



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería en Informática



TFG del Grado en Ingeniería Informática  
Conversión de la aplicación docente  
Thoth a JavaScript  
Documentación Técnica



Presentado por Francisco Javier Páramo Arnáiz  
en Universidad de Burgos — 2 de julio de 2017

Tutores: D. Álvar Arnaiz González  
Dr. César Ignacio García Osorio

---

# Índice general

---

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>III</b>
<b>Índice de tablas</b>	<b>V</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	24
<b>Apéndice B Especificación de Requisitos</b>	<b>27</b>
B.1. Introducción . . . . .	27
B.2. Objetivos generales . . . . .	27
B.3. Catálogo de requisitos . . . . .	29
B.4. Especificación de requisitos . . . . .	30
<b>Apéndice C Especificación de diseño</b>	<b>51</b>
C.1. Introducción . . . . .	51
C.2. Diseño de datos . . . . .	51
C.3. Diseño procedural . . . . .	54
C.4. Diseño arquitectónico . . . . .	55
<b>Apéndice D Documentación técnica de programación</b>	<b>58</b>
D.1. Introducción . . . . .	58
D.2. Estructura de directorios . . . . .	58
D.3. Manual del programador . . . . .	60
D.4. Compilación, instalación y ejecución del proyecto . . . . .	62
D.5. Pruebas del sistema . . . . .	65

*ÍNDICE GENERAL*

II

<b>Apéndice E Documentación de usuario</b>	<b>67</b>
E.1. Introducción . . . . .	67
E.2. Requisitos de usuarios . . . . .	67
E.3. Instalación . . . . .	68
E.4. Manual del usuario . . . . .	68
<b>Bibliografía</b>	<b>75</b>

---

# Índice de figuras

---

A.1.	Burndown del sprint 2 . . . . .	3
A.2.	Burndown del sprint 3 . . . . .	5
A.3.	Burndown del sprint 4 . . . . .	6
A.4.	Burndown del sprint 5 . . . . .	7
A.5.	Burndown del sprint 6 . . . . .	8
A.6.	Burndown del sprint 7 . . . . .	9
A.7.	Burndown del sprint 8 . . . . .	10
A.8.	Burndown del sprint 9 . . . . .	12
A.9.	Burndown del sprint 10 . . . . .	13
A.10.	Burndown del sprint 11 . . . . .	14
A.11.	Burndown del sprint 12 . . . . .	15
A.12.	Burndown del sprint 13 . . . . .	17
A.13.	Burndown del sprint 14 . . . . .	18
A.14.	Burndown del sprint 15 . . . . .	19
A.15.	Burndown del sprint 16 . . . . .	20
A.16.	Burndown del sprint 17 . . . . .	21
A.17.	Burndown del sprint 18 . . . . .	22
A.18.	Burndown del sprint 19 . . . . .	23
B.1.	Diagrama de caso de uso general. . . . .	32
B.2.	Diagrama de caso de uso Crear/Modificar Gramática[1]. . . . .	32
B.3.	Diagrama de caso de uso Algoritmos. . . . .	33
B.4.	Diagrama de caso de uso Opciones. . . . .	36
C.1.	Diagrama de clases del núcleo de la gramática[1]. . . . .	52
C.2.	Diagrama de clases de la estructura de Símbolo. . . . .	52
C.3.	Diagrama de clases de los Algoritmos de la gramática [1]. . . . .	53
C.4.	Diagrama de clases del Parser de la gramática [1]. . . . .	54
C.5.	Diagrama de clases del Inicio de sesión junto con el registro. Ambos se comunican con la base de datos. . . . .	54

C.6. Proceso de inicio de sesión en la aplicación. . . . .	55
C.7. Proceso de comprobación de una gramática. . . . .	56
C.8. Diagrama de paquetes dentro de Client. . . . .	57
D.1. Arquitectura RPC simple. . . . .	61
D.2. Diagrama sobre la comunicación entre el cliente y el servidor en GWT. . . . .	62
D.3. Diagrama de clases que muestra como utiliza el parser de la gramática. Diagrama realizado con ObjectAid. . . . .	63
D.4. Dirección local que aparece al ejecutar en modo SuperDevMode. .	64
D.5. Acción de desplegar la aplicación con App Engine. . . . .	64
D.6. Opciones de despliegue. . . . .	65
E.1. Login en Thoth. Es lo que primero se ve al entrar en la aplicación.	68
E.2. Pantalla de registro. . . . .	69
E.3. Fallo al introducir el <i>email</i> . Se muestra el campo en rojo. . . . .	69
E.4. Pantalla principal con el menú de la aplicación y el panel para crear una gramática. . . . .	70
E.5. En la parte superior comprobar gramática y abajo renombrar. Ambos botones son los mismos que en las otras versiones de Thoth. .	70
E.6. Comprobación de una gramática y como se actualiza la tabla inferior.	71
E.7. Renombrado de símbolos. . . . .	71
E.8. Ejemplo de una vista del algoritmo Eliminación de SA. . . . .	72
E.9. Vista del algoritmo para el Cálculo de First Follow. . . . .	72
E.10. Vista del reconocimiento con TASP. Se aprecia la zona donde introducir la palabra y los botones. . . . .	73
E.11. Menú con el nombre del usuario. El desplegable muestra el cierre de la sesión. . . . .	74
E.12. Mensaje de confirmación para cambiar el idioma. . . . .	74

---

# Índice de tablas

---

A.1. Tabla de los costes totales . . . . .	25
A.2. Tabla de software y sus licencias . . . . .	26
B.1. Caso de uso Registro. . . . .	33
B.2. Caso de uso Inicio de sesión. . . . .	34
B.3. Caso de uso Cierre de sesión. . . . .	34
B.4. Caso de uso Crear gramática. . . . .	35
B.5. Caso de uso Modificar gramática. . . . .	35
B.6. Caso de uso Buscar y renombrar símbolo. . . . .	36
B.7. Caso de uso de algoritmos. . . . .	37
B.8. Caso de uso de Eliminar símbolos no terminables. . . . .	38
B.9. Caso de uso de Eliminar símbolos no alcanzables. . . . .	39
B.10. Caso de uso de Eliminar símbolos anulables. . . . .	40
B.11. Caso de uso de Eliminar producciones no generativas. . . . .	41
B.12. Caso de uso de Limpiar gramática. . . . .	42
B.13. Caso de uso de Eliminar recursividad directa. . . . .	43
B.14. Caso de uso de Eliminar recursividad indirecta. . . . .	44
B.15. Caso de uso de Eliminar recursividad. . . . .	45
B.16. Caso de uso de Factorizar por la izquierda. . . . .	46
B.17. Caso de uso de Forma normal de Chomsky. . . . .	47
B.18. Caso de uso de First y follow. . . . .	48
B.19. Caso de uso de Reconocimiento con TASP. . . . .	49
B.20. Caso de uso de Cambio de idioma. . . . .	50
B.21. Caso de uso de Mostrar Acerca de. . . . .	50

## *Apéndice A*

---

# Plan de Proyecto Software

---

### A.1. Introducción

En el apartado aquí descrito se comenta cómo ha sido el seguimiento de este proyecto. La metodología utilizada es Scrum que se emplea en los proyectos ágiles y que consiste en un desarrollo colaborativo en el que se hacen entregas periódicas de las diferentes partes del proyecto hasta juntar un todo que forme el proyecto final.

Para ello hemos hecho uso de herramientas como GitHub y ZenHub que ayudan en la labor de definir objetivos semanales o Milestones que a su vez están compuestos por problemas denominados issues. También hemos podido ver el progreso semanal, comunicar aspectos relacionados con el trabajo o agrupar los issues según el tipo.

Como en cualquier proyecto de este tipo, a veces, algunos objetivos semanales se han modificado debido a que resultaban más complejos de lo esperado o inútiles para el fin buscado o no accesibles como es el caso, por ejemplo, de los elementos para el desarrollo de interfaces con Vaadin, que requerían una licencia de pago muy elevada. Todo ello forma parte de la gestión de proyectos ágiles.

### A.2. Planificación temporal

El desarrollo del proyecto está compuesto por entregas periódicas denominadas sprints y que han tenido una duración de una semana, a excepción del sprint número nueve que por motivos de vacaciones académicas tuvo que ser del doble de tiempo. Aunque al principio el seguimiento del proyecto no fue el correcto en cuanto al manejo de las herramientas GitHub y ZenHub, posteriormente se trató de mejorararlo.

Se trató de llevar una carga constante de trabajo para mantener un ritmo con el que se llegase a la fecha de entrega, pero en alguno de los últimos sprints se tuvo que aumentar la cantidad porque había más materia sobre la que trabajar y mejorar.

En total fueron unas 21 semanas en las que se simulan los *sprints* mediante *milestones* con objetivos definidos en las reuniones entre tutores y alumno, desde el mes de febrero hasta junio. Cada *issue* va dentro de *milestone* determinado, especificando el tipo de *issue* y el tiempo estimado hasta su resolución. Se hace uso de *Git* como gestor de versiones.

### Sprint 0 (1/2/2017 - 8/2/2017)

En este primer *Sprint* determinamos que posibles herramientas de traducción pueden ser útiles para este proyecto sin profundizar demasiado. Se hace también la primera toma de contacto con Thoth para ver su funcionamiento básico.

Por otro lado, analizamos las posibilidades del proyecto, determinando los caminos en los que puede derivar dicho proyecto. Es decir, en estos primeros pasos no sabemos como funcionan estas herramientas, es por ello que el resultado sea más simple del esperado o por el contrario resulten complicaciones que lleven un tiempo mayor al esperado.

Además de esto, hacemos una introducción a las herramientas de documentación y gestión. Por ello aclaramos que:

- Para la gestión de versiones haremos uso de GitHub, asociando un gestor de tareas llamado Zenhub que funciona como un *plugin* en el buscador.
- Realizo un primer contacto con L<sup>A</sup>T<sub>E</sub>X

En esta primeras semanas no me aclaro mucho con el funcionamiento del gestor de versiones, y es por ello que no hago un buen uso de los *commits* ni de los *issues* que proporciona Github.

### Sprint 1 (8/2/2017 - 15/2/2017)

Ya en la segunda semana realizó una evaluación más exhaustiva de las herramientas de traducción, analizando los pros y los contras de ellas. Por lo tanto tomamos la decisión de centrarnos en GWT como la principal y con la que vamos a llevar a cabo el proyecto.

Definimos como tareas semanales:

- Evaluar los pros y contras de las diferentes herramientas de traducción de código.

- Documentar esa evaluación con L<sup>A</sup>T<sub>E</sub>X, familiarizándome con la manera de documentar.
- Realizar las primeras pruebas con GWT, de una forma simple.

En esta semana hago alguna prueba simple con GWT, gracias a los ejemplos que proporciona la página oficial a modo de tutorial. También realicé alguna prueba simple con JSweet para ver su funcionamiento real y si es, de verdad, útil para poder llevar a cabo el proyecto. En el último ejemplo que hago me ocurre un problema con GWT que no termino de solucionar y que me obliga a posponer la prueba de ese ejemplo para el siguiente *sprint*.

También profundizo algo más en la documentación y en el uso de las herramientas para documentar. Hasta este punto no he asociado las tareas o *issues* con los *milestones* y por lo tanto no queda registrado el tiempo del *sprint*.

### Sprint 2 (15/2/2017 - 22/2/2017)

En este tercer *sprint*, lo primero que hago es solucionar el anterior problema que tuve con GWT. Consistía en configurar bien el entorno de Eclipse y el *plugin* para poder hacer la ejecución de GWT con *Super Dev Mode*, alternativa implantada por los desarrolladores para evitar la necesidad de instalar una extensión de GWT en el buscador en el cual se lanza la aplicación. El *burndown chart* del *sprint* 2 se ilustra en la figura A.1.

Figura A.1: Burndown del sprint 2



También en esta semana descubrimos una nueva herramienta relacionada con el tema, que se llama Vaadin y que nos puede servir, por lo menos para hacer una comparativa más completa de las herramientas de traducción.

La parte más importante de esta semana es la prueba de traducción del *core* de Thoth, que aunque no sale como esperamos, ha resultado útil para conocer con mayor profundidad tanto la aplicación como la herramienta.

Por lo tanto como tareas para este *sprint*:

- Solucionar el error surgido con GWT.
- Realizar pruebas de traducción con el *core* de Thoth.
- Incluir la nueva herramienta en la comparativa.

Muy a mi pesar, en el *milestone* de esta semana, aunque he pasado cada tarea al estado de realizada o «done» no las he cerrado hasta darnos cuenta al final del *sprint*, es por ello que el *burndown* queda de esta manera.

### Sprint 3 (22/2/2017 - 1/3/2017)

Ya en la cuarta semana se hace un intento más completo para traducir la aplicación. Como pudimos comprobar en la semana pasada, GWT no traducía las librerías de la parte visual de Thoth. Es por ello que decidimos hacer un ejemplo de forma manual que consiste en programar parte de la vista que está asociada a las partes más relevantes del núcleo.

Por ejemplo, nos centramos en hacer una prueba con la gramática, que está dentro del núcleo, creando una pantalla con un «text label» para comprobar si funcionaba la traducción de esa parte del núcleo. De esta forma podríamos ver como hacía la traducción de todo el núcleo, ya que las otras partes de las que se compone son similares en el uso de librerías y bibliotecas.

Quedan así asignadas las tareas para del *sprint* número tres:

- Transformar la Gramática del núcleo de Thoth.
- Transformar el Autómata del núcleo.
- Transformar la simulación del núcleo de Thoth.
- Documentar toda esta parte.

Al final solo se pudo llevar a cabo la tareas de la Gramática y la documentación porque no se pudo avanzar a las otras como muestra la imagen [A.2](#).

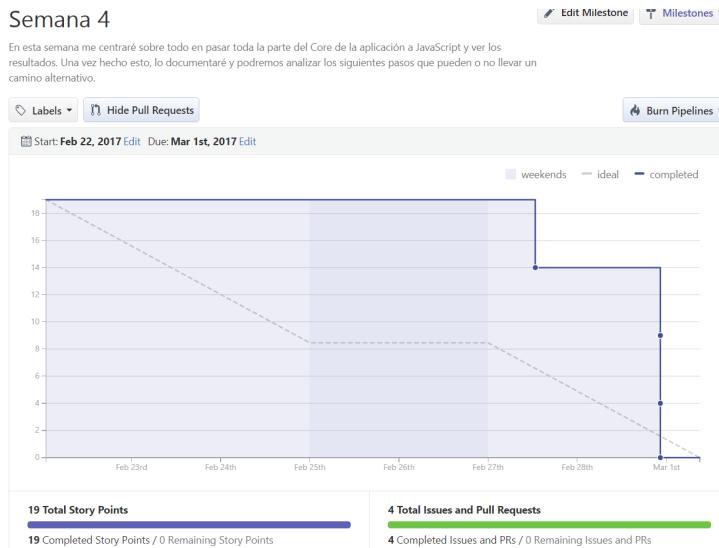


Figura A.2: Burndown del sprint 3

Nuestra idea era probar a tratar de traducir todo, núcleo incluido, ejecutando esas partes en el cliente de GWT pero vimos que esto no es posible. Intentamos hacerlo de varias formas eliminando partes no esenciales de la aplicación para reducir errores de compilación hasta darnos por vencidos y ver que esa no era la solución.

#### Sprint 4 (1/3/2017 - 8/3/2017)

Hemos intentado pasar la aplicación en la parte del servidor y probar por nuestra cuenta. Sólo hemos metido el núcleo en el paquete servidor para ver que surgía. Como vimos que no había una comunicación entre el cliente y el servidor hicimos varios intentos, probando con el paquete de *shared* o compartido, pero GWT también traduce ese paquete a JavaScript por lo que seguía dando los mismo errores que en el cliente.

Por lo tanto en este *sprint* tenemos esta tareas:

- Solucionar un error en la traducción.
- Realizar pruebas cliente-servidor con el núcleo de la aplicación.
- Cambios y mejoras en la documentación.

Una vez nos dimos cuenta de que el fallo de tratar de hacer la aplicación en la parte del servidor era que GWT no reconocía algunas de las librerías claves,

### Semana 5

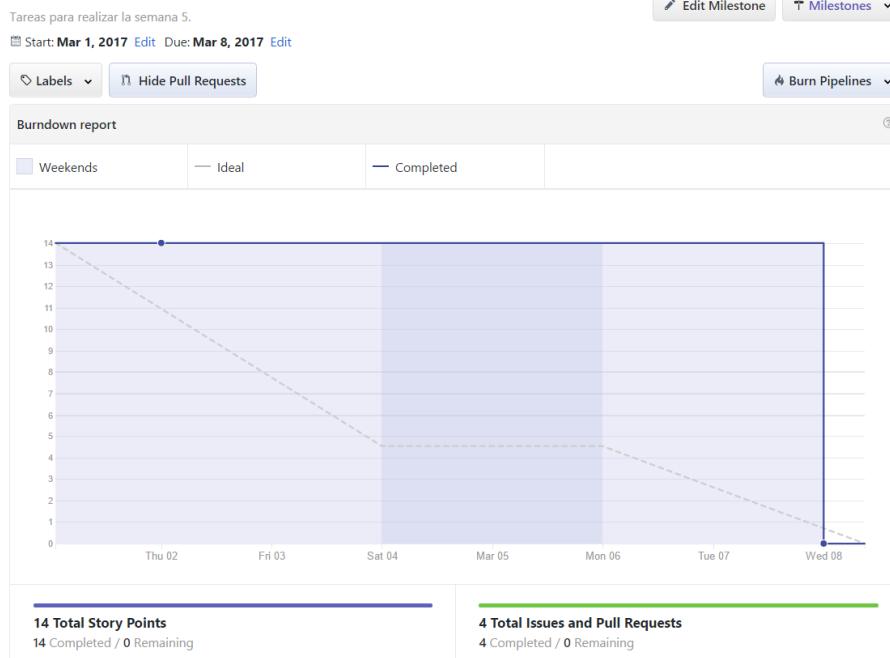


Figura A.3: Burndown del sprint 4

tanto en la parte visual como en el núcleo de la aplicación, decidimos buscar otros caminos alternativos.

El funcionamiento de GWT consiste en traducir a «JavaScript» la parte del cliente y la compartida. En consecuencia decidimos hacer pruebas en las que las partes mas fundamentales del núcleo se encontrasen en el lado del servidor. De esta forma cuando el cliente necesitase hacer algún uso de métodos con librerías no reconocidas por GWT, simplemente llamase al servidor ya que este podría soportar dichos métodos.

