño do documento según sexa necesario. Consulta-lo art. 5 do Regulamento de PFC da ESEI)

TÍTULO: DARE. Distributed aAutomator Runtime Environment

## Introducción

En la actualidad la WWW presenta una cantidad ingente de información. Un problema al que nos enfrentamos es que esta información es difícil de procesar y extraer automáticamente, ya que en su inmensa mayoría está destinada a ser leída por humanos. Aunque actualmente existen esfuerzos e iniciativas como la web semántica para codificar toda esa información de una manera más formal y tratable por computadoras, hoy por hoy no es posible acceder a la mayor parte de la información disponible en la web por medio de procedimientos automáticos.

En este contexto surge aAutomator, una herramienta creada por el grupo SING (*Sistemas Informáticos de Nueva Generación*) para la extracción automática de información procedente la web. La utilización de Aautomator está fundamentalmente orientada a su aplicación en el ámbito bioinformático, donde existe multitud de información no estructurada, valiosa desde un punto de vista biológico, de difícil acceso y procesamiento. En este sentido, un uso típico de la herramienta consiste en extraer datos de un sitio web, combinarlos con otra información proveniente de un portal distinto, y finalmente mostrar los resultados generando un nuevo fichero HTML personalizado.

Actualmente, aAutomator ofrece un entorno de ejecución para unos agentes software llamados *robots*. Los robots pueden visitar varias páginas de forma concurrente y extraer la información deseada, todo ello siguiendo una especificación de comportamiento basada en un descriptor XML. El grupo SING ha desarrollado paralelamente una herramienta que permite la generación de forma visual de este fichero (aAutomatorFX).

Aunque el uso de XML como lenguaje de representación es adecuado para comunicar la herramienta visual con aAutomator, dista mucho de ser ideal para su uso directo en la creación de robots on-line. En concreto, la aproximación existente actualmente está más orientada a facilitar su interpretación por aAutomator, que a ser creada directamente por programadores. A mayores, existen situaciones en las que es preferible prescindir de la utilización de aAutomatorFX. Por ejemplo, un programador desearía poder invocar directamente aAutomator en sus programas, pero actualmente se ve obligado a tener que crear previamente el fichero descriptor en XML de un robot.

aAutomator ha sido desarrollado utilizando Java como lenguaje de programación. Aún siendo esta una opción respaldada por una amplia comunidad de usuarios, sería además interesante poder ofrecer la posibilidad de ejecución de robots desde otras plataformas. Sería por lo tanto deseable, el permitir que aAutomator fuera utilizado por clientes de otras plataformas.

Con el fin de mejorar las carencias actuales de aAutomator y de posibilitar su utilización por una comunidad mayor de usuarios se realiza la presente propuesta.

## Objetivos

Para lograr la correcta consecución de las ideas planteadas en el apartado de introducción de la presente propuesta, se identifican los siguientes objetivos clave en el presente proyecto:

**Facilitar la creación de los robots.** Para ello se diseñará e implementará un nuevo lenguaje de programación de propósito específico: lo que recientemente se denomina DSL (Lenguaje Específico de Dominio). Este lenguaje permitirá la especificación de los robots de manera más sencilla y sin perder poder expresivo. Es decir, se podrá especificar la misma funcionalidad que a través del XML. Se potenciarán dos objetivos:

- *El lenguaje será lo más conciso posible sin sacrificar legibilidad.*
- *Simplificar los casos de uso más comunes.*

**Ofrecer aAutomator como un servicio.** Esto permitirá acceder a la funcionalidad de aAutomator desde otras arquitecturas. Este servicio será accesible por medio de HTTP y seguirá el estilo arquitectónico REST (*Representational State Transfer*). Este servicio está orientado a ser consumido por software, y no directamente por los usuarios finales. El programa cliente siempre proporcionará un robot más los parámetros de entrada para el robot. Se distinguirán dos tipos de operaciones:

- *Ejecución síncrona.* Se procederá inmediatamente a la ejecución del robot y a la devolución de los resultados. El robot se podrá proporcionar en XML o en la DSL definida por este proyecto. Los resultados se devolverán como texto plano, ya que los robots pueden construir resultados de tipo muy heterogéneo.
- *Ejecución periódica.* En vez de ejecutar inmediatamente el robot suministrado, se ejecutará cada cierto periodo de tiempo. Este periodo de tiempo será configurable por el administrador del servicio. Los resultados de la ejecución del robot se publicarán en la URL devuelta al cliente. Los resultados anteriores se mantendrán en un histórico con un tope de antigüedad configurable por el administrador.

