



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



Presentado por Francisco Javier Páramo Arnaiz
en Universidad de Burgos — 1 de julio de 2017

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



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática



Dr. César Ignacio García Osorio y D. Álgar Arnaiz González, profesores del departamento de Departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Francisco Javier Páramo Arnaiz, con DNI 71293768-R, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Conversión de la aplicación docente Thoth a JavaScript.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 1 de julio de 2017

Vº. Bº. del Tutor:

Vº. Bº. del co-tutor:

D. Álgar Arnaiz González

Dr. César Ignacio García Osorio

Resumen

La teoría de autómatas y lenguajes formales es uno de los campos más antiguos en el ámbito de la informática que, a pesar de los constantes avances, sus principales conceptos continúan siendo de referencia para el estudio teórico y práctico de la informática. Esto es la base sobre la que nació el mundo de las tecnologías que conocemos hoy en día.

Thoth es un antiguo trabajo de fin de carrera de la extinta Ingeniería Técnica en Informática que, cuenta con varias versiones realizadas por alumnos de la Universidad de Burgos y que sirve, precisamente, como herramienta de ayuda en la docencia de la teoría de autómatas y lenguajes formales. Este proyecto pretende llevar Thoth a la web, sumando nuevas funcionalidades propias de este tipo de aplicaciones, como el registro de usuarios o el inicio de sesión, que la harán más útil y completa.

Para ello se hace un estudio sobre las posibles tecnologías que ayudarán a conseguir el objetivo. De esta forma se determina como herramienta para lograr este cambio el *framework* denominado *Google Web Toolkit*, que consigue traducir parte del código escrito en Java a código JavaScript. GWT, según sus siglas, es una tecnología algo antigua pero muy útil para este proyecto, que combina código del lado del cliente y código del lado del servidor, cada uno con una funcionalidad, y que se comunican entre ellos vía *Remote Procedure Call*.

El trabajo fundamental es adaptar el código Java para que el *framework* logre interpretarlo y traduzca lo necesario para que la aplicación se ejecute en cualquier navegador. Se trata de aprender los entresijos de GWT, sacando partido de la dualidad cliente-servidor y conservando todo lo bueno, que es mucho, realizado en la aplicación de escritorio de Thoth.

Descriptores

Autómatas y lenguajes formales, procesadores de lenguajes, transformación de gramáticas, análisis descendente, aplicación cliente-servidor

Abstract

The theory of automata and formal languages is one of the oldest fields in the ambit of computer science in which, despite the constant advances, its main concepts continue to be a reference for the theoretical and practical study of computer science. This is the basis of the world of technologies that we learn today.

Thoth is an old end-of-course work of the extinct Technical Engineering in Computer Science that, has several versions realized by students of the University of Burgos and that serves like tool for help in the teaching of the theory of automata and formal languages. This project aims to bring Thoth to the web, adding new features of this type of applications, such as user registration or login, which will make it more useful and complete.

This framework, commonly known by its acronym, GWT, is a somewhat old technology but very useful for this project, it combines client-side code and server-side code, each with its own function, which communicate with each other via Remote Procedure Call.

The main work is to adapt the Java code to make the framework interpret it and translate what is necessary for the application to run in any browser. It's about learning about GWT, taking advantage of the client-server duality and retaining all the benefits, which is a lot, provided by the desktop version of Thoth.

Keywords

Automata and formal languages, language processors, grammars transformation, top-down parser, client-server application

Índice general

Índice general	III
Índice de figuras	V
Índice de tablas	VI
Introducción	1
Objetivos del proyecto	3
Conceptos teóricos	5
3.1. ¿Qué es una gramática y para qué sirve?	5
3.2. Recursividad	7
3.3. Tipos de gramáticas	7
3.4. Cifrado de contraseñas y seguridad	9
Técnicas y herramientas	13
4.1. GWT	13
4.2. WebSwing	14
4.3. JSweet	15
4.4. DukeScript	15
4.5. Vaadin	16
4.6. Herramientas para el cifrado de contraseñas	17
4.7. Plataforma <i>Google Cloud</i>	18
Aspectos relevantes del desarrollo del proyecto	20
5.1. Internacionalización	20
5.2. Google Web Toolkit	21
5.3. Registro e inicio de sesión de usuarios	23
Trabajos relacionados	25