En los primeros intentos nos dimos cuenta de que estas llamadas no se podían hacer de una forma simple, ya que la comunicación entre cliente y servidor no funcionaba y no obteníamos los resultados que esperábamos. Aún así seguimos haciendo pruebas para asegurarnos, metiendo dentro del paquete «compartido» las partes del núcleo mas cercanas a lo que nosotros consideramos la vista. El problema seguía siendo esa comunicación. Interpretaba como del lado del cliente lo que nosotros queríamos que formara parte del servidor, dando errores debido a que GWT no trabaja con esas librerías.

### Sprint 5 (8/3/2017 - 15/3/2017)

Principalmente, en este quinto *sprint*, se llevan a cabo las pruebas para entender y poder evaluar la comunicación cliente-servidor, por medio de unos ejemplos. Además de eso, planteamos la idea de realizar un *login* y validación de usuarios, pero solo como idea, ya que no es prioritario al cien por cien importancia.

#### Semana 6

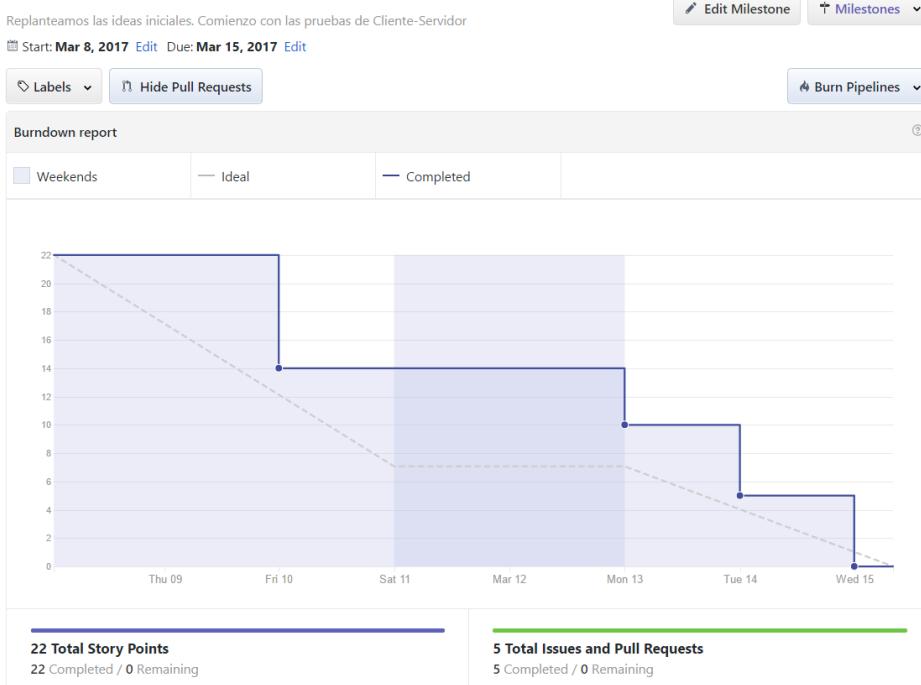


Figura A.4: Burndown del sprint 5

Así que en este sprint tenemos esta tareas:

- Ejemplo cliente-servidor con GWT.
- Login y validación en GWT.

Como se puede ver en el *burndown chart* del *sprint 5* en la figura A.4, se trabajó de forma continua en esta semana. La comunicación entre el cliente y el servidor se lleva a cabo mediante la comunicación RPC (*Remote Procedure Call*). Es por ello que se hace necesario entender y practicar el funcionamiento de esta práctica. Los ejemplos realizados han sido dos: el primero es un ejemplo o tutorial ofrecido por la página oficial de GWT, que consiste en hacer un visor

del stock que cambia de forma aleatoria sus valores. Y el segundo ejemplo consistió en hacer un pequeño ejemplo de llamada de funciones con más clases que en el anterior.

### Sprint 6 (15/3/2017 - 22/3/2017)

En esta semana nos metemos ya en serio con la aplicación propiamente dicha. Lo primero que hacemos es conseguir que la comunicación entre en núcleo (ya hecho) y su uso sea fluido. Para ello lo que hacemos es incluir algunas partes en el cliente y otras en el servidor. En el cliente sólo podemos incluir las clases más simples, más primitivas de la aplicación porque su contenido es entendido por GWT y puede hacer la traducción a JavaScript sin problemas de librerías.



Figura A.5: Burndown del sprint 6

Por lo tanto las tareas son básicamente dos:

- Llevar a cabo el primer prototipo o sección de la aplicación.
- Documentar y corregir errores anteriores en la documentación.

La realización del prototipo nos lleva tiempo ya que se necesita comprender muy bien el funcionamiento interno de Thoth y así poder definir un diseño del software adecuado según ese funcionamiento. Una vez hecho eso parece simplificarse los problemas que al principio se tenían. Puede verse el *burndown chart* de este *sprint* en la figura A.5.

### Sprint 7 (22/3/2017 - 29/3/2017)

Parece ser que en esta octava semana ya podemos empezar a trabajar más en la programación del diseño e implementar funcionalidades. Lo primero que tenemos que hacer es que esa comunicación entre métodos de resultados «más» visibles e integrarlos en una «GUI» denominada en español como interfaz gráfica de usuario. Para más detalles sobre la semana, ver la figura A.6

Así es que definimos como tareas las siguientes:

- Reestructurar la aplicación y limpiar el código.
- Mejorar la GUI con Vaadin, ya que la anterior es muy básica.
- Implementar el algoritmo "Eliminar símbolos no terminales"

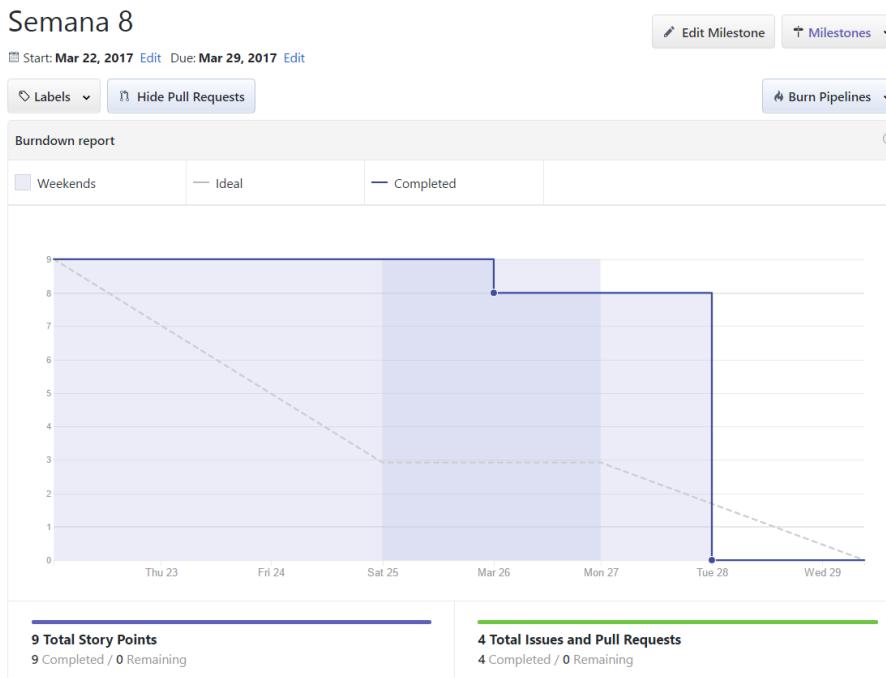


Figura A.6: Burndown del sprint 7

Primeramente hay que organizar el código haciéndolo más legible y limpiarlo de comentarios, y pruebas para ver su funcionamiento. Queremos que los resultados queden de una forma similar al Thoth original, para conservar su esencia, usabilidad, y buen diseño. Para ello hacemos uso de Vaadin, una herramienta que se puede utilizar como un *plugin* en eclipse y que se integra perfectamente con GWT.

### Sprint 8 (29/3/2017 - 5/4/2017)

Al principio de este octavo *sprint* o 9 semana, estuve pendiente de la respuesta por parte de Vaadin sobre si me podía conceder o no una licencia gratuita, así que me centré en incluir el algoritmo de eliminación de símbolos no terminales, mejorando lo que tenía hasta el momento, que era solo una pequeña interfaz con una funcionalidad que no era la exacta. Por ello me centré en corregirla. Así lo hicimos.

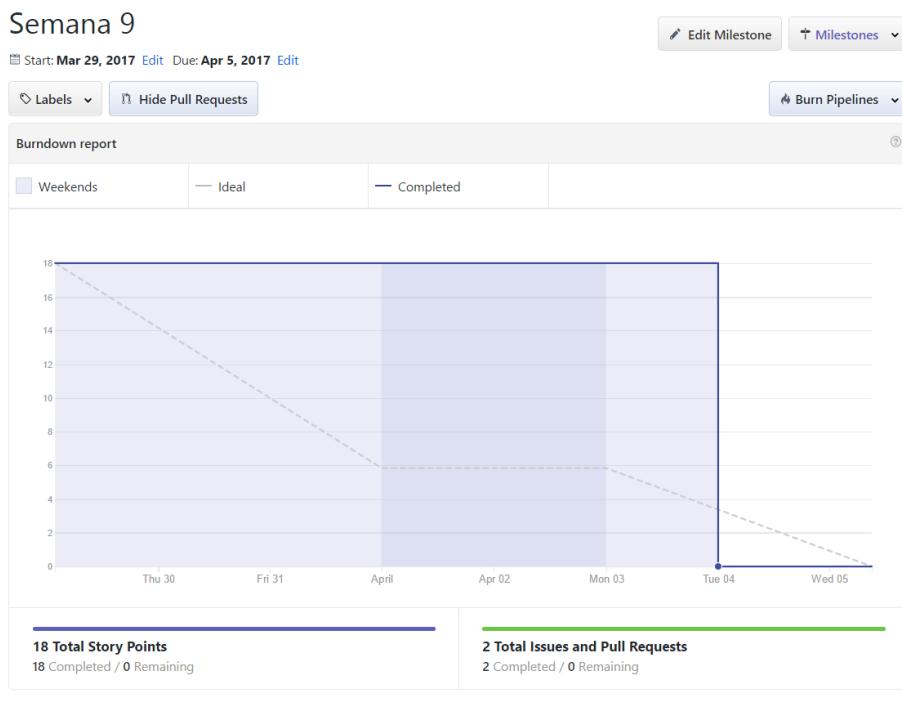


Figura A.7: Burndown del sprint 8

Posteriormente llegó la respuesta de Vaadin explicando que no me podía conceder la licencia y que buscarse otras opciones dentro de las que ellos mismos me ofrecían. Al principio lo intenté pero resultó que no supe como hacerlo. Lograba hacer un proyecto de Vaadin pero no conseguía incluirlo en el mío propio de GWT.

Así que comencé a hacer la interfaz por mi mismo, con las posibilidades de GWT. Puede verse en el *burdown chart* de la ilustración A.7 como la carga de trabajo se concentra sobre todo en el final.

Las tareas para esa semana fueron:

- La inclusión del algoritmo de Eliminación de símbolos no terminales, de una forma mejorada.
- Recabar información sobre internacionalización en GWT.

Logramos hacer que funcionase el algoritmo como queríamos y estuvimos mejorando el diseño y la funcionalidad de la interfaz de dicho algoritmo.

### Sprint 9 (05/4/2017 - 19/4/2017)

Es el *sprint* más largo hasta la fecha ya que incluye dos semanas de trabajo por coincidir con las vacaciones de Semana Santa. Engloba fundamentalmente el hacer los demás algoritmos, con sus respectivas vistas. El *burdown chart* del *sprint* 9 se ilustra en la figura A.8.

- Implementar los algoritmos restantes.
- Mejorar el código, haciéndolo más comprensible y organizar la parte visual.
- Hacer la documentación sobre la parte del diseño y sobre las herramientas que tiene la UBU relacionadas con este proyecto.

En estas dos semanas no pude realizar todos los algoritmos como en un principio pretendía porque la parte visual se alejaba de lo que tenía hecho hasta el momento, es decir, necesitaba hacer nuevas vistas que llevaban más tiempo del pensado y no me dio tiempo, pero si que incluí la mayor parte de ellos. Estuvimos estudiando como aplicar el resaltado de las producciones. En el Thoth original utilizaba una librería, java swing, que ya comprobamos anteriormente que no podía ser incluida en GWT en este proyecto así que buscamos otras alternativas como hacer un resaltado a mano con HTML. De momento lo dejamos ahí.

### Sprint 10 (19/4/2017 - 26/4/2017)

Una vez pasadas las vacaciones de Semana Santa, el proyecto tiene una forma más madura y podemos ir haciendo añadidos más funcionales a la aplicación. Es por ello que decidimos empezar con la internacionalización, viendo

### Semanas 10 y 11.

Como en este milestone coincide con las vacaciones de Semana Santa, durará el doble que los anteriores.

Start: Apr 5, 2017 Edit Due: Apr 19, 2017 Edit

Edit Milestone Milestones

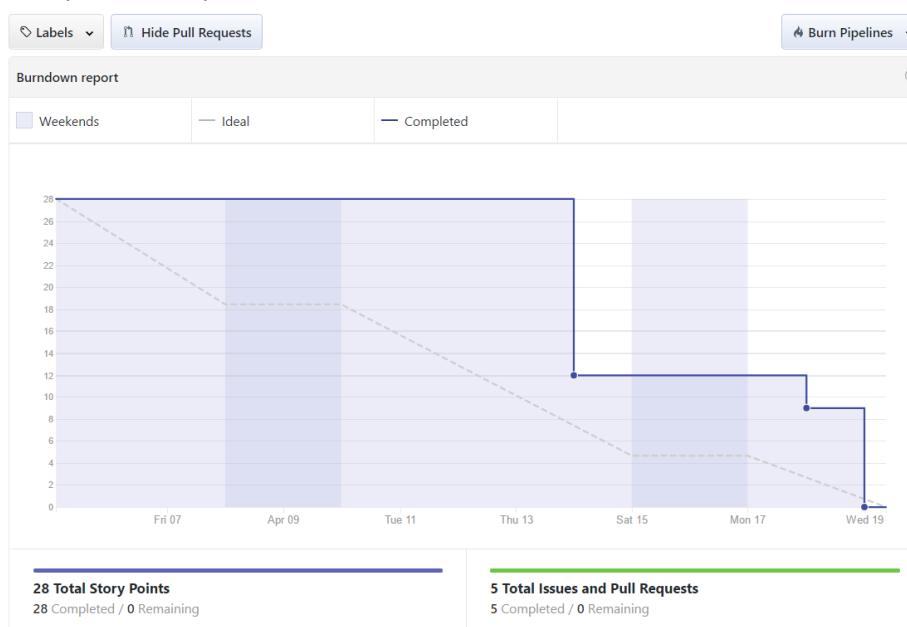


Figura A.8: Burndown del sprint 9

como funcionaba para poder entenderla y una vez entendida incluirla en el proyecto. Además de esto, después de comprobar en la semana pasada que el resultado lo podíamos hacer con HTML nos dedicamos de lleno a ello. La verdad es que nos llevó muchos quebraderos de cabeza. El porqué no era otra cosa que teníamos que tener en cuenta todo el funcionamiento interno, «destripar» que contenía cada variable, como funcionaban cada método de GWT etc. Todo este trabajo nos llevó mucho tiempo, ya que estuvimos todo el *sprint* a base de prueba y error con los diferentes algoritmos hasta que funcionó en todos de la forma que deseamos. En el *burndown chart* A.9 se puede apreciar como no es hasta el final del *sprint* cuando se cierran los *issues*.

Estas fueron las tareas correspondientes a esta semana.

- Aplicar internacionalización.
- Cambiar la visualización de los paneles.
- Resultado de producciones en los algoritmos con HTML.
- Incluir los demás algoritmos.

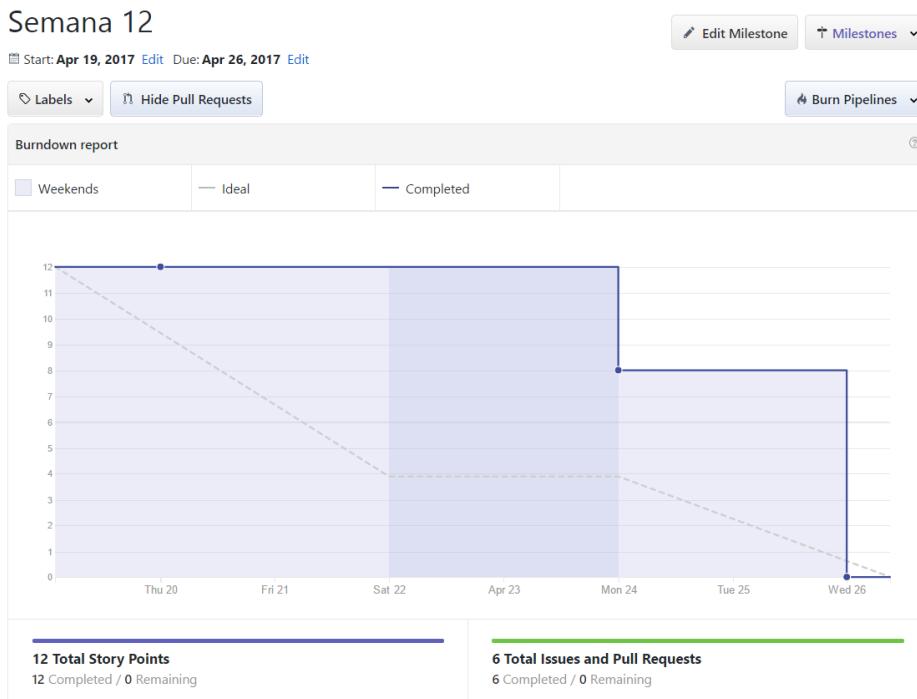


Figura A.9: Burndown del sprint 10

- Incluir pestañas y elementos de Vaadin.

La parte visual, es decir, la inclusión de pestañas en realidad no lo llegamos a aplicar bien, de la forma deseada y se trató en el siguiente *sprint*. Además los elementos de Vaadin no se pudieron incluir ya que la única forma de añadirlos a un proyecto GWT son con licencias de pago. La visualización de los paneles simplemente fue una recolocación para que quedase más agradable a la vista.

### Sprint 11 (26/4/2017 - 03/5/2017)

En esta semana seguimos acumulando un pequeño «bug» en el resultado de las producciones, y es que la interpretación de las comillas dobles «“» no las interpretaba como nosotros queríamos y por ello no hacía un buen borrado del resultado, manteniéndose en cada paso. La solución fue simplemente sustituir las comillas dobles por las comillas dobles de java «/«». Pero las dos grandes tareas de esta semana consistieron en aplicar el algoritmos *FirstFollow* y un *TabLayoutPanel* para hacer un panel con pestañas como los que se ven en los navegadores web.