**Crear librerías para acceder a aAutomator desde varias plataformas.** Para ello se crearán librerías en varios lenguajes (Java, Python, etc.) que serán clientes del servicio. De este modo, se verificará que se puede integrar el servicio de aAutomator en distintas plataformas.



## Descripción técnica

La presente propuesta se compone de dos partes principales: (i) la creación de un traductor que generará el XML consumido por aAutomator y (ii) la oferta de aAutomator como servicio.

El traductor tomará un lenguaje origen más accesible para el usuario y lo transformará en el descriptor XML que necesita aAutomator. El lenguaje origen a desarrollar se puede considerar un lenguaje específico de dominio, correspondiente con un lenguaje sencillo desarrollado para facilitar una tarea en concreto. En este sentido ofrecerá un mayor grado de abstracción. Awk, make o SQL son ejemplos de lenguajes específicos de dominio. Para implementar la DSL se evaluarán dos posibilidades: (i) su implementación a partir de las facilidades proporcionadas por un lenguaje existente o (ii) la creación de un nuevo lenguaje desde el inicio. La primera posibilidad es viable en lenguajes con una sintaxis flexible y con facilidades para metaprogramación. La segunda alternativa ofrece una mayor flexibilidad, puesto que no existirían las limitaciones del sistema anfitrión. Como inconveniente, tendría una mayor dificultad de implementación y menor integración con el sistema anfitrión.

Se estudiará cuál de las dos opciones es más adecuada y se realizará una implementación. Cualquiera de las dos opciones permite mantener la infraestructura actual de aAutomator intacta, ya que se traducirá desde el nuevo lenguaje origen al XML que consume aAutomator.

A continuación se muestra un diagrama explicativo de la primera parte del proyecto. Se puede apreciar que el traductor recibirá una instancia válida de la DSL a definir y generará un documento XML que especificará el comportamiento del robot. Finalmente aAutomator recibirá esa especificación y la ejecutará.



En relación con la segunda parte de este proyecto (implementar aAutomator como servicio), se llevará a cabo utilizando el protocolo HTTP. HTTP es una opción bien soportada y documentada que facilitará la integración del servicio en el mayor número de plataformas. El servicio seguirá el estilo arquitectónico REST. REST define una serie de restricciones: el estado y la funcionalidad de un servicio son abstraídos en recursos, son accesibles a través de una sintaxis uniforme para definir direcciones (URL), un conjunto limitado de verbos para acceder a, y modificar, representaciones de los recursos, las respuestas se pueden cachear y es un protocolo cliente-servidor sin estado. Estas restricciones proporcionan una serie de beneficios en cuanto a extensibilidad y escalabilidad, además de facilitar la implementación de clientes.

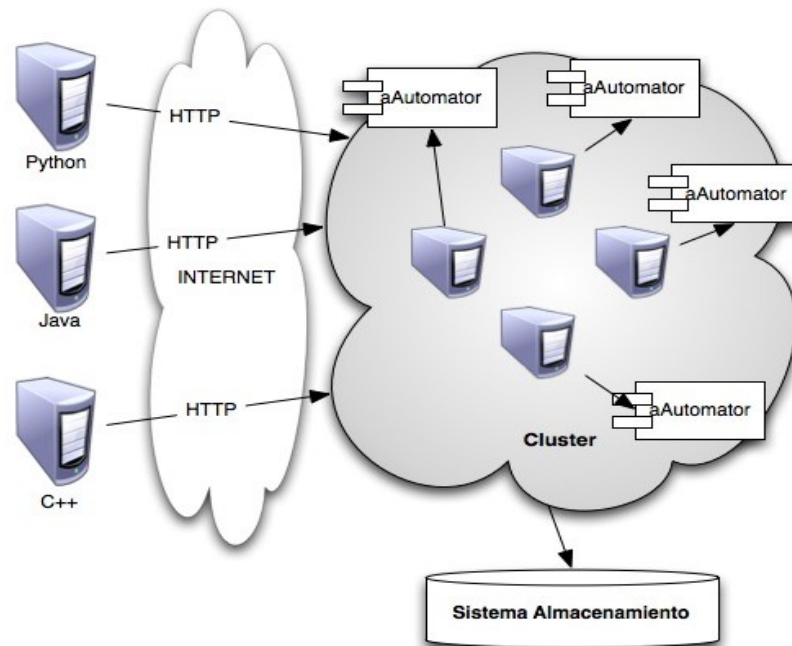
El servicio se implementará de modo que se pueda ejecutar tanto en un único servidor como en un cluster de equipos genéricos. Una de las restricciones de REST, que las peticiones al servicio sean sin estado, facilita la distribución de carga entre el cluster de equipos. Para verificar la validez del diseño se realizarán varias pruebas. Estas pruebas comprobarán la escalabilidad de la solución propuesta y servirán de ayuda para la toma de decisiones en el diseño e implementación del servicio.