<i>ÍNDICE GENERAL</i>	IV
Conclusiones y Líneas de trabajo futuras	26
7.1. Conclusiones	26
7.2. Líneas de trabajo futuras	27
Bibliografía	28

Índice de figuras

3.1. Aplicación de la función hash a diferentes datos introducidos. . . .	10
3.2. Ejemplo de ataque con tablas de búsqueda.	11
3.3. Resultado de aplicar tablas de búsqueda al cifrado hash con semilla.	11
4.4. Base de datos en <i>Google Cloud</i> . Se puede apreciar la entidad «User» que muestra la base de datos.	19
5.5. Vista de los estilos aplicados sobre un botón del menú desde el navegador.	23

Índice de tablas

4.1. Tabla comparativa entre Vaadin y GWT.	17
--	----

Introducción

Este proyecto nació con el objetivo de llevar Thoth [3], un antiguo proyecto escrito en Java, a la web. Para ello se utilizarán tecnologías web, que puedan ser utilizadas en diferentes dispositivos haciéndolo accesible a todo el mundo. Con ayuda de la herramienta conocida como GWT o *Google Web Toolkit*¹ según su denominación inglesa, traduciré la aplicación a Javascript en el lado del cliente haciendo posible la utilización de la aplicación directamente desde un navegador, algo que antes era imposible.

Thoth [3] es un antiguo proyecto enfocado a la actividad docente y relacionado con los procesadores de lenguaje, que fue realizado por varios alumnos de la Universidad de Burgos como trabajo de fin de carrera. Esta aplicación cuenta con dos versiones hasta la fecha y también existen otros desarrollos con la misma inspiración como Web Thoth [2].

Una de las principales preguntas que nos podemos hacer al ver este proyecto es por qué traducir la aplicación al lenguaje JavaScript. En primer lugar, todos los navegadores actuales son capaces de interpretar el código escrito en JavaScript y soportan al menos alguna de las versiones ECMAScript [4], siendo la última la séptima edición, disponible desde 2016. Con ayuda de la tecnología AJAX, se puede ejecutar la aplicación en el cliente, es decir, en el navegador de un usuario, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Utilizo GWT para este proyecto como herramienta para transformar la aplicación de escritorio a entornos web. Pese a que actualmente no es el mejor *framework* para desarrollar, sí es una de las bases de otro mucho más moderno y potente. Hablo de Vaadin, que se encuentra más actualizado y con más bibliotecas para un aspecto visual más moderno, pero que para poder hacer disfrutar de su versión completa, se necesita una licencia bajo pago. La forma

¹<http://www.gwtproject.org/>

de trabajar con GWT es creando el código en Java y el compilador hará una traducción a los lenguajes JavaScript y HTML.

Una de las cuestiones más importantes de este *frameworks* es que GWT no es capaz de admitir todas las bibliotecas de Java y por eso debimos reescribir el código adaptándolo a las capacidades de este *framework*. Encontramos otras opciones de las que posteriormente hablaré, que son capaces de hacer la traducción completa de Thoth en Java a HTML5 de una forma muchísimo más sencilla y directa sin necesidad, si quiera, de esforzarse demasiado. Pero ese no es el objetivo de un trabajo de fin de grado, sino el de plantear un desafío suficientemente grande con el que el alumno pueda hacer uso del conjunto de muchos de los conocimientos aprendidos a lo largo de grado.

Objetivos del proyecto

El objetivo principal de este trabajo de final de grado consiste en transformar el proyecto previo de Thoth, que está escrito en Java, a la tecnología web JavaScript. Esta conversión se llevará a cabo por medio de GWT, herramienta elegida posteriormente a un estudio previo (disponible en el capítulo Técnicas y Herramientas).