Las tareas quedaron así.

- Solucionar *bug* en el resultado.

- Implementar algoritmo *FirstFollow*.
- Incluir *TabLayoutPanel* para hacer diferentes pestañas.

El algoritmo *FirstFollow* fue fácil hasta el momento de mostrar los resultados en las tablas. Resulta que al tratar de imprimir los resultados de tipo *Object* pasándolos como un *string* no los reconocía bien. Este tipo de errores no son fáciles de detectar en GWT, ya que la forma para verlos claramente es imprimiendo los valores por pantalla y tratando de analizarlos, en que formato están etc. Al final descubrimos que el problema era al tratar de pasara al formato string valores que GWT interpretaba como *undefined* y que no eran más que valores en blanco, espacios en blanco dentro de la tabla. Se pueden ver mas detalles en la figura A.10.

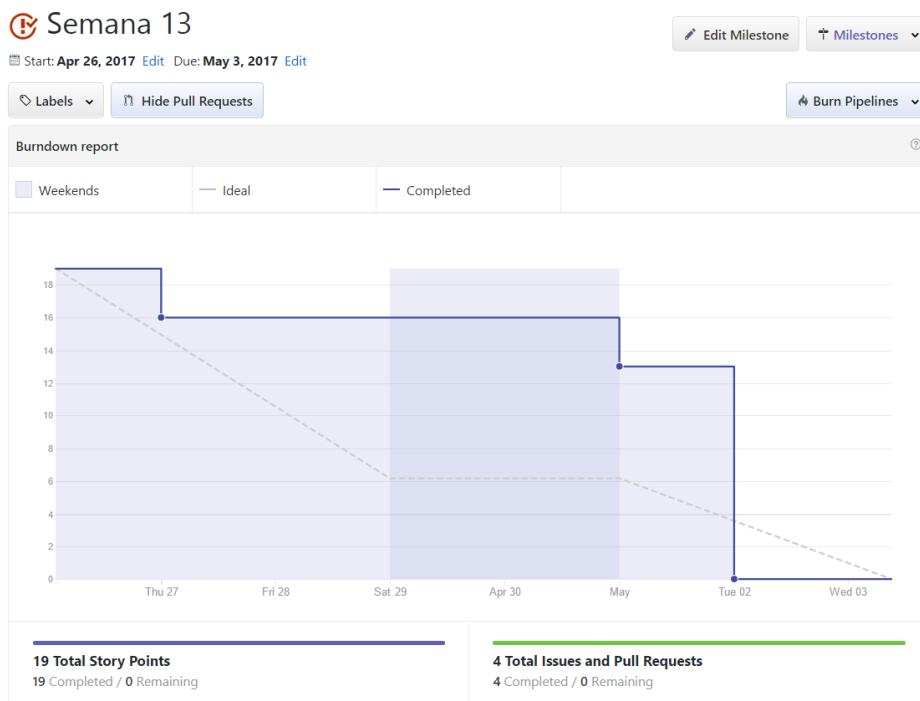


Figura A.10: Burndown del sprint 11

El otro gran quebradero de cabeza fue el tratar de incluir pestañas en la aplicación. No es nada fácil añadir una pestaña con una nueva vista sin que esta reemplace otra, se superponga o cualquier otra cosa. A día de hoy no sabemos bien el porqué de esto al cien por cien. Lo que si es cierto es que no es completamente necesaria y se puede hacer un reemplazamiento de otras vistas.

### Sprint 12 (03/5/2017 - 10/5/2017)

Entrado en el mes de mayo, la recta final del proyecto, y con una aplicación base ya consolidada, por denominarlo así, nos dedicamos sobre todo ha hacer mejoras en cuanto a temas visuales y funcionales. Además de esto el objetivo en este mes es hacer un registro e inicio de sesión de los usuarios para así tener un control detallado del uso. Con todo esto por un lado, también era hora de dedicarle un porcentaje de tiempo mayor a documentar el proyecto. Véase la ilustración A.11.

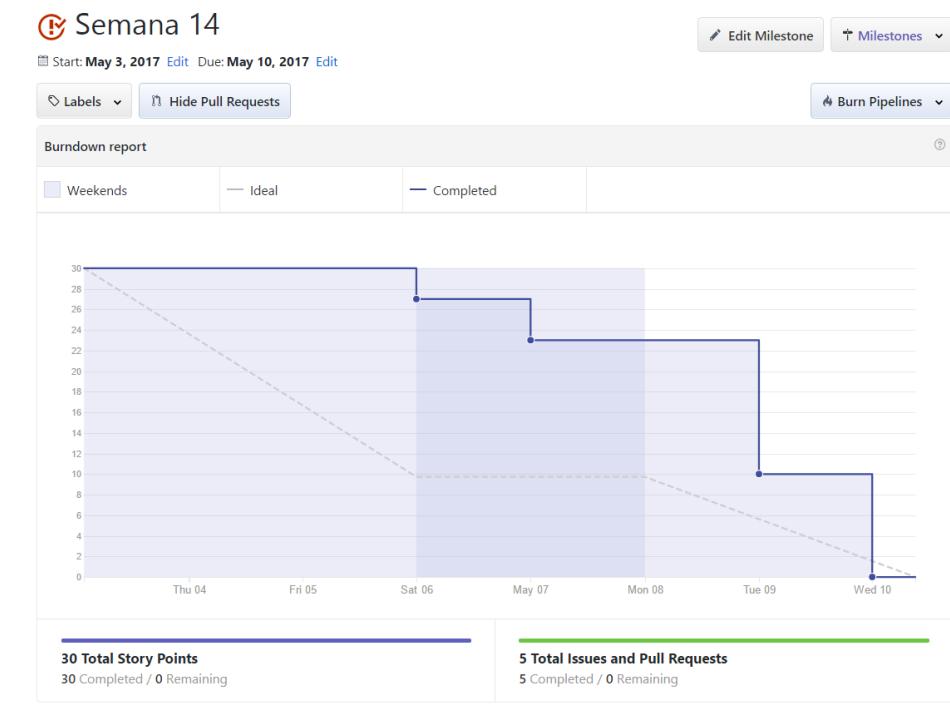


Figura A.11: Burndown del sprint 12

Para esta semana determinamos que las tareas quedaría de esta manera:

- Comenzar a estudiar y probar el inicio de sesión.
- Actualizar la documentación.
- Mejoras en la interfaz (botones, colores y formas).
- Añadir un control de errores más elaborado y hacer la aplicación más robusta.
- Cambiar la funcionalidad de los botones en los algoritmos.

La tarea de cambiar la funcionalidad de los botones en los algoritmos hace referencia a que regrese o vaya a una vista u otra dependiendo del botón pulsado o de las acciones asociadas a ello. Las mejoras visuales son hechas a mano, como ya he comentado anteriormente, y es por ello que es un trabajo largo, en el que hay que estar compilando continuamente para ver los efectos resultantes, que no siempre son los esperados. El inicio de sesión acabó en un ejemplo con varios errores y algo chapucero que sirvió para comprender el funcionamiento por lo que no se mejoró nada en la aplicación.

### Sprint 13 (10/5/2017 - 17/5/2017)

Llegados a este punto comenzamos a «pulir» lo que tenemos añadiendo estilos con CSS y mejorando pequeños aspectos como es el caso de la vista de los algoritmos. Al resaltar los pasos en un algoritmo, tuvimos la «necesidad» de utilizar un *RichTextArea* para poder manejar el código HTML y poder así subrayarlo. Más tarde nos dimos cuenta de que este tipo de área no se puede inhabilitar para que no se editable y esto se trata de un bug de GWT, la función está pero no hace nada. Otros usuarios también se quejaron de ello<sup>1</sup>. Así que decidimos meterlo todo dentro de un panel como texto normal en HTML, que al fin y al cabo era lo que ya habíamos hecho. El *burdown chart* se ilustra en la figura A.12.

Los issues quedaron definidos así:

- Cambiar las interfaces de las gramáticas.
- Incluir nuevos estilos.
- Documentar aspectos como la introducción y los conceptos teóricos.
- Añadir una nueva clase para hacer la internacionalización.
- Solucionar un bug que surgió sobre la vista de los algoritmos.
- Aplicar un pequeño ejemplo de inicio de sesión.

Se pudo hacer un ejemplo en un proyecto a parte para hacer un simple inicio de sesión sin base de datos. Para lograr el inicio se hizo por medio de *cookies*. El objetivo siguiente sería poder aplicar esto a nuestro proyecto pero con una pequeña base de datos en la cual se registraban los usuarios y luego hacían inicio de sesión. También añadimos clases para que encapsularan la internacionalización y poder lanzar mensajes de aviso.

---

<sup>1</sup><https://code.google.com/archive/p/google-web-toolkit/issues/1488>

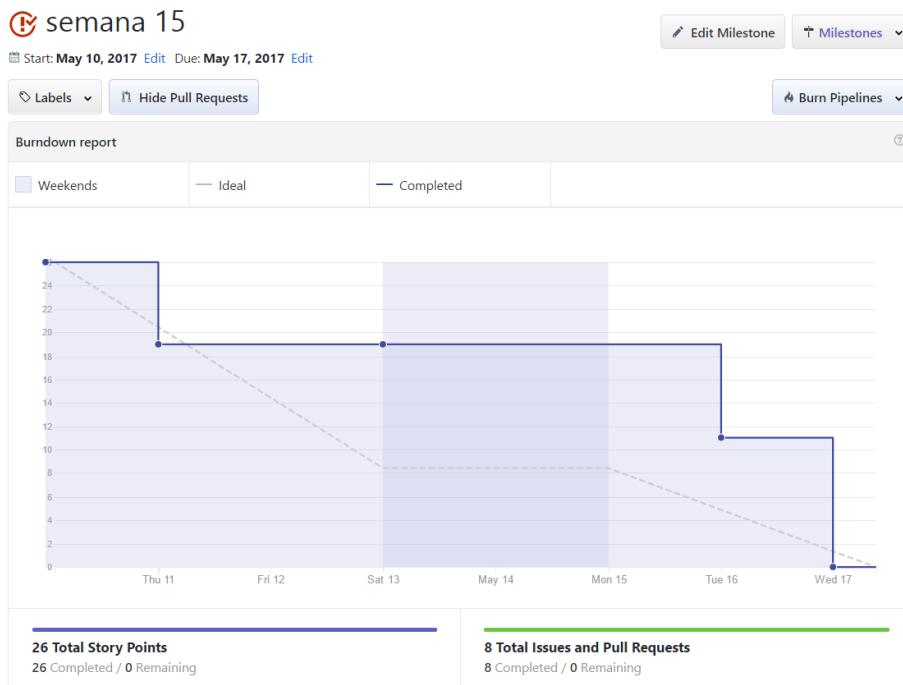


Figura A.12: Burndown del sprint 13

### Sprint 14 (17/5/2017 - 24/5/2017)

En el *sprint* 14, es decir la decimosexta semana, se centraron los esfuerzos sobre todo en hacer un registro e inicio de sesión de usuarios. La idea inicial era la de guardar información de nombre, apellido, un correo que sirviera de identificador único y una contraseña. Para ello se creó una pequeña base de datos, casi podríamos hablar de repositorio. Una vez registrados esos datos el usuario tendría un perfil creado y podría acceder a la aplicación.

Las tareas se centraron en documentar y hacer el registro y sesión de usuarios:

- Añadir un *login* o inicio se sesión.
- Corregir aspectos de la documentación.
- Incluir alguna mejora en el menú principal.

Además de esto, surgió un *bug* o pequeño error con los estilos «css». Y es que al crear varios módulos para realizar varias vistas en la aplicación los estilos, que son propios de cada módulo, se solapaban y se mezclaban mostrando en un módulo el estilo del otro y viceversa. El error no tenía mucha

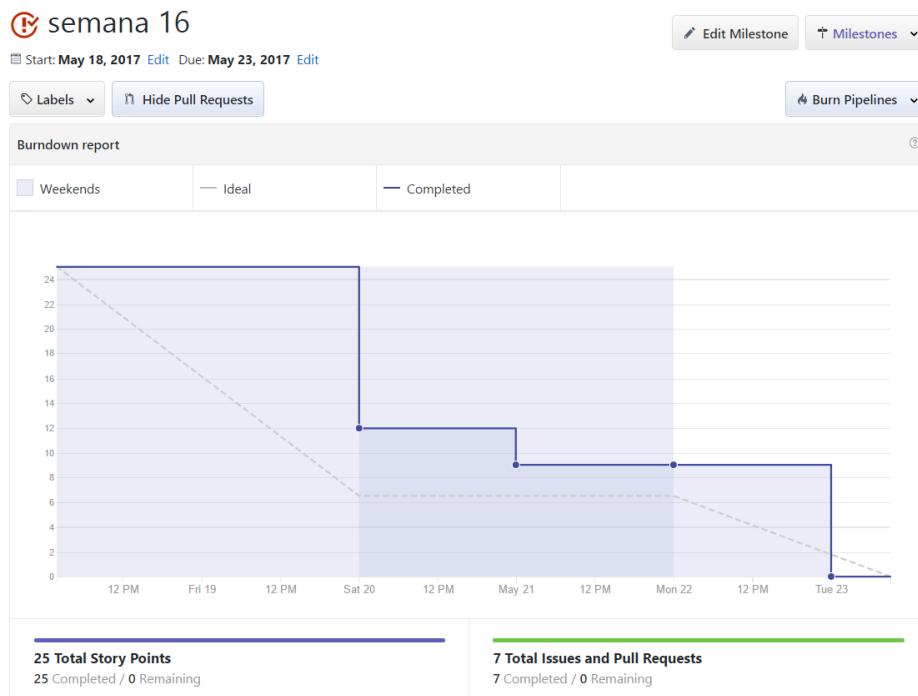


Figura A.13: Burndown del sprint 14

complicación y se pudo solucionar sin problema. Puede verse el *burndown* en la ilustración A.13. El registro e inicio de sesión me llevo tiempo, dos *sprints* el hacer lo básico y otros dos «pulirlo» ya que los conceptos en cuanto a carga de módulos, y comunicación RPC eran en parte nuevos. Lo demás fue principalmente documentación, mejorarla y continuar con lo anterior.

### Sprint 15 (24/5/2017 - 31/5/2017)

Ya en la semana 17, una vez que tenemos la base de datos haciendo el registro de usuarios, nos metemos con la codificación de las contraseñas. Consiste en crear una clave secreta generada por medio de una función con la que ciframos la contraseña para mantener la privacidad de los datos de los usuarios registrados.

Los *issues* para este sprint son:

- Cifrado de contraseñas.
- Registro de más información.
- Mejorar el aspecto del *login*.

El registro de más información consiste en añadir a la base de datos la fecha de registro de un usuario o la última vez que inició sesión en la aplicación. La resolución de los *issues* queda reflejada en la siguiente gráfica ilustrada en la figura A.14.

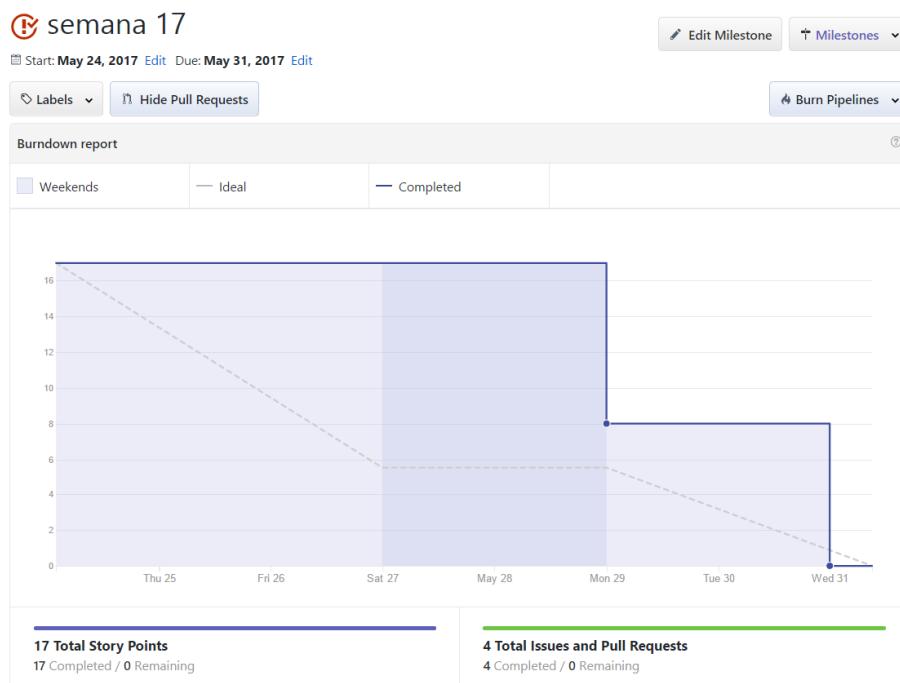


Figura A.14: Burndown del sprint 15

Además de esto, añadimos algunos estilos a las vistas de inicio y registro de sesión aunque posteriormente lo volveremos a modificar, hasta estar a gusto con el diseño.

### Sprint 16 (31/5/2017 - 7/6/2017)

En el *sprint* número dieciséis tratamos de pulir un poco lo que tenemos. Para ello limpiamos el código refactorizando y dando nombres a las variables con un sentido más específico, comentando un poco el código y eliminando aquello que no nos es útil. Arreglamos algún *bug* que aparece en el renombrado de nodos por ejemplo.

Para esta semana las tareas son las siguientes.

- Cifrado de contraseñas.
- Validación de *e-mail*.

- Añadir tratamiento de errores en el *login*.
- Refactorizar el código.

Seguimos con el cifrado de contraseñas porque decidimos mejorarlo y dejarlo más claro y con más posibilidades, dependiendo de las interacciones y de las variables podemos crear un nivel de cifrado más simple o más complejo.

A su vez comprobamos que la dirección de correo introducida cumpla con la «plantilla» de una dirección real. Consiste en hacer una comprobación por medio de HTML y una función en Java bastante simple. El *burndown* queda de esta manera [A.15](#).