Por otro lado el sistema requiere que se ejecuten robots periódicamente, por lo que se hace necesario un sistema de almacenamiento. Este sistema de almacenamiento guardará la descripción de los robots, su frecuencia de ejecución y los resultados producidos. Para llevar a cabo su implementación surgen varias opciones. Una primera alternativa consiste en emplear una base de datos relacional. La ventaja es que se proporciona un modelo bien conocido y flexible para almacenar datos. El mayor inconveniente es que se podría convertir en un cuello de botella del cluster. Una forma de aliviar este problema sería la utilización de las capacidades de replicación que incorpore la base de datos. Una segunda opción sería la utilización de una base de datos clave-valor (como Berkeley DB). Dado que los datos a almacenar no son muy complejos, podría ser una solución más directa y eficiente que una base de datos relacional. El inconveniente es que al instalarse en un único equipo, podría convertirse también en el cuello de botella del cluster. Una última alternativa pasa por el empleo de una base de datos distribuida no relacional del estilo de Project Voldemort o Cassandra. Estas bases de datos replican los datos en varios servidores y



proporcionan buena escalabilidad horizontal. Su mayor inconveniente es su inmadurez. Realizado el estudio correspondiente, se elegirá una de estas alternativas para la implementación del sistema de almacenamiento.

La figura que consta a continuación proporciona una visión general de la implementación de aAutomator como servicio. Se pueden apreciar varios equipos clientes que accederían a través de HTTP al cluster de equipos. Cada equipo del cluster tendrá una o varias instancias de aAutomator ejecutándose. Todos los equipos del cluster podrán acceder al sistema de almacenamiento.



En cuanto al traductor del nuevo lenguaje a XML de aAutomator, se implementará sobre la plataforma Java por la existencia de numerosos lenguajes de script que podrían ser adecuados para definir la DSL interna, además de la existencia de varias herramientas para definir DSL externas. Por ejemplo, si al final se opta por construir una DSL externa se podrán usar librerías como JavaCC o ANTLR.

La implementación del servicio también se realizará sobre la plataforma Java, debido a la madurez de esta plataforma así como la existencia de numerosas librerías para muchas de las tareas requeridas. La nueva especificación JAX-RS también será de gran ayuda a la hora de facilitar la implementación del servicio siguiendo las restricciones de REST.

## Fases do trabalho e estimación temporal

Se ha planificado el trabajo en tres etapas: una primera etapa en la que se estudiará el funcionamiento de aAutomator, una segunda etapa en la que se desarrollará la DSL y una tercera etapa en la que se implementará aAutomator como servicio. Se considera conveniente ir realizando la documentación conforme se va realizando el trabajo, por lo que será una etapa que se desarrollará en paralelo salvo al final del proyecto que se realizará en solitario. Desarrollar la DSL y ofrecer aAutomator como servicio se compondrán de varias fases. Estas fases, como se aprecia en el diagrama de Gantt, se solapan en gran medida porque se ha considerado que es altamente conveniente un alto grado de retroalimentación entre las distintas fases. Por ejemplo, después de una breve etapa de análisis y diseño en solitario se empieza a realizar la implementación pero sin abandonar la fase de diseño que se ve así continuamente refinada con la información obtenida al implementar.