El desarrollo consiste en una página web dinámica que sea capaz de hacer las mismas funcionalidades sobre gramáticas formales y sus algoritmos que en Thoth, poniendo a prueba mis propios conocimientos sobre Java así como otros lenguajes sobre tecnologías web como son HTML, CSS o JavaScript.

Pero la razón por la que pretendemos pasar de una aplicación de escritorio a otra que se ejecuta en el navegador es básicamente porque no es necesario instalar o descargar material para poderla utilizar Thoth, simplemente algunos conocimientos básicos sobre gramáticas formales. Por lo tanto aporta una clara ventaja con respecto a aplicaciones denominadas «de escritorio» como es el Thoth original sobre el que nos basamos en este proyecto.

La conversión de Thoth a GWT es necesaria porque amplía las posibilidades de mejora de las anteriores versiones. Pretendemos aportar mayor funcionalidad debido a que, al ser una aplicación web, podemos hacer un registro de los usuarios, con inicio de sesión, registro de actividades, etc.

De hecho, uno de los puntos fuertes de este proyecto consiste en aprovechar la mayor parte de utilidad de las aplicaciones web. Aunque en las anteriores versiones la pretensión era sobre todo didáctica en materia relacionada con el estudio de los procesos del lenguaje, ahora, además de eso, también queremos llegar a poder saber la utilidad que se le da a Thoth. Para ello, gracias al registro de información sobre los usuarios, podremos saber cuando se ha registrado alguien, o iniciado sesión, qué gramáticas ha usado y más funcionalidades que puedan surgir. Eso sí, siempre manteniendo la privacidad del usuario, cifrando la contraseña de registro.

Por último está el objetivo de experimentar con herramientas que no conocemos, y de esta forma ir aprendiendo a solucionar problemas nuevos. Al fin y al cabo lo que se trata de aprender en la universidad es superar estos retos por uno mismo, aplicando las técnicas que se nos enseñan.

Los objetivos del proyecto quedan reflejados de esta manera:

- Estudio de aplicaciones de conversión automática de Java a la web.
- Convertir la parte de las gramáticas de Thoth mediante GWT
- Crear un sistema de registro de usuarios para el acceso a la aplicación.
- Crear un sistema de sesiones de acuerdo al registro.

Conceptos teóricos

En mi experiencia como profesor de programación, diseño 3D y robótica para alumnos de primaria, lo primero que les enseñábamos era a definir la programación como el lenguaje de comunicación entre nosotros, los humanos, y los ordenadores o los robots. Todo esto pertenece a los conceptos más puramente teóricos sobre la computación y que es, en sí, la base de la informática.

Pues bien, a *grosso modo* el concepto es el mismo ahora. Como cada uno, hombre y máquina, «habla» un idioma diferente se debe establecer un lenguaje que sea la vía de comunicación entre ambos.

Para poder entender todo lo que se va a tratar a continuación, debo explicar primero el concepto de *autómata* o *máquina abstracta*. Un autómata es un modelo matemático o un dispositivo teórico que recibe una cadena de símbolos como entrada y que al procesarla, genera un cambio de estado produciendo una salida determinada. Esta salida puede reconocer palabras y determinar si la entrada pertenece a un determinado lenguaje o no. En el símil anterior, un autómata es como el corrector, que determina si algo está bien escrito o no.

Entonces ya sabemos que, para que haya comunicación entre un ordenador o robot y un humano, tiene que haber un lenguaje y un autómata. Pero también algo más: una gramática.

3.1. ¿Qué es una gramática y para qué sirve?

Una gramática formal es un mecanismo para la generación de cadenas de caracteres que definen un determinado lenguaje formal, y que utiliza un conjunto de reglas de formación. Por lo tanto, podemos entenderlo dentro del contexto de las ciencias de la computación y la lógica matemática. Las cadenas de caracteres resultantes son a su vez «bien formadas» cuando pertenecen al lenguaje formal con el que se trabaja [1].