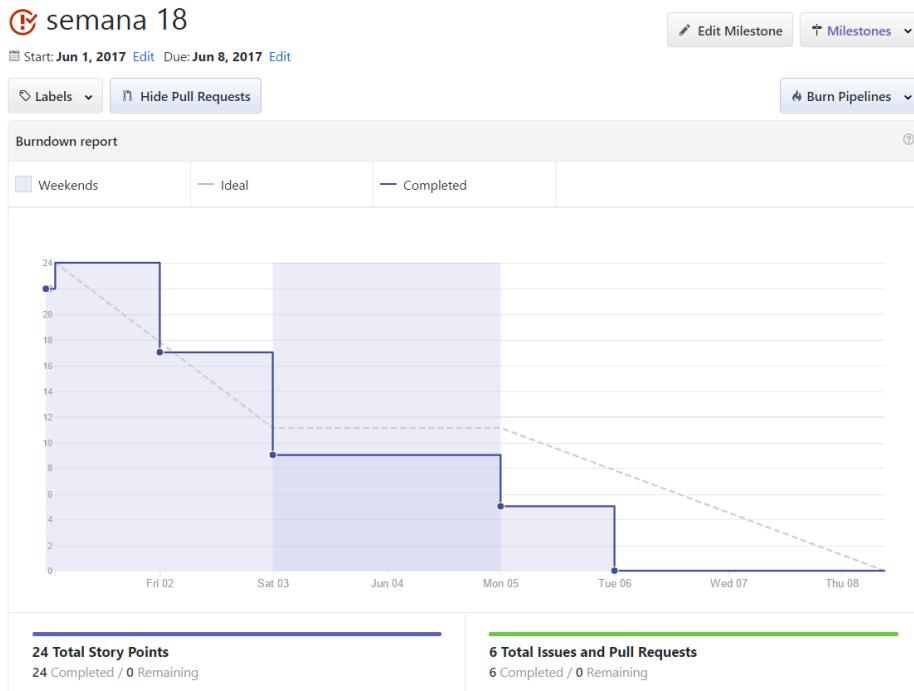


Figura A.15: Burndown del sprint 16

Además de esto añadimos imágenes de fondo, el escudo de la universidad de Burgos en el inicio de sesión y registro de usuarios. Esta semana tiene un poco menos de carga de trabajo porque coincide con fechas de exámenes.

### Sprint 17 (7/6/2017 - 14/6/2017)

En esta semana, ya centrados en la documentación, pretendemos hacer una mejora importante en el proyecto que consiste en incluir un control de

sesiones. Hasta ahora el usuario podía registrarse y hacer *login*, pero aún así podía acceder a la edición de una gramática.

Como hicimos en el *sprint* numero 12 en un prototipo, trataremos de aplicar esto a nuestro proyecto.

Para esta semana las tareas son las siguientes.

- Documentación de conceptos teóricos.
- Documentación de del seguimiento del proyecto.
- Cambios en la interfaz de los algoritmos.
- Control de sesiones.

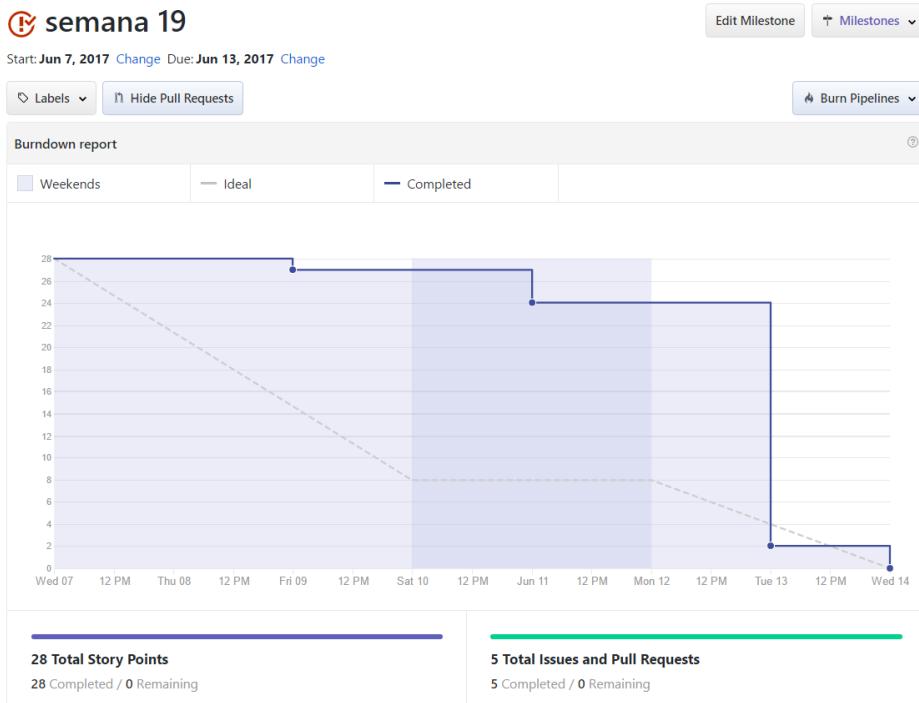


Figura A.16: Burndown del sprint 17

Nos lleva trabajo hacer el control de sesiones porque hay que probar cada acción volviendo a compilar el proyecto una y otra vez. Se pueden ver la gráfica de esta semana [A.16](#).

### Sprint 18 (14/6/2017 - 21/6/2017)

Como ya hemos hecho en las anteriores semanas, documentaremos la memoria sobre todo lo concerniente a las técnicas y herramientas y los conceptos teóricos.

Plantemos el *logout* como un botón en el menú, con el nombre del usuario. Para ello se deberán hacer dos peticiones al servicio, una para obtener el nombre de la sesión de la base de datos y otra para cerrar la sesión.

Comenzamos también con las pruebas del sistema.

Tareas de la semana:

- Documentación herramientas y técnicas.
- Solucionar un *bug* de la tabla TASP.
- Pruebas de sistema.
- Comprobación de email.
- Cierre de sesión.

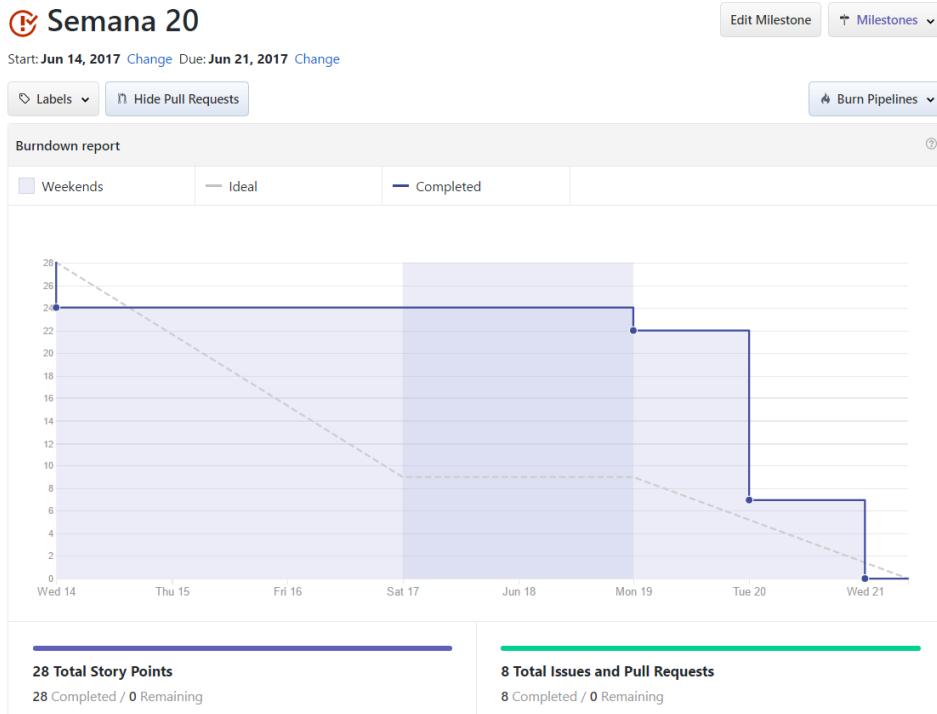


Figura A.17: Burndown del sprint 18.

Hacemos además un cambio en el cifrado de las contraseñas, siendo más rápido y sin posibilidad de recuperar la contraseña original. Se hace alguna comprobación de email, pero sin llegar asegurar que existe realmente. Se puede ver la gráfica aquí [A.17](#).

### Sprint 19 (21/6/2017 - 28/6/2017)

Para terminar, en el último *sprint* a parte de documentar todo lo que falta, tenemos que solucionar pequeños errores y hacer el despliegue de la aplicación.

Las tareas son:

- Desplegar la aplicación.
- Solucionar un *bug* el borrado de la palabra en TASP.
- Pruebas de sistema sobre fallos.

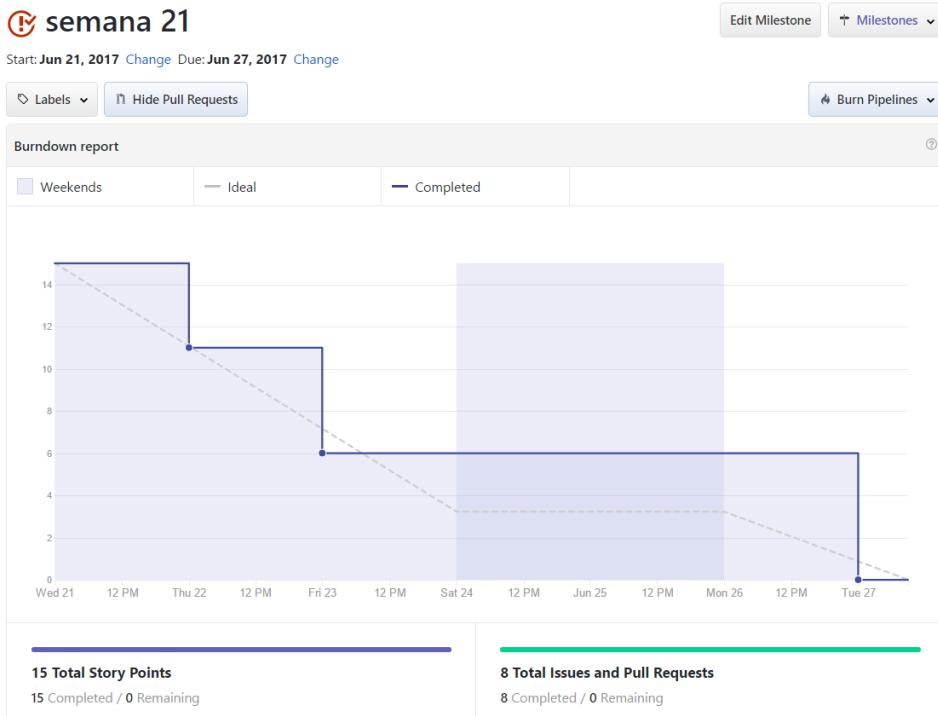


Figura A.18: Burndown del sprint 19.

A parte de las pruebas para comprobar que el inicio de sesión también probamos que se producen errores cuando deben producirse y debemos comprobar que los mensajes que se muestran son los indicados.

Logramos hacer el despliegue después de varios intentos, y es que en un principio pensamos hacerlo con un *host* gratuito. Más tarde vimos que desde *App Engine* se podía hacer de manera «sencilla». Se puede ver el *burdown chart* aquí [A.18](#).

### A.3. Estudio de viabilidad

En este apartado se realiza un estudio de viabilidad del proyecto que consiste en utilizar la técnica del análisis coste/beneficio. Con ello se obtiene una valoración de la justificación económica del proyecto.

El estudio de la viabilidad se divide en la viabilidad económica y la viabilidad legal. Para el estudio de viabilidad económica se supone que el proyecto ha sido realizado dentro del ámbito empresarial. Quiere decir que se considera que los alumnos han recibido un sueldo por en base a la legislación por realizarlo.

#### Viabilidad económica

Para analizar la viabilidad económica, hay que tener en cuenta los costes de personal y los costes de material.

#### Costes de personal

Se cuenta como coste personal el trabajo del alumno, considerado como un programador junior. La duración del trabajo a sido de 5 meses, unos con más carga de trabajo que otros. Estimo unas 460 horas de trabajo y analizando un poco los sueldo de los programadores junior en Java que ronda los 9 euros la hora, el resultado es el siguiente.

$$460h * 9\text{€}/h = 4140\text{€}$$

A esto, debemos sumarle la Seguridad Social y otros impuestos que se deben pagar por empleado. En la actualidad, los costes a cargo de una empresa son los siguientes:

- Contingencias Comunes: 23,60 %
- Desempleo de Tipo General: 5,50 %
- Fondo de Garantía Social: 0,2 %
- Formación profesional: 0,6 %
- **Total:** 29.9 %

Costes	Importe €
Personal	4140
Seguridad Social	1237,86
Material	126,66
<b>Total</b>	<b>5504,52</b>

Tabla A.1: Tabla de los costes totales

Que esto equivale a un gasto por un trabajador de:

$$4140\text{€} * 0,299 = 1237,86\text{€}$$

### Costes de material

Como costes materiales se tienen en cuenta el software y el *hardware*. El sistema operativo es Windows versión 8.1 Home, que tiene un precio de 45€. A parte de esto, el software es completamente gratuito y no necesita ninguna licencia bajo pago. La plataforma *Google Cloud*, es gratis durante el primer año y la licencia caduca el 26 de junio de 2018. Además de regalo tengo 300 dólares a gastar en dos meses de servicios. En cuanto al *hardware* tendremos en cuenta el equipo en el cual se ha desarrollado. Se trata de un computador con un valor de 980€, que teniendo en cuenta su amortización media de unos 5 años de uso, el cálculo sería el siguiente.

$$\frac{980\text{€}}{12\text{meses} * 5\text{periodos}} * 5\text{meses de uso} = 81,66\text{€}$$

Por lo tanto los costes totales quedaría reflejados en la tabla A.1.

### Viabilidad legal

En el marco de la viabilidad legal, se debe considerar las licencias de los programas y librerías utilizados para la realización de la aplicación.

Para el desarrollo del proyecto se ha hecho uso del *framework* GWT, que cuenta con la Licencia Apache 2.0, que es una licencia software libre permissiva compatible con GPL y aprobada por la *Open Source Initiative*. Esto se traduce en que el software producido bajo los términos de esta licencia otorga a usuario la utilización del software para cualquier propósito de modificación o distribución.

Las librerías que se emplean también son gratuitas y de código abierto, como se puede observar en la siguiente tabla A.2.

Librería	Versión	Descripción	Licencia
App Engine SDK for Java	1.9.34	Conjunto de librerías para trabajar con la plataforma App Engine.	Apache License 2.0
GWT SDK	2.7.0	Contiene las librerías principales y el compilador que se necesitan para escribir aplicaciones web.	Apache License 2.0
JRE System Library	1.8.0-60	Contiene las librerías mínimas de Java para ejecutar la aplicación.	
JUnit	0.12.3	Framework que sirve para hacer pruebas de sistema.	Eclipse Public License 1.0
Google Plugin	3.9.6	Sirve para desarrollar utilizando GWT y despliega aplicaciones en el entorno de App Engine.	Eclipse Public License v 1.0
Google App Engine Maven Integration	3.9.6	Una herramienta de gestión de proyectos de software.	Eclipse Public License 1.0
ObjectAid Class Diagram	1.1.14	Herramienta de visualización de código y creación de diagramas para Eclipse.	Eclipse Public License v 1.0

Tabla A.2: Tabla de software y sus licencias

## *Apéndice B*

---

# Especificación de Requisitos

---

## B.1. Introducción

En esta sección se definen los requisitos de la aplicación así como los objetivos de una forma más esquemática y concreta que en otros apartados.

## B.2. Objetivos generales

Estos han sido los principales objetivos del proyecto según la línea de tiempo. Primero creamos un prototipo cliente-servidor para ir conociendo el funcionamiento y una vez entendido, pasar a construir la aplicación con la definición de la gramática. En la construcción de la aplicación se necesita haber entendido el prototipo porque ya ahí se aplica la funcionalidad del cliente y del servidor. Más tarde se van incluyendo los algoritmos sobre las gramáticas formales. Por último, una vez se ha construido la parte esencial de la aplicación podemos centrarnos en crear un registro e inicio de sesión de usuarios.

### Llevar Thoth a la web

El objetivo del proyecto es básicamente llevar a Thoth a la web, manteniendo las mismas funcionalidades y conservando su forma. Se pretende mantener el núcleo intacto, ya que es el corazón de Thoth. Partiendo de él, debemos formar todo el aspecto visual, el modelo de vistas en pestañas y las funcionalidades.

Por eso se decide utilizar una herramienta que haga la conversión de forma directa y que el resultado sea, en cuanto a funcionalidad, el mismo que el de la aplicación original.

### Buscar una herramienta adecuada

Es básico elegir una herramienta adecuada para traducir la aplicación. De ello dependerá todo el proyecto y nos obligará a hacerlo de una manera u otra.

Partiendo de aquí, se podrá ver las posibilidades de mejora. Si la traducción es simple, tendremos tiempo para realizar todas las mejoras en mente, sino, gran parte del proyecto consistirá en traducir.

### Construir la aplicación con la definición de gramática

Consiste en hacer ya el comienzo de la aplicación empezando por la «definición de gramática» que es la parte de Thoth en la que se introduce una gramática en un campo de texto y el programa es capaz de identificarla y mostrar sus propiedades. Para ello es necesario haber logrado en objetivo anteriormente descrito, ya que se necesita alojar en el servidor el *parser* de la gramática y hacerle peticiones desde el cliente para interpretarla y mostrar sus propiedades.

Es el comienzo de la aplicación y la base sobre la que trabajar posteriormente, ya que para implementar los algoritmos es necesario tener definida una gramática.

### Construir los algoritmos sobre gramáticas

El objetivo es implementar cada uno de los algoritmos que se aplican sobre una gramática en Thoth. Es una de las utilidades más importantes de esta aplicación y la cual otorga mas posibilidades a los usuarios. Esta compuesta por los algoritmos siguientes:

- Eliminación de símbolos no terminales.
- Eliminación de símbolos no alcanzables.
- Eliminación de símbolos anulables.
- Eliminación de producciones no generativas.
- Eliminación de recursividad directa.
- Eliminación de recursividad indirecta.
- Factorización por la izquierda.
- Forma normal de Chomsky.
- Análisis descendente LL(K) formado por el cálculo de *First* y *Follow* y reconocimiento por medio de TASP o Tabla de Análisis Sintáctico Predictivo.

Hay además dos funcionalidades que engloban varios de estos algoritmos como son la de limpiar gramática y eliminar recursividad.

Varios de ellos tienen la particularidad de que su ejecución esta hecha paso a paso, señalando dichos pasos con un resaltado con colores.