Dedicación semanal prevista (en horas/semana):	
Fase	Estimación temporal (en semanas)
Estudio aAutomator	2
DSL	8
Análisis y diseño	4
Implementación	5
Pruebas DSL	3
Cluster	8
Análisis y diseño	4
Implementación	5
Pruebas Verificación	4
Documentación	20
<b>TOTAL PROXECTO</b>	<b>22</b>

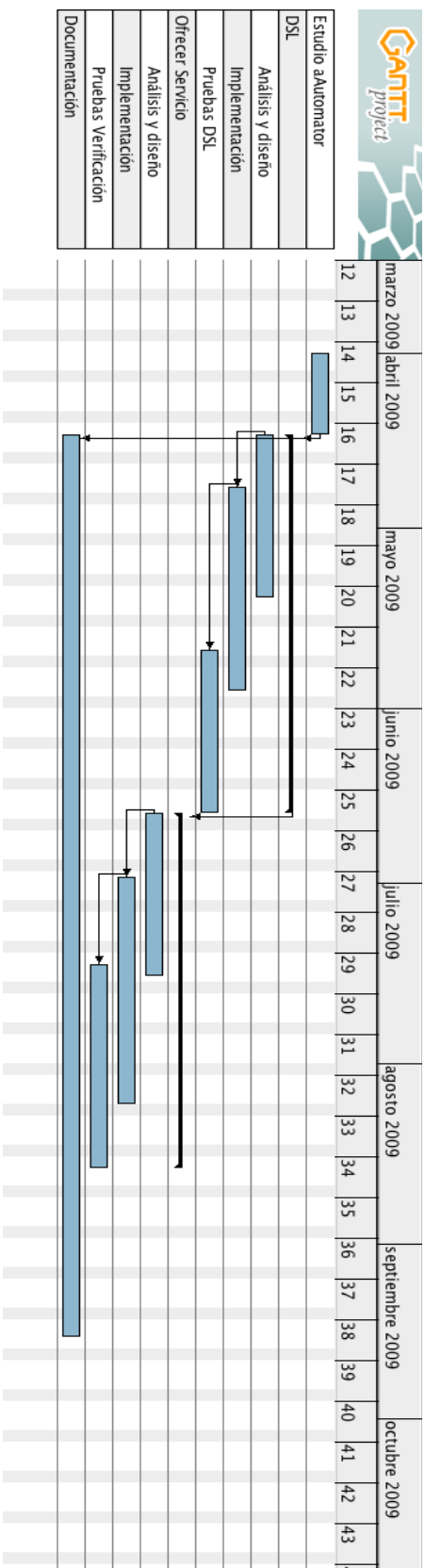


Diagrama de Gantt

## Medios materiales necesarios.

### Software

- JDK 1.6.
- IDE java, p. ej. eclipse.
- War container, p. ej. Tomcat o Glassfish.
- LaTeX

### Hardware

- Equipo desarrollo.
- Cluster de equipos genéricos para pruebas.

## Bibliografía.

D. Glez-Peña; J.R. Méndez; F. Fdez-Riverola (2007) *aAUTOMATOR: Herramienta flexible para la extracción de información en sitios web bioinformáticos*. Conferencia Ibero-Americana WWW/Internet 2007: CIAWI 2007. Vila Real, Portugal. 07 - October

Language Workbenches: The Killer-App for Domain Specific Languages? Martin Fowler.  
<http://martinfowler.com/articles/languageWorkbench.html>

Tesis Roy Fielding. <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

ANTLR. <http://www.antlr.org/>

JavaCC. <https://javacc.dev.java.net/>

Jruby. <http://jruby.codehaus.org/>

Project Voldemort. <http://project-voldemort.com/>

Berkeley DB. <http://www.oracle.com/database/berkeley-db/je/index.html>

Cassandra. <http://code.google.com/p/the-cassandra-project/>

JAX-RS. <http://jcp.org/en/jsr/detail?id=311>

## Observacións.

Dado el carácter de investigación de la presente propuesta: la creación de un traductor de una DSL y la codificación de las funcionalidades de aAutomator como servicio, destacar que la documentación no se adapta perfectamente al esquema definido por el artículo 12.1 del Reglamento de PFC.

En este sentido, dejar constancia de que trataremos de ceñirnos lo máximo posible al esquema propuesto, aunque algunos documentos pueden cambiar en función de su viabilidad y utilidad para el usuario final. Hacer constar que el usuario final del producto desarrollado se comunicará con él a nivel de programación. En este sentido, será de especial utilidad la generación de manuales y documentación técnica que posibilite un rápido aprendizaje de las interfaces de programación establecidas.