¿Y por qué es tan importante? Siguiendo con el ejemplo del principio, la gramática es la que va a determinar si lo que se introduce en el autómata es correcto o no. Un conjunto de reglas que nos indicará por qué al juntar una serie de caracteres de una forma se van a poder entender.

Por otro lado, la denominación de la gramática formal, desde un punto de vista más formal, es una cuádrupla compuesta por:

- Un alfabeto de **símbolos terminales** o tókenes denotado con la letra griega Σ .
- \mathcal{N} que es un alfabeto formado por **símbolos no terminales**.
- Un alfabeto de **producciones** denominado \mathcal{P} .
- Y por último, un símbolo llamado **axioma** o símbolo inicial, denotado por \mathcal{S} y tal que $\mathcal{S} \in \mathcal{N}$.

El alfabeto total que compone la gramática está formado, según lo anterior, por $\Sigma \cup \mathcal{N}$, es decir, por el conjunto de los símbolos terminales y no terminales.

Una **producción** tiene esta estructura y está compuesta por un par ordenado de cadenas.

$$x \rightarrow y$$

A la parte izquierda se la denomina **antecedente** y la derecha **consecuente**. A las producciones también se las denomina **reglas de derivación**.

Pongamos un ejemplo de una gramática simple y veamos de que está formada. Se suele utilizar la notación

$$x \rightarrow y$$

$$z \rightarrow w$$

para indicar una o varias producciones, en vez de

$$(x, y) \in \mathcal{P}$$

$$(z, w) \in \mathcal{P}$$

siendo \mathcal{P} el conjunto de producciones.

Por otro lado, si hay más de una producción que comience con el mismo elemento la notación sería de esta forma

$$x \rightarrow y|z|w$$

en lugar de ser

$$x \rightarrow y$$

$$x \rightarrow z$$

$$x \rightarrow w$$

Cuando la gramática tiene una producción $x \rightarrow y$, se dice que w **deriva directamente** de v ($v \Rightarrow w$) si $\exists z, u \in \Sigma^*$ tales que $v = zxu, w = zyu$.

Sea Σ un alfabeto, un conjunto de producciones \mathcal{P} y $v, w \in \Sigma^*$, decimos que w **deriva** de v , si existen $u_0, u_1, \dots, u_n \in \Sigma^*$ y una secuencia de transformaciones tales que:

$$v = u_0 \Rightarrow u_1$$

$$u_1 \Rightarrow u_2$$

$$\dots$$

$$u_{n-1} \Rightarrow u_n = w$$

Esta secuencia de transformaciones se dice que es una **derivación** de longitud n . Se habla de **lenguaje** generado por una gramática G , denotado por $L(G)$, y se define como el conjunto de todas las sentencias de la gramática G .

$$L(G) = \{x \in \Sigma^* : \mathcal{S} \Rightarrow^+ x\}$$

De esta manera, se dice que dos gramáticas son equivalentes $G_1 \equiv G_2$, si generan el mismo lenguaje, es decir, si $L(G_1) = L(G_2)$.

3.2. Recursividad

Por último, una gramática es recursiva en un símbolo no terminal U cuando existe una forma sentencial de U que contiene a U .

$$U \Rightarrow^+ xUy \text{ donde } x, y \in (\Sigma \cup \mathcal{N})^*$$

Así pues, la gramática será recursiva cuando lo sea para algún no terminal, si $x = \varepsilon$ la gramática es recursiva por la izquierda, y si $y = \varepsilon$ se dice que es recursiva por la derecha.

3.3. Tipos de gramáticas

Hay varios tipos de gramáticas según sus características [1].