### Mantener un registro de usuarios

Como la prioridad principal es llevar Thoth a la web y que funcionase correctamente, hacer el registro de usuarios es el objetivo que se encuentra en la última posición. Forma parte de las mejoras con respecto a las anteriores versiones de Thoth, ya que ninguna disponía de esta funcionalidad. Aprovechando la ventaja de que GWT trabaja con comunicación cliente-servidor, decidimos hacer una base de datos en la que registramos los usuarios de la aplicación así como parte de su actividad en Thoth.

Con ello se podrá saber el uso que se le da al proyecto o si los usuarios son activos mostrando, por ejemplo, la hora del último acceso de un usuario o si es utilizado por muchas personas, entre otras ideas. Se debe comprobar que los datos introducidos en el momento de registro, cumplen una serie de requisitos. Además se pretende hacer inicio de sesión para que en caso de que salga de la aplicación, no necesite volver a iniciar el proceso de *login*. De esta forma queremos que el usuario se sienta cómodo al utilizar Thoth.

## B.3. Catálogo de requisitos

El proceso de realización del proyecto demanda los siguientes requisitos:

- Crear un prototipo de aplicación cliente-servidor con GWT
  - Construir la parte del cliente.
  - Construir la parte del servidor.
  - Realizar comunicación entre ambas vía RPC.
- Construir la aplicación con la definición de gramática
  - Construir una vista principal para editar un texto.
  - Realizar peticiones al servidor.
  - *Parsear* el texto y convertirlo en una gramática.
  - Devolver la gramática al cliente.
  - Trabajar con la gramática en el lado del cliente.
- Construir los algoritmos sobre gramáticas

- Construir cada una de las vistas de los algoritmos.
  - Trabajar con la gramática recibida y realizar el algoritmo.
  - Resaltar los pasos realizados por el algoritmo gracias al manejo del texto por medio del lenguaje HTML.
  - Devolver el resultado a la vista principal en caso de éxito.
- Construir un registro de usuarios
- Construir un módulo nuevo.
  - Establecer el cambio entre módulos de forma correcta.
  - Crear una clase en el servidor que se comunique con el módulo y el cliente.
  - Registrar información en una base de datos.
  - Comprobar identificación de un usuario.
  - Manejar la información del usuario.
  - Establecer una sesión.

## B.4. Especificación de requisitos

A continuación, en este apartado se enumeran los casos de uso con una breve descripción de cada uno de ellos.

- Registro de usuario: Registra los datos de un nuevo usuario.
- Inicio de sesión: Permite el acceso de un usuario a la aplicación.
- Cierre de sesión: Aborta la sesión actual del usuario.
- Crear gramática: Posibilita la creación de una gramática.
- Modificar gramática: Sirve para realizar cambios sobre la gramática.
- Algoritmos: Aporta los algoritmos propios de la gramática.
- Idioma: Cambia el idioma de los textos de la aplicación.
- Ayuda: Muestra información sobre la aplicación.
- Renombrar símbolo: Busca un símbolo para después reemplazarlo por otro en una gramática.
- Comprobar gramática: Comprueba una gramática según una sintaxis definida.

- Eliminar símbolos no terminales: Suprime los símbolos no terminables de una gramática.
- Eliminar símbolos no alcanzables: Suprime los símbolos no alcanzables de una gramática.
- Eliminar símbolos anulables: Quita los símbolos que son anulables de una gramática.
- Eliminar producciones no generativas: Quita las producciones no generativas de una gramática.
- Limpiar gramática: Por orden realiza los siguientes algoritmos:
  - Eliminar símbolos anulables.
  - Eliminar símbolos no terminables.
  - Eliminar símbolos no alcanzables.
  - Eliminar producciones no generativas.
- Eliminar recursividad directa: Suprime la recursividad directa de todas las producciones recursivas.
- Eliminar recursividad indirecta: Suprime la recursividad indirecta de todas las producciones.
- Eliminar recursividad: sigue los siguientes pasos
  - Eliminar recursividad directa.
  - Eliminar recursividad indirecta.
- Factorizar por la izquierda: Consiste en obtener el prefijo común del consecuente en todas las producciones que tengan el mismo antecedente, agrupar esas gramáticas al identificar su prefijo común.
- Forma normal de Chomsky: consiste en transformar la gramática de forma que en el consecuente solo aparezca o bien un terminal o bien dos no terminales.
- Calculo Firsts Follow: Genera las tablas de First y Follow de una gramática.
- Reconocimiento TASP: Genera la Tabla de Análisis Sintático Predictivo de una gramática y permite comprobar si una palabra es reconocida o no por la gramática

### Diagrama de casos de uso

Como hay bastantes casos de uso, se utilizarán varios diagramas para documentarlo. Primero se muestra un diagrama del caso de uso general de la aplicación B.1. Posteriormente el diagrama B.2 muestra los casos de uso de crear y modificar una gramática.

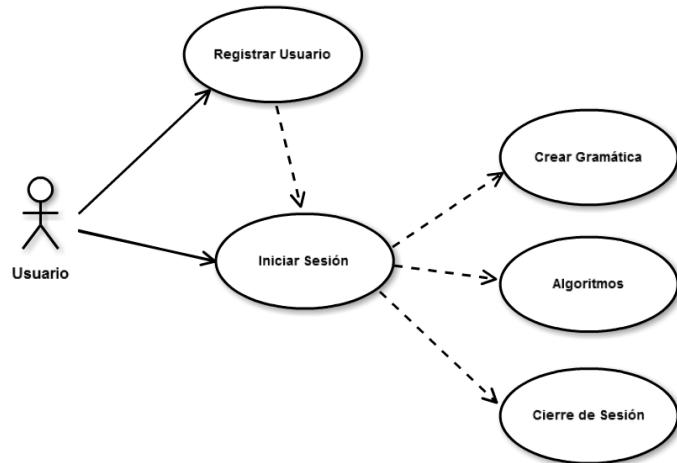


Figura B.1: Diagrama de caso de uso general.

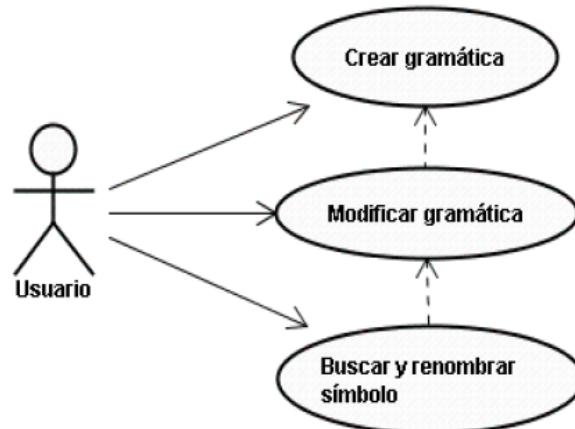


Figura B.2: Diagrama de caso de uso Crear/Modificar Gramática[1].

<b>Caso de uso 1</b>	Registrar un nuevo usuario.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Un usuario puede registrarse en la aplicación rellenando los datos que sean necesarios
<b>Precondiciones</b>	
<b>Acciones</b>	
1 Ejecutar la aplicación. 2 Pulsar botón de registro 3 Rellenar los campos requeridos. 4 Pulsar en registro.	
<b>Postcondiciones</b>	La información de usuario deberán guardarse en la base de datos.
<b>Excepciones</b>	El campo email es incorrecto. Alguno de los campos no se ha completado
<b>Frecuencia</b>	Alta
<b>Importancia</b>	Alta

Tabla B.1: Caso de uso Registro.

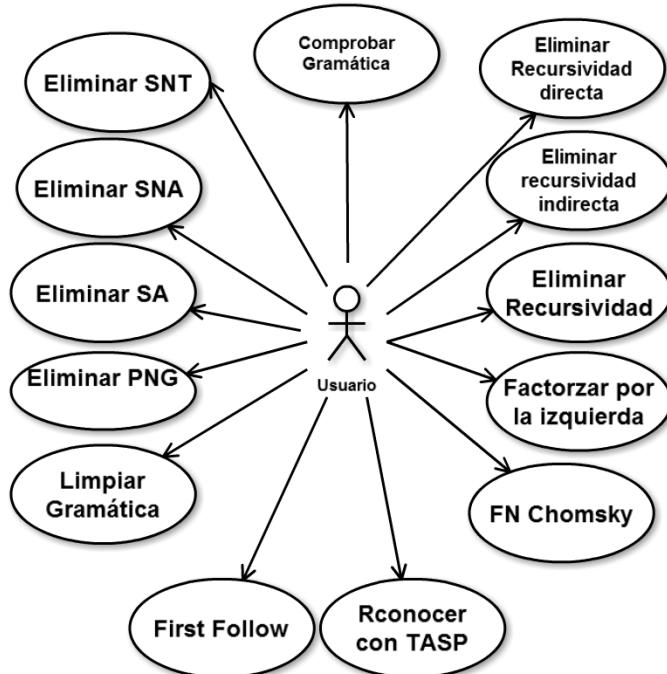


Figura B.3: Diagrama de caso de uso Algoritmos.

<b>Caso de uso 2</b>	Inicio de sesión de un usuario.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	El usuario introduce sus datos de correo electrónico y la contraseña correspondiente para acceder.
<b>Precondiciones</b>	El usuario debe tener una cuenta registrada.
<b>Acciones</b>	1 Inserta la dirección de correo. 2 Inserta la contraseña asociada al correo
<b>Postcondiciones</b>	Se redirigirá a la vista principal de la aplicación donde se encuentra el menú. Podrá acceder a la aplicación por un determinado tiempo sin volver a iniciar sesión.
<b>Excepciones</b>	El campo email es incorrecto. Alguno de los campos no se ha completado El email no está registrado La contraseña asociada al email es incorrecta
<b>Frecuencia</b>	Alta
<b>Importancia</b>	Alta

Tabla B.2: Caso de uso Inicio de sesión.

<b>Caso de uso 3</b>	Cierre de sesión.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Cierre de la sesión actual del usuario.
<b>Precondiciones</b>	Se debe tener una sesión abierta.
<b>Acciones</b>	1 Pulsar en el nombre del usuario con la sesión abierta.
<b>Postcondiciones</b>	Volverá a la pantalla de inicio de sesión. Para volver a acceder a la aplicación deberá volver a iniciar sesión.
<b>Excepciones</b>	Fallo en la sesión. Usuario nulo.
<b>Frecuencia</b>	Alta
<b>Importancia</b>	Alta

Tabla B.3: Caso de uso Cierre de sesión.

<b>Caso de uso 4</b>	Crear gramática.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Introducir una gramática Los campos de características de la gramática están vacíos.
<b>Precondiciones</b>	Se debe tener una sesión abierta.
<b>Acciones</b>	1 Acceder al menú principal. 2 Introducir una gramática 3 Comprobar dicha gramática.
<b>Postcondiciones</b>	La comprobación ha sido correcta y no ha emitido ningún mensaje de error. La tabla características se encuentra completada con los datos.
<b>Excepciones</b>	Gramática con sintaxis incorrecta.
<b>Frecuencia</b>	Alta
<b>Importancia</b>	Alta

Tabla B.4: Caso de uso Crear gramática.

<b>Caso de uso 5</b>	Modificar gramática.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Consiste en realizar cambios sobre una gramática ya escrita en el panel.
<b>Precondiciones</b>	Se debe tener una sesión abierta. Debe haber introducido una gramática.
<b>Acciones</b>	1 Introducir una nueva gramática 3 Comprobar/checkear dicha gramática.
<b>Postcondiciones</b>	La comprobación ha sido correcta y no ha emitido ningún mensaje de error. La tabla características se encuentra completada con los datos.
<b>Excepciones</b>	Gramática con sintaxis incorrecta.
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.5: Caso de uso Modificar gramática.

<b>Caso de uso 6</b>	Buscar y renombrar símbolo.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Una vez se tenga una gramática sustituir algún símbolo de ella por uno nuevo.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.        Debe haber introducido una gramática.</p> <p>1 Si la gramática esta comprobada pasa a 4.        2 Si la gramática no esta comprobada pasa a 3.        3 Comprobar/checkear dicha gramática.        4 Busca el símbolo introducido        5 Sustituye el símbolo por otro nuevo introducido por el usuario.</p>
<b>Acciones</b>	Aparece una gramática nueva con las sustituciones.
<b>Postcondiciones</b>	Gramática con sintaxis incorrecta. El usuario no pulsa en aceptar El símbolo a sustituir no existe.
<b>Excepciones</b>	Baja
<b>Frecuencia</b>	Alta

Tabla B.6: Caso de uso Buscar y renombrar símbolo.

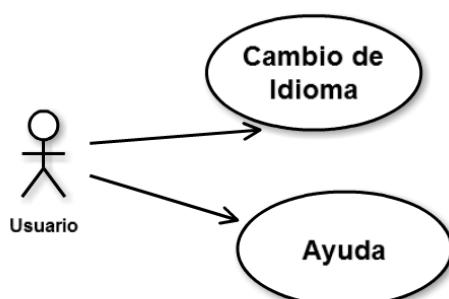


Figura B.4: Diagrama de caso de uso Opciones.

<b>Caso de uso 7</b>	Realizar algoritmos.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se limpia la ventana y aparece una vista con el algoritmo.  Se debe tener una sesión abierta.
<b>Precondiciones</b>	Debe haber creado una gramática. Haber comprobado la gramática.
<b>Acciones</b>	1 El usuario pincha en el menú en algoritmos. 2 Aparece un desplegable con todos los algoritmos.
<b>Postcondiciones</b>	Aparece una nueva vista con los paneles del algoritmo.
<b>Excepciones</b>	Gramática con sintaxis incorrecta. El usuario no ha comprobado la gramática
<b>Frecuencia</b>	Alta
<b>Importancia</b>	Alta

Tabla B.7: Caso de uso de algoritmos.

<b>Caso de uso 8</b>	Eliminar símbolos no terminables.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se eliminan aquellos símbolos que no sean terminables, que sean de paso, de la gramática.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona eliminar símbolos no terminables.</li> <li>2. Se crea una vista de la gramática y de la nueva gramática en la que se van a realizar los pasos del algoritmo eliminación de símbolos no terminables.</li> <li>3. El usuario puede: <ul style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ul> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece una nueva vista en el menú con la gramática resultante.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.8: Caso de uso de Eliminar símbolos no terminables.

<b>Caso de uso 9</b>	Eliminar símbolos no alcanzables.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se eliminan aquellos símbolos que no sean alcanzables
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona eliminar símbolos no alcanzables.</li> <li>2. Se crea una vista de la gramática y de la nueva gramática en la que se van a realizar los pasos del algoritmo eliminación de símbolos no alcanzables.</li> <li>3. El usuario puede:           <ol style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ol> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece una nueva vista en el menú con la gramática resultante.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.9: Caso de uso de Eliminar símbolos no alcanzables.

<b>Caso de uso 10</b>	Eliminar símbolos anulables.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	<p>Se Eliminan aquellos símbolos que sean terminables</p> <p>Se debe tener una sesión abierta.</p>
<b>Precondiciones</b>	<p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona eliminar símbolos anulables.</li> <li>2. Se comprueba que haya producciones épsilon.</li> <li>2.1.Si no hay producciones épsilon, se abre un warning.</li> <li>2.2.Si hay, abre una ventana con la vista de la gramática y de la nueva gramática y en la que se van a realizar los pasos del algoritmo eliminación de símbolos no terminables.</li> <li>3 El usuario puede <ul style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ul> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece una nueva vista en el menú con la gramática resultante.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar</p> <p>La gramática no tiene producciones épsilon.</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.10: Caso de uso de Eliminar símbolos anulables.

<b>Caso de uso 11</b>	Eliminar producciones no generativas.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se eliminan aquellas producciones que no sean generativas.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona eliminar producciones no generativas.</li> <li>2. Se abre una vista con la gramática y la nueva gramática y en la que se van a realizar los pasos del algoritmo</li> <li>3. El usuario puede: <ul style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ul> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece una nueva vista en el menú con la gramática resultante.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.11: Caso de uso de Eliminar producciones no generativas.

<b>Caso de uso 12</b>	Limpiar gramática.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se eliminan: símbolos no terminables, no alcanzables, anulables y producciones no generativas.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<p>1. El usuario selecciona limpiar gramática</p> <p>2.1. Si no es necesario limpiar la gramática se indicará en un mensaje de error</p> <p>2.2 Realiza la limpieza de forma normal</p>
<b>Postcondiciones</b>	Aparece la gramática ya limpia.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>No es necesario limpiar la gramática</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.12: Caso de uso de Limpiar gramática.

<b>Caso de uso 13</b>	Eliminar recursividad directa.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se encarga de la eliminación de la recursividad directa de una gramática
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona eliminar recursividad directa</li> <li>2. Se abre una vista de la gramática y de la nueva gramática y en la que se van a realizar los pasos del algoritmo de eliminación de recursividad directa:</li> <li>3. El usuario puede: <ul style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ul> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece la gramática sin recursividad directa.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar.</p> <p>La gramática no tiene recursividad directa</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.13: Caso de uso de Eliminar recursividad directa.

<b>Caso de uso 14</b>	Eliminar recursividad indirecta.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se encarga de la eliminación de la recursividad indirecta de una gramática
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona eliminar recursividad indirecta</li> <li>2. Se abre una vista de la gramática y de la nueva gramática y en la que se van a realizar los pasos del algoritmo de eliminación de recursividad indirecta:</li> <li>3. El usuario puede: <ul style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ul> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece gramática sin recursividad indirecta.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar.</p> <p>La gramática no tiene recursividad indirecta</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.14: Caso de uso de Eliminar recursividad indirecta.

<b>Caso de uso 15</b>	Eliminar recursividad.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Elimina la recursividad de la gramática, tanto la directa así como la indirecta.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona eliminar recursividad</li> <li>2. Se realiza los algoritmos:           <ol style="list-style-type: none"> <li>2.1. Eliminar recursividad directa.</li> <li>2.2. Eliminar recursividad indirecta</li> </ol> </li> <li>3.1. Si no hay recursividad directa ni indirecta aparece un mensaje de error.</li> <li>3.2. En el área de texto aparece la gramática sin recursividad.</li> </ol>
<b>Postcondiciones</b>	Aparece la gramática sin recursividad.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar.</p> <p>La gramática no tiene recursividad directa ni indirecta</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.15: Caso de uso de Eliminar recursividad.

<b>Caso de uso 16</b>	Factorizar por la izquierda.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se encarga de factorizar la gramática que se le pasa.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona factorizar por la izquierda</li> <li>2. Se abre una vista de la gramática y de la nueva gramática y en la que se van a realizar los pasos del algoritmo</li> <li>3. El usuario puede:           <ol style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ol> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece una vista con la gramática factorizada.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar.</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.16: Caso de uso de Factorizar por la izquierda.

<b>Caso de uso 17</b>	Forma normal de Chomsky.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Transforma la gramática a forma normal de Chomsky.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona forma normal de Chomsky</li> <li>2. Se abre una vista de la gramática y de la nueva gramática y en la que se van a realizar los pasos del algoritmo</li> <li>3. El usuario puede:           <ol style="list-style-type: none"> <li>3.1.Pulsar sobre siguiente paso hasta que se llegue al final del algoritmo.</li> <li>3.2.Pulsar sobre todos los pasos.</li> </ol> </li> <li>4. El usuario pulsa sobre aceptar.</li> </ol>
<b>Postcondiciones</b>	Aparece una vista con la nueva la gramática.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar.</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.17: Caso de uso de Forma normal de Chomsky.

<b>Caso de uso 18</b>	Calcular el First y el Follow.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Calcula el first y el follow de cada uno de los no terminales de la gramática.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
	<ol style="list-style-type: none"> <li>1. El usuario selecciona cálculo de First y Follow.</li> <li>2. Se abre una ventana emergente que pregunta si se desea continuar.</li> <li>3. Si el usuario pulsa en «si» se abre una vista de la gramática y de la nueva gramática y en la que se van a realizar los pasos del algoritmo</li> <li>4. El usuario puede:</li> </ol>
<b>Acciones</b>	<p>a) Pulsar sobre el botón de first El sistema muestra el calculo de first.</p> <p>b) Pulsar sobre el botón de follow El sistema muestra el follow.</p> <p>c) Pulsar sobre el botón de TASP El sistema realiza el análisis descendente por reconocimiento con TASP.</p>
<b>Postcondiciones</b>	Aparece una vista con la nueva la gramática.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar.</p> <p>El usuario pulsa «no».</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.18: Caso de uso de First y follow.

<b>Caso de uso 19</b>	Reconocimiento con TASP.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Introducir una palabra para que sea reconocida por el autómata de pila que representa la TASP.
<b>Precondiciones</b>	<p>Se debe tener una sesión abierta.</p> <p>Debe haber creado una gramática.</p> <p>Haber comprobado la gramática.</p>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona reconocimiento con TASP.</li> <li>2. Se abre una ventana emergente que pregunta si se desea continuar.</li> <li>3. Si el usuario pulsa en «si» se abre una vista de la gramática y de la nueva gramática y en la que se van a realizar los pasos del algoritmo</li> <li>4. El usuario puede Introducir una palabra y: <ul style="list-style-type: none"> <li>a) Pulsar sobre siguiente paso hasta que se llegue al final del análisis.</li> <li>b) Pulsar sobre todos los pasos para que lo haga directamente</li> <li>c) Pulsar en borrar palabra y volver a paso 4.</li> </ul> </li> <li>5. Se mostrará si la palabra ha sido reconocida o no.</li> </ol>
<b>Postcondiciones</b>	Mensaje de palabra reconocida o no reconocida.
<b>Excepciones</b>	<p>Gramática con sintaxis incorrecta.</p> <p>El usuario pulsa cancelar.</p> <p>El usuario pulsa «no».</p>
<b>Frecuencia</b>	Media
<b>Importancia</b>	Alta

Tabla B.19: Caso de uso de Reconocimiento con TASP.

<b>Caso de uso 20</b>	Cambiar idioma.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Elegir el idioma en el que se quiere mostrar los textos de la aplicación
<b>Precondiciones</b>	Se debe tener una sesión abierta.
	<ol style="list-style-type: none"> <li>1. El usuario selecciona idioma en el menú.</li> <li>2. El usuario elige el idioma</li> <li>3. Se abre una ventana emergente que pregunta si se desea continuar.</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>3.1. Si el usuario pulsa en «si» se reinicia la vista</li> <li>3.2. El usuario pulsa «no» la ventana emergente desaparece sin hacer nada.</li> </ol>
<b>Postcondiciones</b>	Los textos se muestran en el idioma elegido.
<b>Excepciones</b>	El usuario pulsa «no» en la ventana emergente.
<b>Frecuencia</b>	Baja
<b>Importancia</b>	Alta

Tabla B.20: Caso de uso de Cambio de idioma.

<b>Caso de uso 21</b>	Mostrar Acerca de.
<b>Versión</b>	1.0
<b>Implica a</b>	Usuario
<b>Descripción</b>	Se muestra información detallada de la aplicación.
<b>Precondiciones</b>	Se debe tener una sesión abierta.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona ayuda en el menú.</li> <li>2. El usuario elige la opción Acerca de.</li> </ol>
<b>Postcondiciones</b>	Se muestra información sobre la aplicación.
<b>Excepciones</b>	
<b>Frecuencia</b>	Baja
<b>Importancia</b>	Alta

Tabla B.21: Caso de uso de Mostrar Acerca de.

## *Apéndice C*

---

# Especificación de diseño

---

## C.1. Introducción

En esta sección se habla sobre la fase de desarrollo software en la que definen la arquitectura, los procedimientos o los datos que se utilizan. Para hacerlo utilizaremos de diagramas que nos ayuden.

## C.2. Diseño de datos

### Gramática

Lo referente a la gramática me baso en el trabajo realizado en las anteriores versiones de Thoth [1] y obtengo la parte del núcleo. Según el diagrama obtenido de la documentación del proyecto, la clase Grammar esta compuesta de *Production* y esta a su vez de *Symbol*. Por medio de la interfaz *TypeHandler* se definen los tipos de gramáticas que existen C.1.

### Símbolos

Los símbolos son la unidad mínima con la que se trabajará en una gramática. Los símbolos en una gramática pueden ser terminales, no terminales. La clase NonTerminal estará formada por una cadena de terminales y la clase Terminal trabajará con un único carácter. También se definen los caracteres especiales como son épsilon, o el fin de cadena (TerminalEnd) C.2.

La especificación de los caracteres que pertenecen a cada tipo viene impuesta por la sintaxis utilizada. Se decidió utilizar minúsculas, dígitos y los operadores básicos ( $*$ ,  $+$ ,  $-$ ,  $\cdot$ ) para los terminales. Para los no terminales se determinó que la primera letra debía ir en mayúscula y a esta le podía seguir cualquier otro terminal o no terminal.

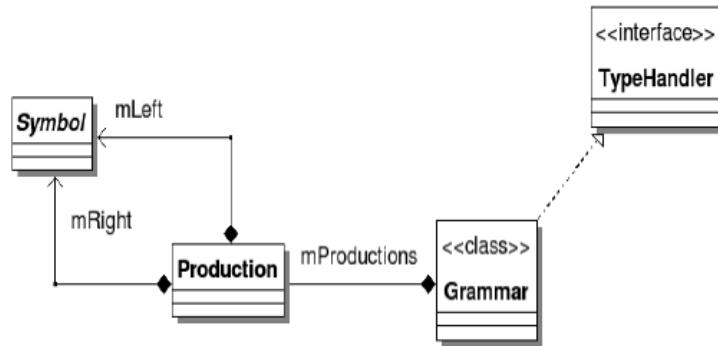


Figura C.1: Diagrama de clases del núcleo de la gramática[1].

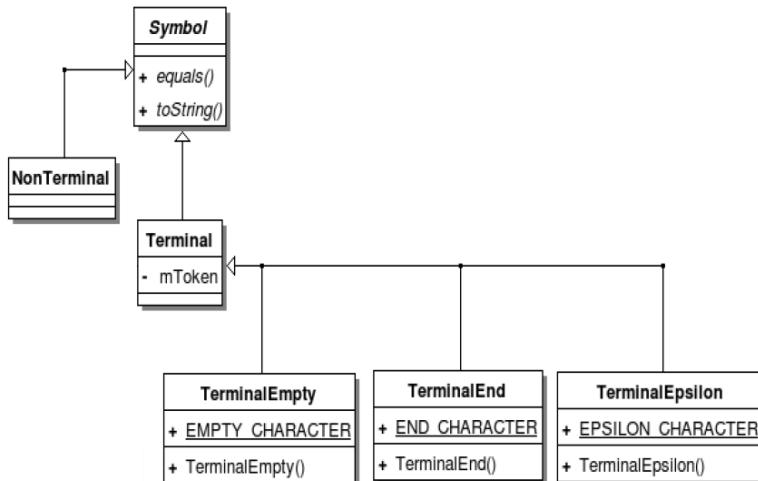


Figura C.2: Diagrama de clases de la estructura de Símbolo.

### Algoritmos de la gramática

Por simplificación se decide juntar todos los algoritmos de limpieza de gramáticas en un mismo caso dentro del diagrama. Además su estructura es similar entre ellos.

En la figura C.3 se aprecia como los algoritmos de limpieza tienen dos atributos principales, uno correspondiente a la gramática que se desea limpiar, definida como «**mOldGrammar**» y otro que representara la gramática resultante, «**mNewGrammar**».

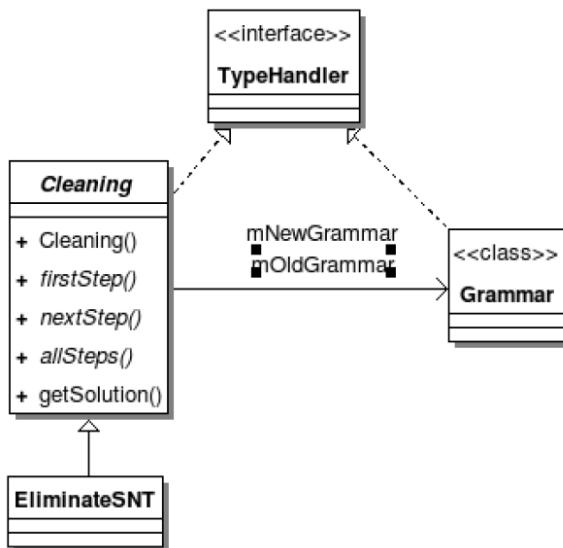


Figura C.3: Diagrama de clases de los Algoritmos de la gramática [1].

### Parser de la gramática

Aquí se define el diagrama mediante el cual el *parser* analiza la construcción de una gramática, en caso de estar correctamente construida nos crea la estructura interna propia, en caso de contener algún fallo nos muestra un error con el primer símbolo incorrecto encontrado. A diferencia del parser de expresión regular aquí no se crea ningún árbol de derivación, en su lugar se va analizando texto plano para ir creando la estructura interna propia. Esta forma de construcción de gramáticas se ha realizado con el fin de no tener que construir una gramática cada nueva producción, solo se creará la gramática cuando queramos utilizarla (instanciación bajo demanda).

### Base de datos

Este es el diseño de la base de datos para gestionar las sesiones y los registros de usuarios. La clase **Login** hace de nexo de unión entre las dos vistas, la de registro y la de inicio de sesión. Esa misma clase realiza las peticiones al servidor por medio de las interfaces **«Registration»**. Las interfaces hacen peticiones al *servlet* de registro, **«RegServlet»**, que realizará las operaciones debidas con la base de datos **«dataStoreService»** y con la clase de inicio de sesión **«sessionServlet»** que cargará el módulo que sea según la sesión.

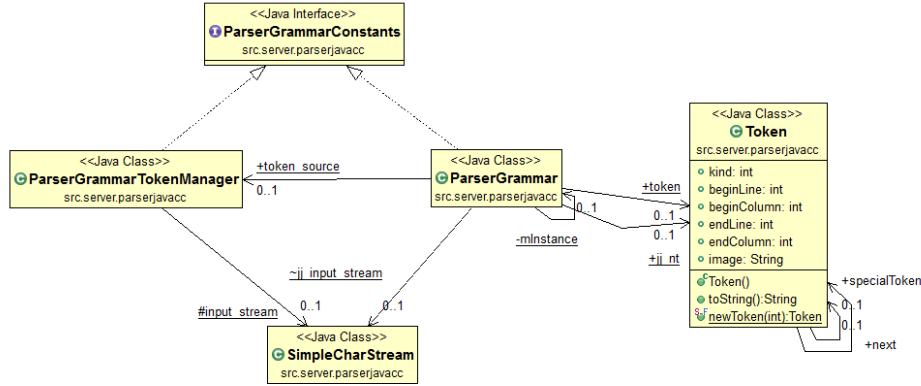


Figura C.4: Diagrama de clases del Parser de la gramática [1].

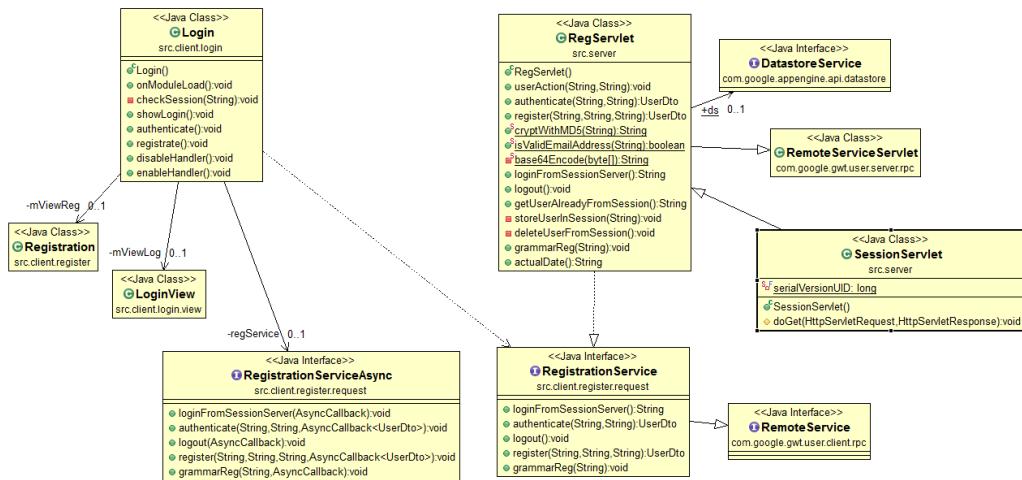


Figura C.5: Diagrama de clases del Inicio de sesión junto con el registro. Ambos se comunican con la base de datos.

### C.3. Diseño procedimental

Vamos a ver ahora el funcionamiento de varios de los procesos que se llevan a cabo en esta aplicación.

Los procesos más importantes en esta aplicación son los dos que se muestran en los siguientes diagramas de flujo. No se representan todos pero si aquellos más importantes que se relacionan con el servidor: ver figuras C.6 y C.7.

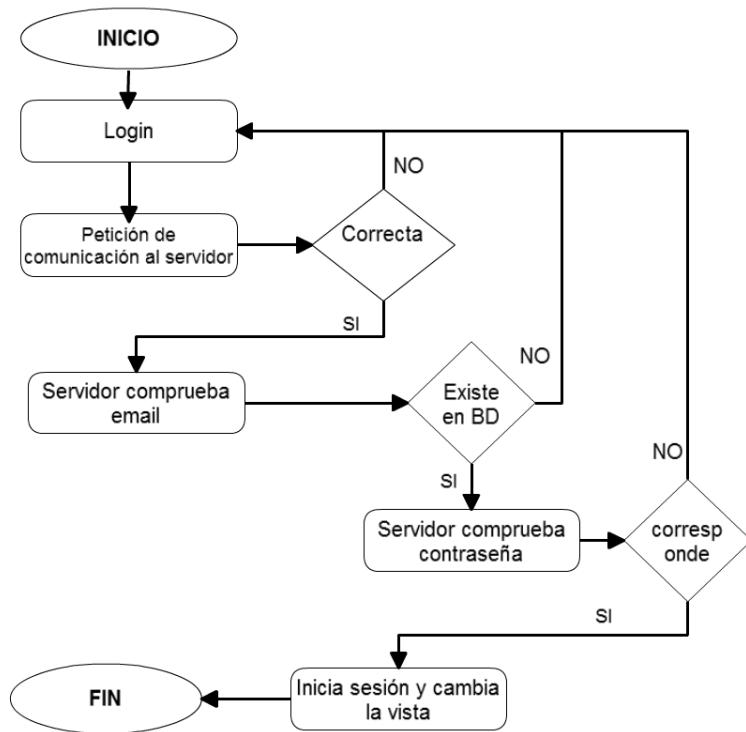


Figura C.6: Proceso de inicio de sesión en la aplicación.

#### C.4. Diseño arquitectónico

En cuanto al diseño arquitectónico, se va a explicar y mostrar de forma general como está diseñado el proyecto, en cuanto a la distribución de paquetes y los patrones de diseño que se han aplicado.

La aplicación se estructura en tres pilares fundamentales.

- src: Este paquete se encuentra todo el código escrito en Java. Esta estructurado en diversas partes siguiendo el diseño modelo-vista-controlador.
  - *client*: Aquí se encuentran todos los archivos programados en Java que será traducidos a JavaScript. No pueden contener librerías que no estén especificadas para GWT. Contiene parte del núcleo de Thoth y toda la parte visual de la aplicación.
  - *server*: En este paquete se guardan las clases que realizan funciones que van a usar librerías no aptas por GWT. Se deben especificar los *servlets* que se utilizan aquí dentro del war. Incluye el parserJavaCC y la base de datos. Su comunicación con el cliente es vía RPC.

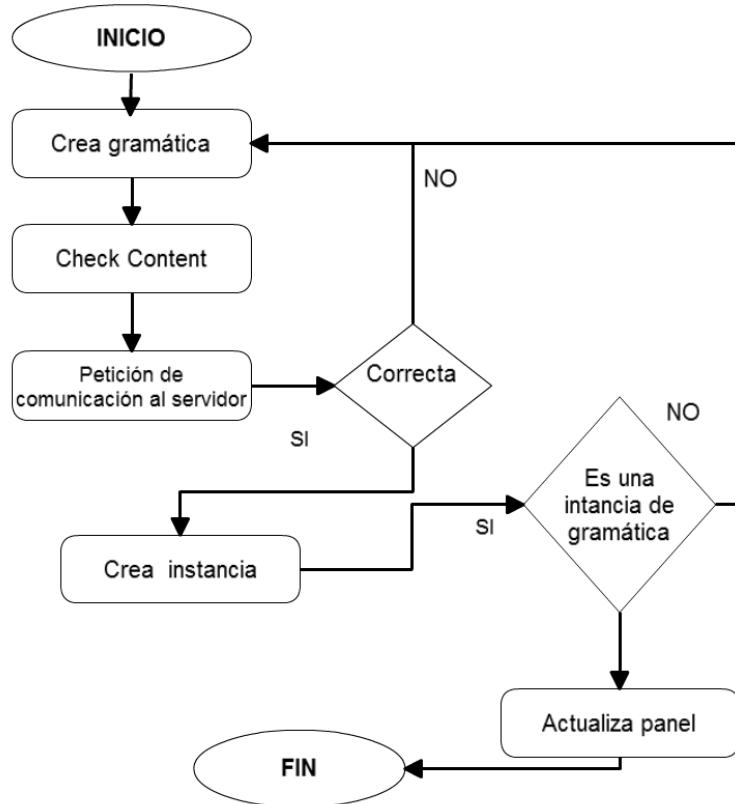


Figura C.7: Proceso de comprobación de una gramática.