- **Gramáticas de Chomsky:** las producciones tienen la forma

$$u \rightarrow v \text{ donde } u = xAy \in (\Sigma \cup \mathcal{N})^+ \wedge A \in \mathcal{N} \wedge x, y, v \in (\Sigma \cup \mathcal{N})^*$$

Es posible demostrar que los lenguajes generados por una gramática de Chomsky se pueden generar por un grupo más restringido, llamadas gramáticas con estructura de frase, es decir, que ambas tienen la misma capacidad generativa. Las gramáticas de frase tienen la siguiente forma de producción:

$$xAy \rightarrow xvy \text{ donde } x, y, v \in (\Sigma \cup \mathcal{N})^* \wedge A \in \mathcal{N}$$

- **Gramáticas dependientes o sensibles al contexto:** Cuenta con las reglas de producción de la forma

$$xAy \rightarrow xvy \text{ donde } x, y \in (\Sigma \cup \mathcal{N})^* \wedge A \in \mathcal{N} \wedge v \in (\Sigma \cup \mathcal{N})^+$$

En este tipo de gramáticas el significado de A depende del contexto o de la posición en la frase. El contexto sería entonces x e y . Además, la longitud de la parte derecha de las producciones es siempre mayor o igual que la de la parte izquierda.

- **Gramáticas independientes del contexto:** las producciones de las gramáticas de este tipo son más restrictivas, de la forma:

$$A \rightarrow v \text{ donde } A \in \mathcal{N} \wedge v \in (\Sigma \cup \mathcal{N})^*$$

Como su propio nombre indica, el significado de A es independiente de la posición en la que se encuentra. La mayor parte de los lenguajes de ordenador pertenecen a este tipo. Una característica importante es que las derivaciones obtenidas al utilizarse esta gramática se pueden representar utilizando árboles.

- **Gramáticas regulares:** es el grupo más restringido. Tienen la forma:

$$A \rightarrow aB \wedge A \rightarrow b \text{ llamadas gramáticas regulares a derechas}$$

$$A \rightarrow Ba \wedge A \rightarrow b \text{ llamadas gramáticas regulares a izquierdas}$$

$$\text{donde } A, B \in \mathcal{N} \wedge a, b \in \Sigma$$

Ambas son equivalentes. Existe también una generalización de este tipo de gramáticas denominadas lineales con reglas de la forma:

$$A \rightarrow wB \wedge A \rightarrow v \text{ lineales a derechas}$$

$$A \rightarrow Bw \wedge A \rightarrow v \text{ lineales a izquierdas}$$

$$\text{donde } A, B \in \mathcal{N} \wedge w, v \in \Sigma^*$$

que son totalmente equivalentes a las regulares normales, pero en muchos casos su notación es más adecuada.

3.4. Cifrado de contraseñas y seguridad

El cifrado de la contraseña es una de las cuestiones más importantes para preservar la seguridad o intimidad del usuario. El porqué es muy simple. Los usuarios normalmente utilizamos las mismas contraseñas o parecidas para cualquier cuenta. Por lo tanto, si hay alguien con acceso a la base de datos, podrá ver la contraseña utilizada por un determinado usuario, poniendo en peligro esa intimidad, no solo en este programa, sino, como ya se ha comentado antes, en las de otras cuentas.

Para evitar esto, el método más simple es el cifrar las contraseñas en la base de datos para que en el caso de que alguien acceda a ella, en el campo «contraseña» no vea la contraseña real, sino el resultado del cifrado de esta. El método de cifrado que se emplea en este proyecto es el del algoritmo *hash* que como veremos más adelante tiene unas características determinadas.

Funcionamiento del algoritmo hash

Se trata de un algoritmo matemático con el que se transforma cualquier cantidad de datos en una serie de datos fija que funciona como una huella dactilar. Esto quiere decir que sea cual sea la cantidad de caracteres de entrada, la salida siempre será fija. Cumple además con dos premisas muy importantes para la seguridad:

- No es reversible, no se puede descifrar por medio de funciones matemáticas y obtener el resultado antes de ser encriptado, sea cual sea la función utilizada (SHA-1, SHA-2 o MD 5 entre otras, como veremos a continuación).
- Cuenta con la propiedad de que si la entrada cambia, aunque sea sólo en un bit, el *hash* resultante será completamente distinto, como se puede ver en la ilustración 3.1. En la imagen se puede apreciar que aunque la entrada tenga un mayor número de caracteres, la salida siempre será de 40 caracteres.

Ahora bien, las acciones llevadas a cabo para preservar esa seguridad serían las siguientes: un usuario crea una cuenta en la aplicación, la contraseña se encripta y se almacena en la base de datos. Cuando el usuario trata de iniciar sesión y escribe la contraseña, esta se encripta y se compara el resultado con aquel que se ha guardado en la base de datos, si son iguales el usuario tendrá acceso; sino, se le requerirá que lo vuelva a intentar.

El problema principal de esto es que con los avances en temas de seguridad siempre hay asociados otros que tratan de «romper» esa seguridad, y en este caso no iba a ser menos. Creo que es importante reconocer los peligros que hay

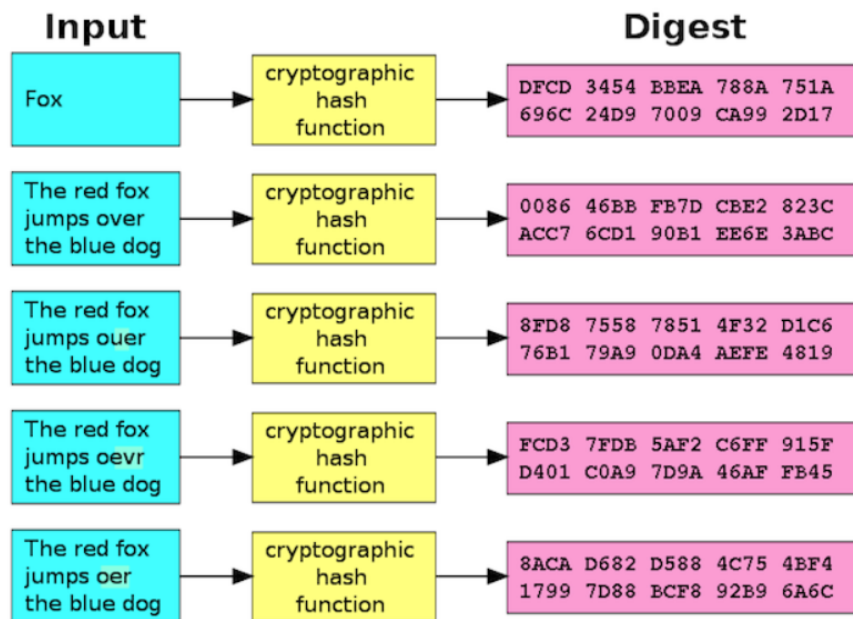


Figura 3.1: Aplicación de la función hash a diferentes datos introducidos <https://blog.kaspersky.com.mx/que-es-un-hash-y-como-funciona/2806/>.

asociados a aplicar este método, pero no deseo extenderme demasiado en este aspecto así que los mencionaré brevemente.


- Ataques de fuerza bruta. Consisten en utilizar diccionarios de palabras con contraseñas habituales e introducirlas hasta que alguna coincida. Se trata del ataque menos eficiente, pero el más difícil de evitar.
- Tablas de búsqueda. Este tipo de ataque sí que supondría un grave problema para la seguridad en el cifrado con algoritmo *hash*. Recordemos que las funciones *hash* solo se pueden encriptar, no descifrar. Lo que hace este ataque es lo siguiente: cuenta con una tabla de contraseñas típicas y su cifrado *hash* y las compara con los *hash* introducidos. Para entender mejor este concepto hay incluso herramientas *online* que pueden hacer este trabajo. Ver ilustración 3.2. También las hay que funcionan al revés. Introduces la contraseña que crees que puede usar alguien, la encripta, la compara con todas las de la base de datos y te dice si alguien la utiliza o no.