- *shared*: En este paquete se alojan clases de ayuda entre el cliente y el servidor. Su comunicación es directa, sin RPC, pero tiene las mismas restricciones que el cliente.
- Test: Contiene las pruebas realizadas sobre diferentes funcionalidades de la aplicación.
- War: Son los archivos que se utilizarán para ejecutar la aplicación en el navegador. Muchos de ellos se generan al compilar la aplicación con GWT. Los html, css y algún xml, están programados por el desarrollador, para configurar la aplicación. GWT genera los archivos con extensión JavaScript, que es la traducción de Java.

El proyecto continúa con el diseño modelo-vista-controlador de las anteriores versiones de Thoth.

Es un patrón de arquitectura de las aplicaciones software. Separa la lógica de negocio de la interfaz de usuario, facilitando la evolu-

ción, por separado de ambos aspectos, y aumentando reutilización y flexibilidad [2].

La estructura de paquetes está definida de la siguiente forma en la ilustración C.8. Se puede ver como esta definido el modelo-vista-controlador y como esta agrupados los paquetes según la funcionalidad. Represento solo el cliente porque creo que es el interesante.

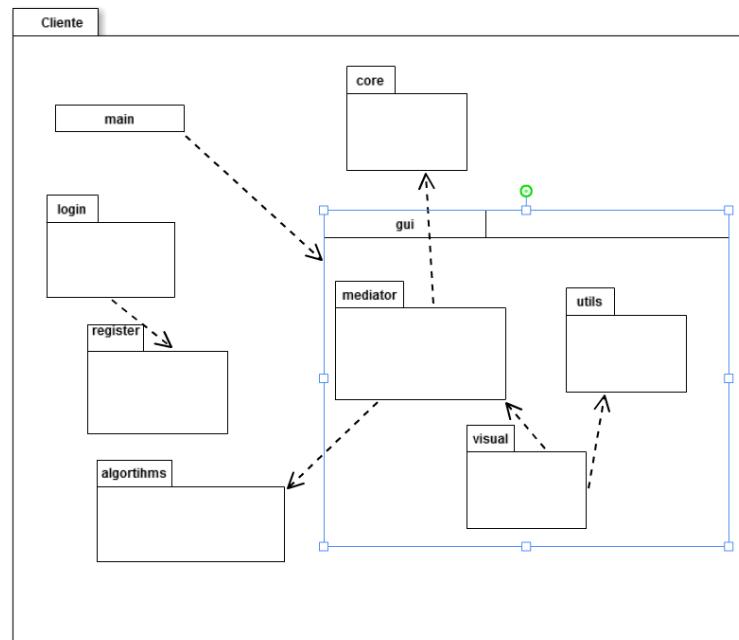


Figura C.8: Diagrama de paquetes dentro de Client.

## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

En esta sección se hablará de aspectos más técnicos pensando en futuros desarrollos. Sirve como ayuda para entender el funcionamiento de la aplicación y de los pasos que se han llevado a cabo para conseguir estos resultados. Se detalla la estructura del programa, de la instalación y configuración de las herramientas necesarias y del proyecto y, por último, de las pruebas realizadas.

## D.2. Estructura de directorios

### Aplicación

Dentro se encuentra la carpeta **Thoth-Web**. En este directorio se almacenan los fichero del código fuente del proyecto en src, el directorio war, y también los test.

- src
  - client
  - shared
  - server
  - módulo Login
  - módulo GramaticaCS
- test

- login
- war
  - imagenes
  - login
  - gramaticacs
  - WEB-INF

Client esta organizado de la siguiente manera:

- algorithms
- core
- gui
- login
- register
- main
- archivos de comunicación con el servidor

Server contiene los siguientes elementos:

- RegServlet.java
- GrammarServiceImp.java
- SessionServlet.java

## Prototipos

En este directorio, se encuentran los prototipos hechos al inicio del proyecto. Sirven para entender el funcionamiento de diferentes aspectos de GWT como, creación de un proyecto, comunicación cliente-servidor y dos pruebas con el núcleo de Thoth cliente-servidor.

- CorePrueba
- GWTProjectThree
- PrototipoCoreC-S
- StockWatcher

## Documentación

En este directorio se incluyen los la memoria y los anexos hechos con L<sup>A</sup>T<sub>E</sub>X, así como las imágenes para documentarlos.

### D.3. Manual del programador

#### Instalación y configuración de GWT:

Para poder trabajar con GWT <sup>1</sup>, se debe descargar el SDK de GWT proporcionado en la página web oficial. Los requisitos previos para crear un aplicación web con *Google Web Toolkit* son básicamente dos: tener instalada la SDK de Java en su versión 1.6, o cualquiera superior a esta, y tener instalado también Apache Ant o en su defecto Apache Maven.

En mi caso, como trabajo con la plataforma Eclipse (versión 4.4.) al descargar el plugin de Google para Eclipse, este viene con GWT y la *App Engine SDK*, todo junto. Se puede descargar desde la siguiente dirección <sup>2</sup>.

Para instalar el plugin desde eclipse, solo tenemos que ir a «help» y en el desplegable elegir la opción «Install new software». Aparecerá una ventana con un recuadro en el que pone «work with» y nos da la opción de escribir una dirección web. Para consultar las direcciones según la versión de Eclipse sólo tiene que visitar <https://developers.google.com/eclipse/docs/download>. Aparecerá los elementos que se pueden instalar, y ahí elegimos el *plugin* de Google. Debemos esperar a que se instale y ya solo hará falta reiniciar Eclipse.

Como último paso, se debe referenciar la SDK de GWT y completar la instalación. Para habrá que ir a la opción «window» de la pantalla principal, hacer *click* en preferencias y aparecerá una ventana con varias opciones. Buscamos la opción de Google y en *web toolkit* y ahí añadir la referencia a la SDK con la dirección en la que se encuentra el archivo. Lo mismo hay que hacer para el *App Engine* pero en este caso añadiendo la referencias al *App Engine*.

El modo de trabajar con GWT es muy simple. Desde el *plugin* podemos crear un nuevo proyecto dese «New Web Application Project», darle un nombre y marcando las opciones del SDK y el App Engine.

#### Parser JavaCC

GWT esta compuesto por una parte cliente, que es la que será traducida a JavaScript, y otra servidor escrita en Java y que no se traducirá. En un principio se prueba a incluir toda la aplicación dentro del lado del cliente, traduciendo todo a JavaScript. Pero como ya se ha dicho en otros apartados,

---

<sup>1</sup><http://www.oracle.com/technetwork/java/javase/downloads/index.html>

<sup>2</sup><http://www.gwtproject.org/usingeclipse.html>

GWT sólo soporta unas determinadas librerías en dicho lado y tiene algunas restricciones más<sup>3</sup> que obligan a incluir el parser de JavaCC en el servidor, en vez del cliente.

Originalmente en el núcleo de Thoth, dentro del directorio «grammar», colgaba un sub-directorio llamado «parserjavacc» en el cual se encontraba el *parser* para la gramática. Es la parte más importante a lo que a la gramática se refiere, ya que permite reconocer si una gramática está bien construida o no.

### Comunicación cliente-servidor

Una vez alojado en el servidor el *parser* habrá que establecer una comunicación entre la parte del cliente en la que habrá un método que llame a la funcionalidad que se desee del servidor.

Para hacerlo, GWT utiliza la técnica denominada por sus siglas en inglés RPC o llamada de procedimiento remoto (*Remote Procedure Call*) que tiene una estructura similar a la siguiente ilustración D.1.

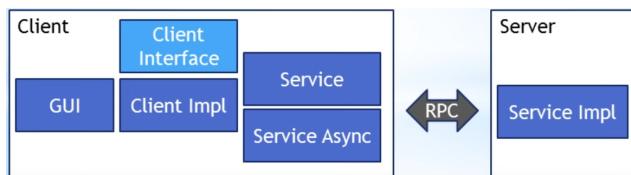


Figura D.1: Arquitectura RPC simple.

Del lado del cliente tenemos, en este caso, una interfaz gráfica de usuario (GUI), una clase dedicada a hacer las llamadas RPC (ClientImpl), dos interfaces que definen los métodos (*Service* y *ServiceAsync*), una clase con los métodos en el lado del servidor (serviceImpl) y por último una interfaz, que no es esencialmente necesaria (ClientInterface).

En términos generales, para utilizar el *parser* de la gramática, que es quizás la clave de todo, hay que llamar al servidor y establecer una comunicación. Es ahí donde utilizamos la clase «GrammarServiceClientImpl» que hace las llamadas a las interfaces necesarias hasta llegar al servidor. Esta clase es un nexo de unión entre la parte del núcleo, que ese encuentra en el servidor, y la parte del cliente, que contiene lo visual y las acciones de los botones entre otros. La comunicación queda representada en el diagrama D.2.

Dentro de la clase «GrammarServiceClientImpl» hay tres constructores según le pasemos una gramática, un texto, simplemente ningún parámetro. El

<sup>3</sup><http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsCompatibility.html>.

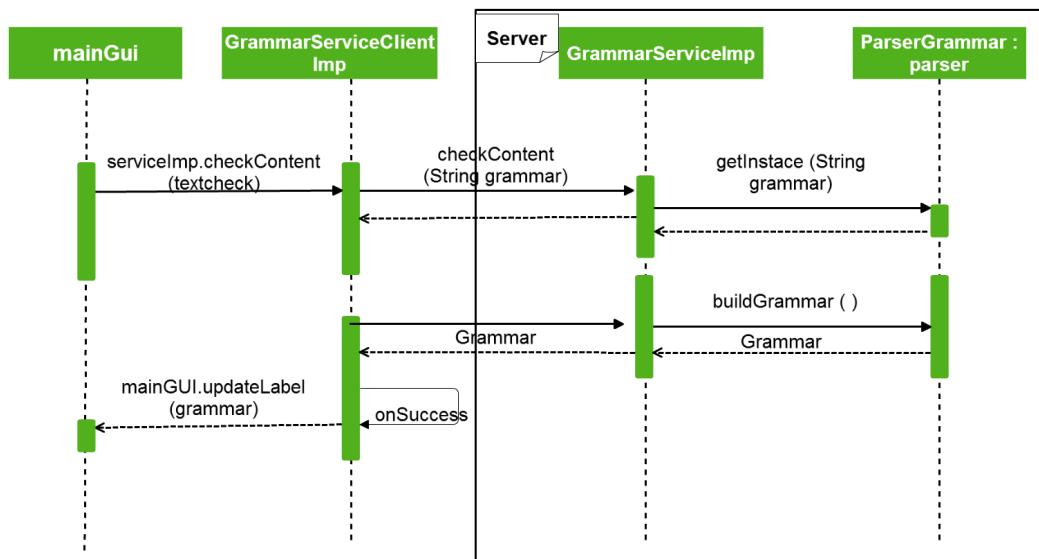


Figura D.2: Diagrama sobre la comunicación entre el cliente y el servidor en GWT.

atributo `url` es simplemente la localización del *servlet* que se utiliza para la comunicación RPC. Estos detalles se aprecian mejor en el diagrama de clases que muestra la imagen D.3.

Una vez hechos todos estos pasos, es posible que no funcione nada, es más, dará un error del tipo `did you forget to inherit a required module?`. Todos los errores parecidos a este tienen que ver con el archivo «`web.xml`» que se encuentra en todos los proyectos GWT dentro de la carpeta «`war`» en «`WEB-INF`». En ese archivo tenemos que definir los *servlets*, dándolos un nombre y especificando la dirección, dentro del proyecto, donde se encuentran. Después deberemos indicar la dirección del patrón, es decir indicar a qué módulo pertenecen y con qué nombre. Ese nombre se lo especificamos anteriormente en la interfaz «`GrammarService`» con el componente «`@RemoteServiceRelativePath("nombre")`».

## D.4. Compilación, instalación y ejecución del proyecto

Antes de empezar a hablar sobre compilación debemos contar con un proyecto en GWT, preferiblemente en Eclipse, habiendo instalado el *plugin* de Google, el SDK de *App Engine* para Java y el SDK de GWT.

Hace unos años Google contaba con un *plugin* para algunos navegadores (Chrome y Firefox, no confundir con el *plugin* de Eclipse) con los que se podía

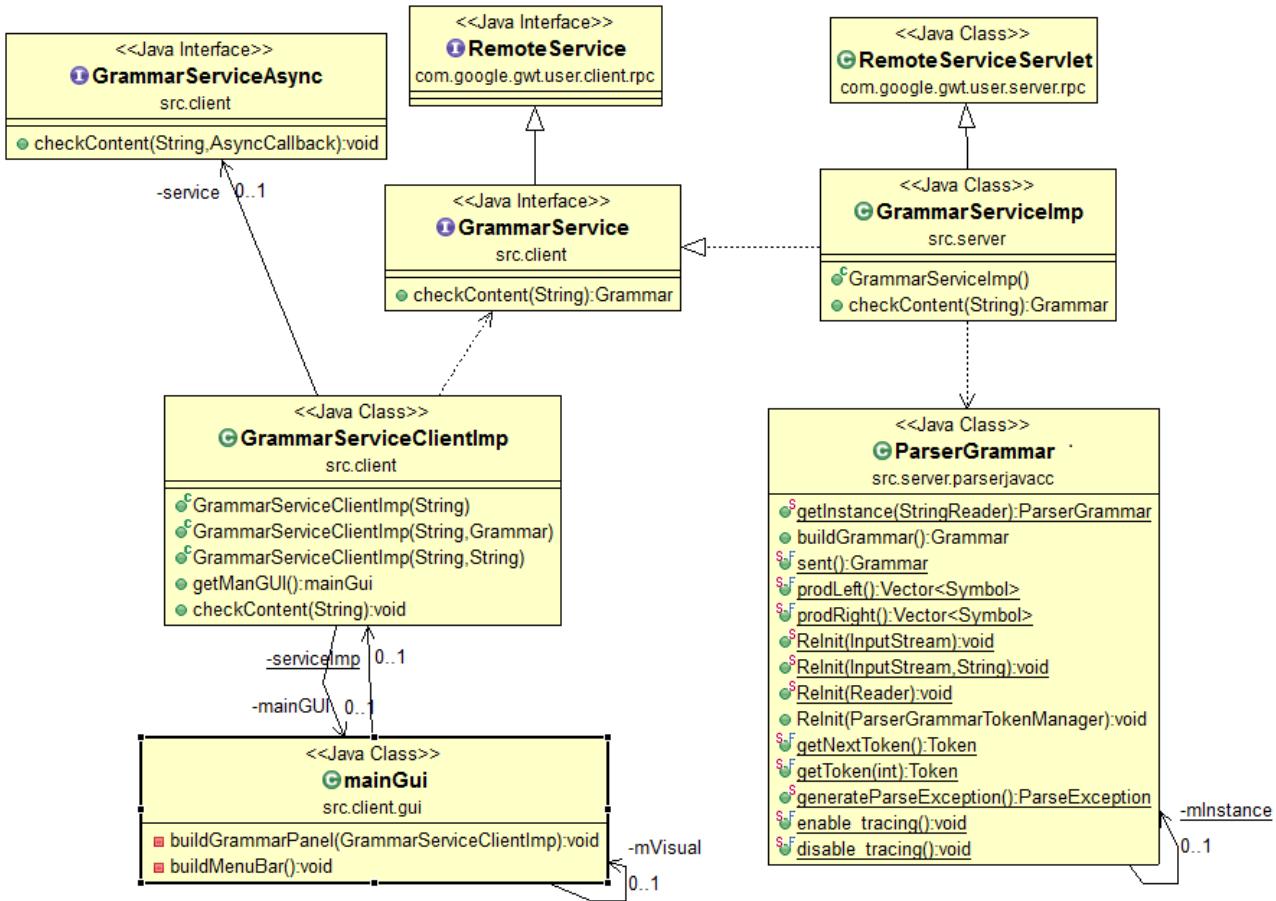


Figura D.3: Diagrama de clases que muestra como utiliza el parser de la gramática. Diagrama realizado con ObjectAid.

ejecutar el proyecto. Pero Google paró el mantenimiento. Afortunadamente los desarrolladores de GWT crearon *GWT Super-Dev Mode* que permite ejecutar cualquier proyecto realizado con esta herramienta.

En la definición del módulo, para poder realizar la ejecución con *Super-Dev Mode* debemos incluir la sentencia `add-linker name="xsiframe"`. Si no lo incluimos podemos incurrir en algún tipo de error al ejecutarlo. Las definiciones de los módulos se encuentran en la raíz de la carpeta principal y tienen una extensión «.gwt.xml». Es habitual que no se llegue a hacer la compilación porque no se han definido los *servlets* dentro del archivo «web.xml».

De esta forma una vez que se haya ejecutado en eclipse, se mostrará una dirección local que podemos abrir en nuestro navegador [D.4](#). Este compilara los archivos HTML, y los JavaScript. Este proceso tardará unos segundo y cuando



Figura D.4: Dirección local que aparece al ejecutar en modo SuperDevMode.

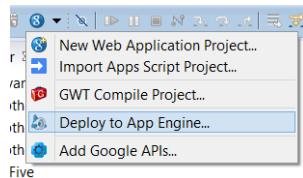


Figura D.5: Acción de desplegar la aplicación con App Engine.

termine se mostrará la aplicación.

Si tenemos una base de datos, para poder acceder a ella, hay que coger la raíz de la dirección local, en nuestro caso «<http://127.0.0.1:8888/>», y añadirle «`_ah/admin`». Desde ahí podemos ver las entidades que hay, y en cada una de ellas los datos almacenados.

## Despliegue

Por otro lado podemos desplegar la aplicación en *App Engine*. Seleccionamos el proyecto que queramos, y nos vamos al *plugin* de Google donde hacemos click en «Deploy App Engine» como muestra la imagen D.5. A continuación debemos dar un nombre al proyecto y en «App Engine project settings...» especificamos el identificador. Para hacerlo, debemos crear una cuenta aquí<sup>4</sup> y crear un proyecto. Automáticamente se le asignara un ID y ese es el que debemos introducir D.6.