Además de estos, hay más métodos, casi todos basados en las tablas de búsqueda. Según estos ataques queda comprobado que la seguridad de este algoritmo depende en gran medida de la contraseña que utilice el usuario.

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
c11083b4b0a7743af748c85d343dfee9fbb8b2576c05f3a7f0d632b0926aadfc
08eac03b80adc33dc7d8fbc44b7c7b05d3a2c511166bd43fcb710b03ba919e7
e4ba5cbd251c98e6cd1c23f126a3b81d8d8328abc95387229850952b3ef9f904
5206b8b8a996cf5320cb12ca91c7b790fba9f030408efe83ebb83548dc3007bd
0ad066a5d29f3f2a2a1c7c17dd082a79
828c1a17681e8566a17a1a4801ea67306010b273
```



Introduzca el texto

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
c11083b4b0a7743af748c85d343dfee9fbb8b2576c05f3a7f0d632b0926aadfc	sha256	@1ph4
08eac03b80adc33dc7d8fbc44b7c7b05d3a2c511166bd43fcb710b03ba919e7	sha256	P8\$5W0RD
e4ba5cbd251c98e6cd1c23f126a3b81d8d8328abc95387229850952b3ef9f904	sha256	bananas
5206b8b8a996cf5320cb12ca91c7b790fba9f030408efe83ebb83548dc3007bd	sha256	crackstation
0ad066a5d29f3f2a2a1c7c17dd082a79	md5	hola mundo
828c1a17681e8566a17a1a4801ea67306010b273	sha1	javier

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Figura 3.2: Ejemplo de ataque con tablas de búsqueda <https://crackstation.net/>.

Cuanto más aleatoria y con más mezcla de caracteres mejor, ya que formará palabras que no se encuentran en los diccionarios o tablas y solo se podrá descubrir por medio de la fuerza bruta. Entonces ¿cómo hacer que las contraseñas sean más resistentes y solucionar este problema?

Se conoce como el cifrado *hash* con sal o semilla. Consiste en añadir un conjunto de caracteres aleatorios, agregarlos a la contraseña y una vez hecho esto, cifrarlo con la función *hash*. De esta manera se consigue que la contraseña sea mucho más aleatoria que la que inicialmente ha introducido el usuario. Así podemos ver como quedaría un intento de «tablas de búsqueda» con este tipo de cifrado se usa para contraseñas muy simples 3.3.

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
b5374dbbe4eeabf3c6e729e9546b49e	Unknown	Unrecognized hash format.
5206b8b8a996cf5320cb1690f326221	Unknown	Unrecognized hash format.
a2dfa93cbda3eaab937ae2993495788	Unknown	Unrecognized hash format.

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Figura 3.3: Resultado de aplicar tablas de búsqueda al cifrado hash con semilla. Imagen sacada de <https://crackstation.net/>

La primera de ellas corresponde con la clave «123» la segunda con la palabra «contraseña» y la tercera con la fecha «16-6-17». Todas son fáciles, pero al añadirle una clave y cifrarlo todo, se vuelve mucho mas complejo. Se podría pensar que, como la clave que se le añade también se guarda en la base de datos sigue sin ser seguro. Partiendo del hecho de que no hay prácticamente

nada seguro al cien por cien, lo que se consigue con esto es dificultar mucho el cálculo, ya que los dos ataques mencionados anteriormente (y varios más basados en estos dos) se ayudan de repositorios de contraseñas. Así, si quisieran descifrarlo, debería de coger el conjunto aleatorio, sumarle la posible contraseña y cifrarlo comparando los resultados sólo con ese *hash* (o contraseña cifrada) ya que el *salt* es único para cada *hash*. Con este método se consigue que los cálculos para descifrar una contraseña sean bastante más costosos de una forma muy simple.

Técnicas y herramientas

Dentro de las diferentes herramientas que utilizaré para la realización de este trabajo, la más importante es aquella con la cual realizaré la conversión a JavaScript. Es por ello esencial hacer una buena elección comparando y analizando las diferentes posibilidades a elegir.

Como principales herramientas para la conversión de Java a JavaScript he podido encontrar GWT (Google Web Toolkit), JSweet, WebSwing, Vaadin y DukeScript. Aunque también hay otras, como Rhino², que descartamos por su poca relevancia o información.