Es importante que el SDK de Java sea el 1.7 ya que de lo contrario se producirá un error y no se desplegará la aplicación. La operación tardará un tiempo y ya solo quedará ir a la plataforma y ver si todo ha salido correctamente.

Si todo ha sido correcto, desde la nube podemos acceder a la base de datos como hacíamos en modo local, incluso con más detalle.

---

<sup>4</sup><https://console.cloud.google.com/appengine>

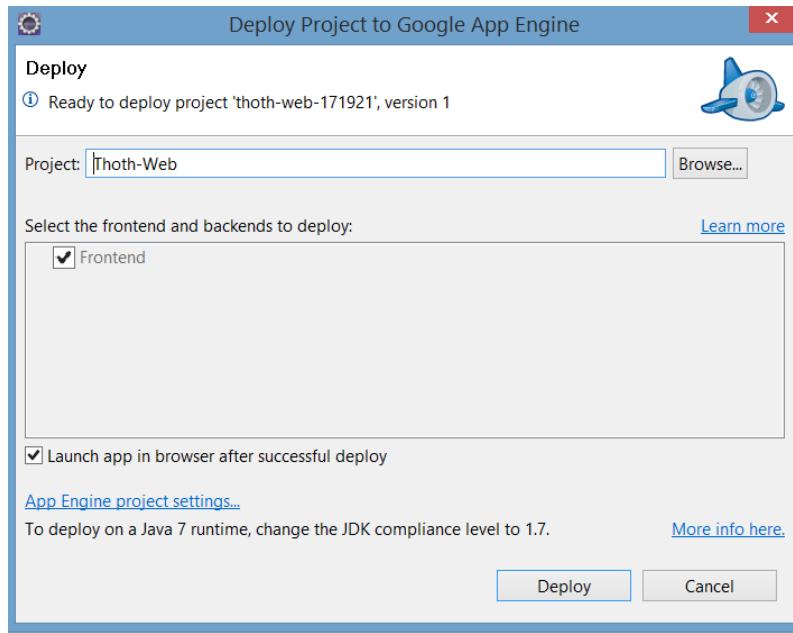


Figura D.6: Opciones de despliegue.

## D.5. Pruebas del sistema

Las pruebas de sistema las hemos realizado con ayuda de Selenium<sup>5</sup>. Selenium es un entorno para realizar pruebas software en varios lenguajes, entre ellos, Java. Esta pensado para utilizarse en aplicaciones web. Para ello se puede descargar una extensión para el navegador, que digamos, grabará los pasos para cada test. Con esto se logra comprobar que el funcionamiento es correcto.

En este proyecto, hemos hecho pruebas sobre las sesiones, analizando, los siguientes casos:

- Inicio de sesión con *email* y contraseña correctos. Se comprueba que el nombre del usuario aparece en el menú. Se cierra la sesión y se vuelve a la vista inicial.
- Inicio de sesión con contraseña incorrecta. Se comprueba que no ha podido acceder. Se prueba si se muestra un mensaje de error.
- Registro con correo no válido. Se comprueba que se muestra un mensaje de error.
- Registro válido. Se comprueba que se registra un usuario y el mensaje de que se devuelve.

---

<sup>5</sup><http://www.seleniumhq.org/>

También se han incluido test de prueba de gramáticas, pero en este caso son iguales a los que se realizaron en las anteriores versiones de Thoth [1].

## *Apéndice E*

---

# Documentación de usuario

---

### E.1. Introducción

En este apéndice se van a mostrar los requisitos, la instalación y un manual para que el usuario final pueda manejar la aplicación de forma fluida. Para ello se ilustrará con figuras que muestren los detalles de los procesos.

### E.2. Requisitos de usuarios

Los requisitos por parte del usuario son mínimos. No es necesario descargarse nada, ni hacer ninguna instalación así que en cuanto a software lo único que necesitará es un sistema operativo común y navegador web actual. Las versiones de compatibles son múltiples.

- Firefox.
- Internet Explorer 8, 9, 10, 11
- Safari 5, 6
- Chromium y Google Chrome.
- Últimas versiones de Opera.

En cuanto a los requisitos *hardware*, es necesaria un equipo con conexión a Internet. No hay requisitos mínimos de procesador o memoria.

La única petición que se le hace al usuario es la de proporcionar unos datos de registro que constan de un nombre con sus apellido, un correo electrónico y una contraseña.

### E.3. Instalación

El proyecto no requiere la instalación de ningún software adicional.

### E.4. Manual del usuario

Una vez cumplamos tengamos un navegador y conexión a Internet ya podemos comenzar a usar la aplicación. Lo primero que tenemos que hacer es introducir la URL <https://1-dot-thoth-web-171921.appspot.com/gramaticacs/> en nuestro navegador. Ya habremos accedido a Thoth. Ahí como inicio nos aparece una ventana de inicio de sesión, como muestra la figura E.1.



Figura E.1: Login en Thoth. Es lo que primero se ve al entrar en la aplicación.

Al ser la primera vez que utilizamos Thoth no tendremos creada una cuenta, así que debemos registrarnos. Si pulsamos en el link *Register* aparecerá una vista con la siguiente pantalla E.2, en la que introduciremos los datos para crear una cuenta. Hay que tener en cuenta que los datos deben estar correctamente rellenados o de lo contrario, la aplicación nos mostrará un error del tipo *Wrong field*, campo incorrecto. Si el fallo es el correo electrónico, se marcará el campo en rojo E.3. En el caso de que el registro haya sido correcto, se mostrará un mensaje indicativo con el siguiente texto *Registration Success*.

Una vez registrados, ahora sí, podremos iniciar sesión. Al hacerlo accedemos a la vista principal de la aplicación que consta de tres elementos principales. Un menú, en el que se despliegan diferentes opciones, un panel en el que introducir la gramática y por último una tabla con las características de la gramática. Esta vista se puede apreciar con detalle en la siguiente imagen E.4.

La parte de azul claro es la que se actualizará, al comprobar una gramática, con las características de esta. Por defecto aparece qué característica es. También podemos apreciar como en el panel aparece ya algo escrito. Esto es para facilitar al usuario la labor al introducir la gramática de escribir ese comienzo. Las gramáticas que se leen en Thoth tienen siempre esa estructura.



Figura E.2: Pantalla de registro.

Full name	javier
E-mail address	<b>javier</b>
Password	•••
	<b>Register</b> <b>login</b>
Wrong field	

Figura E.3: Fallo al introducir el *email*. Se muestra el campo en rojo.

Para empezar, podemos centrarnos en la introducción de la gramática. En el panel podemos copiar, cortar, seleccionar y pegar cualquier carácter. Para comprobar una gramática solo tenemos que pulsar el botón de «chequear» en la parte superior. Por otro lado si queremos buscar y reemplazar un símbolo, el botón de renombrado aparece debajo [E.5](#).

Entonces, comenzamos introduciendo una gramática. En este caso los símbolos  $\epsilon$  se representan con la letra E mayúscula de esta forma  $\hat{E}$ . Comprobamos la gramática y en la parte inferior aparecen las características de esta, como se ve en la captura [E.6](#).

Para renombrar aparecerá una ventana como la que aparece en [E.7](#). En la parte superior introducimos el carácter a sustituir. Como se ve, aparece en forma de desplegable los caracteres iguales o similares. En la gramática es diferente la «a» que la «A». Abajo se introduce el carácter que le sustituye.

Cuando se pulsa en aceptar se actualizará la gramática con la sustitución hecha. Entonces habrá que salir del panel.

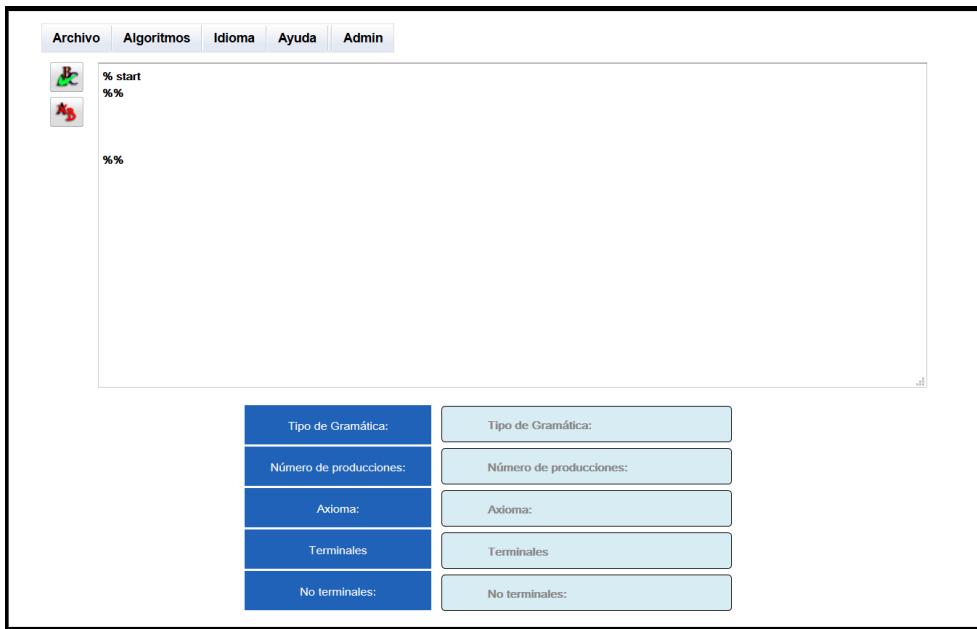


Figura E.4: Pantalla principal con el menú de la aplicación y el panel para crear una gramática.

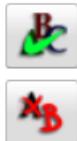


Figura E.5: En la parte superior comprobar gramática y abajo renombrar. Ambos botones son los mismos que en las otras versiones de Thoth.

Dentro de las opciones del menú, quizá la más utilizada es la de elegir uno de los algoritmos a aplicar a la gramática. En el desplegable se pueden ver todas las opciones que hay. Para no extenderlos, mostraré las vistas de los algoritmos que sean diferentes a las demás. Antes de aplicar el algoritmo se debe comprobar la gramática, de lo contrario, la aplicación la comprobará automáticamente.

El algoritmo escogido es el de Eliminación de Símbolos Anulables como se puede apreciar en la imagen E.8. Es muy similar a los dos primeros algoritmos. Se puede apreciar como se resaltan las producciones en color verde y otras en rojo. Esto sólo ocurre cuando se aplica el algoritmo paso a paso. De lo contrario, no mostrará el proceso, y simplemente aparecerá a un lado la gramática antigua y al otro la nueva.

The screenshot shows a software interface for grammar analysis. At the top, there's a menu bar with 'Ayuda', 'Algoritmos', 'Idioma', and 'Javier Paramo'. Below the menu is a toolbar with icons for different grammar types: LR(0), LR(1), LR(2), LR(3), NFA, DFA, and NFA-DFA. The main workspace contains the following text:

```
% start S
%%
S : ABC|dCA;
A : aE|aAcBb;
B : AA|bB|C Bc;
C : AB|cC|e;

%%
```

Below this, there's a table showing the analysis results:

Tipo de Gramática:	Independiente del contexto
Número de producciones:	11
Axioma:	S
Terminales	c, d, a, E, b, e
No terminales:	S, A, B, C

Figura E.6: Comprobación de una gramática y como se actualiza la tabla inferior.

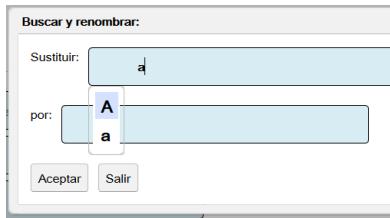


Figura E.7: Renombrado de símbolos.

En verde se iluminan las producciones que pasan a la nueva gramática en ese paso, y en rojo se resalta el símbolo anulable que se está analizando. Si pulsamos en «cancelar» se vuelve a la vista principal de edición de gramática con la gramática antigua. Por el contrario se pulsamos en «aceptar» volveremos a la vista principal pero con la gramática resultante. El botón aceptar solo se activará una vez se haya terminado de aplicar el algoritmo, de lo contrario no tendría sentido.

Los demás algoritmos tiene un *modus operandi* parecido, por lo que omitiré el proceso. Resaltar que tanto el algoritmo de Limpieza de Gramática como el de Eliminar Recursividad se aplican de forma directa. Es decir cuando se ejecutan no se pasa a una vista nueva, sino que se actualiza la gramática ya en el panel.

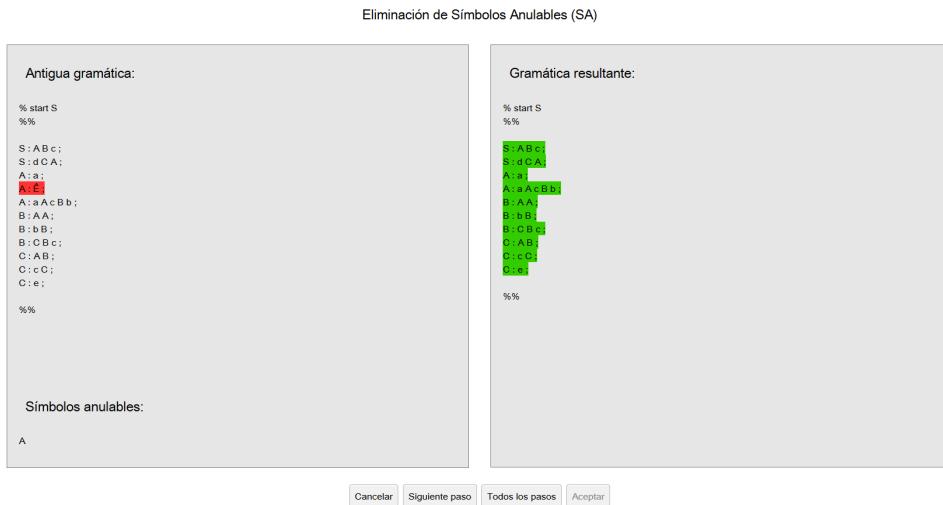


Figura E.8: Ejemplo de una vista del algoritmo Eliminación de SA.

Los algoritmos que cambian bastante en la forma de proceder son el cálculo del First y el Follow y el reconocimiento por medio de TASP. El cálculo de First Follow deriva al reconocimiento con TASP, ya que ambos son algoritmos de análisis ascendente. En el ejemplo de First-Follow con cada botón se crea la tabla correspondiente E.9. Los botones de desactivan a mediada que se utilizan.

Cálculo First y Follow							
Gramática							
<pre>% start E %% F' : ""; F' : E; G : 0; G : 1; G : u; G : (E); E : tE'; E' : +tE'; E' : ē; T : FT'; T' : FT''; T' : ē; F : GF'; %%</pre>							
First							
E	F'	G	T	E'	F	T'	
0	*	0	0	+	0	-	
1	ē	1	1	ē	1	ē	
u		u	u		u		
(		(	(		(		
Follow							
E	F'	G	T	E'	F	T'	
\$	-	*	+	\$	-	+	
)	+	-	\$	)	+	\$	
	\$	+	)		\$	)	
)	\$				)		

Figura E.9: Vista del algoritmo para el Cálculo de Fisrt Follow.

Antes de ejecutar ambos algoritmos se pide confirmación adicional al usuario antes de proceder. Se conservan la forma de las anteriores versiones de Thoth. Como estos dos algoritmos no generan gramática nuevas no existe un botón de aceptar.

Para el reconocimiento con TASP, se muestra primero la Tabla de Análisis Sintáctico Predictivo (TASP) y aparece un recuadro en el que introducir la palabra a analizar [E.10](#). Verificamos que es la palabra que queremos, o podemos borrarla con los dos botones que aparecen a su lado. A continuación se crea la traza. podemos ejecutarla paso a paso o todos los pasos a la vez, el resultado es el mismo. Al terminar de calcular la traza se mostrará un mensaje advirtiendo si se ha reconocido o no la palabra. podremos volver a la vista principal.

TASP

	E	F'	G	T	E'	F	T'
\$		F':È;			E':È;		T':È;
*		F':*;					
0	E : T E' ;		G : 0 ;	T : F T' ;		F : G F' ;	
1	E : T E' ;		G : 1 ;	T : F T' ;		F : G F' ;	
u	E : T E' ;		G : u ;	T : F T' ;		F : G F' ;	
(	E : T E' ;		G : ( E ) ;	T : F T' ;		F : G F' ;	
)		F':È;			E':È;		T':È;
+		F':È;			E' : + T E' ;		T':È;
-		F':È;				T' : - F T' ;	

Traza de reconocimiento:

Palabra:	1*\$	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Pila	Tope	Entrada	Carácter	Salida
\$ E	E	1 * \$	1	
\$ E' T	T	1 * \$	1	E : T E' ;
\$ E' T' F	F	1 * \$	1	T : F T' ;
\$ E' T' F' G	G	1 * \$	1	F : G F' ;
\$ E' T' F' 1	1	1 * \$	1	G : 1 ;
\$ E' T' F'	F'	* \$	*	POP
<input type="button" value="Cancelar"/> <input type="button" value="Siguiente paso"/> <input type="button" value="Todos los pasos"/>				

Figura E.10: Vista del reconocimiento con TASP. Se aprecia la zona donde introducir la palabra y los botones.

Volviendo al menú podemos apreciar que en las opciones aparece el nombre

con el que se ha registrado el usuario. Si pulsamos en ello podremos ver como se despliega la opción para cerrar la sesión (ver captura E.11). Al cerrar sesión se volverá a la vista de *login* y deberemos acceder de nuevo escribiendo el correo y la contraseña.



Figura E.11: Menú con el nombre del usuario. El desplegable muestra el cierre de la sesión.

Siguiendo con el menú, la opción del idioma permite cambiar los textos de la aplicación, al idioma elegido. Como la vista se va a reiniciar y la gramática desaparecerá, es necesario mostrar un mensaje que solicita confirmación adicional, como se aprecia en la figura E.12. Si pulsamos en «si» se llevará a cabo el reinicio, y como ya mantenemos una sesión activa, no será necesario volver a iniciar sesión. Si pulsamos «no» el *popup* desaparecerá.

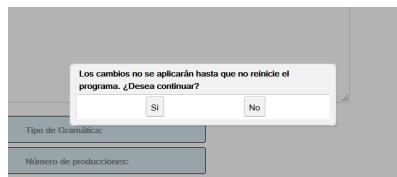


Figura E.12: Mensaje de confirmación para cambiar el idioma.

---

## Bibliografía

---

- [1] César García-Osorio, Javier Jimeno Visitación, and Íñigo Mediavilla Saiz. Proyecto thoth v2 ( simulador de autómatas y máquinas de turing), 2007.
- [2] Juan Pavón Mestras. El patrón modelo-vista-controlador, 2009.