4.1. GWT

Google Web Toolkit es un framework ampliamente conocido por los desarrolladores web, entre otras cosas, gracias a ser de código abierto, de su gran utilidad y calidad además de ser completamente gratuito³. Contiene una SDK que proporciona un conjunto de APIs de Java que permiten el desarrollo de aplicaciones AJAX (*Asynchronous JavaScript And XML*) escritas en Java. Posteriormente compila el código en JavaScript ya optimizado dando rapidez a la aplicación web.

AJAX es una técnica que se emplea en el desarrollo web, para ejecutar aplicaciones en el lado del cliente en el navegador, comunicándose de manera asíncrona con el servidor. Todas las comunicaciones que se hacen en GWT por medio de RPC son asíncronas, esto quiere decir que las peticiones no se bloquean unas a otras mientras están esperando respuesta, y una vez recibidas, se ejecutan en segundo plano.

Básicamente, permite a los desarrolladores compilar código Java en archivos JavaScript ya optimizados de forma autónoma, proporcionando así todas

²<https://github.com/mozilla/rhino?files=1>

³<http://www.gwtproject.org/>

las ventajas de las aplicaciones escritas en este último lenguaje.

GWT permite compartir código escrito en Java en la parte del servidor con código JavaScript en la parte del cliente lo que nos lleva pensar que la aplicación resultante será fiel a la idea inicial del Thoth.

Nos decantamos por GWT porque, a parte de que supone un aprendizaje para mí como alumno, también es la base de otros *frameworks* de los que más tarde hablaré. Ha sido muy utilizado anteriormente y ahora está digamos que en decadencia. El soporte actual es mínimo y sobre todo en el tema visual anda algo anticuado. En el desarrollo del proyecto hemos llegado a ver y probar «*bugs*» que según otros usuarios ya descubrieron hace un par de años.

Aun así, hay bastante información con la que he podido trabajar, y una comunidad grande, que aunque ahora se ha «pasado» a otros *frameworks* más actuales, han dejado huella y soluciones a muchos de los problemas con los que he trabajado.

4.2. WebSwing

En cuanto a esta herramienta, es algo diferente a las demás. WebSwing⁴ la descubrimos debido a una duda que nos surgió al principio del proyecto. Y es que la aplicación Thoth original cuenta con muchísimos elementos de la biblioteca gráfica «*Swing*», es digamos toda la estructura visual que utiliza. El problema surgió cuando al programar en GWT no podíamos hacer uso de ella, ya que, al ser algo visual debe ir en la parte del cliente y como ya hemos mencionado, es donde se hace la traducción a JavaScript.

Se trata de un servidor web que permite la ejecución de aplicaciones que utilicen la biblioteca gráfica Swing desde el navegador, utilizando sólo HTML5. De esta forma toda la aplicación de Thoth se ejecutaría en el navegador conservando su aspecto de siempre y manteniendo las ventajas de una aplicación web.

Lamentablemente, esta herramienta se descubrió cuando el proyecto ya estaba muy avanzado. Decidimos entonces continuar por la senda marcada en un principio. Se hizo alguna prueba para ver su funcionamiento y los resultados fueron muy buenos.

Además, utilizar esta herramienta no hubieran supuesto ningún reto como informático y facilitaría tanto el proyecto que este quedaría en nada más que unas simples mejoras de Thoth hechas con poco desarrollo. Por lo tanto, la descartamos después de haberla probado.

⁴<http://webswing.org/>

4.3. JSweet

JSweet⁵ es básicamente un «transpiler», es decir, un compilador que traduce un código en un lenguaje a otro lenguaje. Al igual que GWT está orientado a objetos, que proporciona una programación segura gracias a que usa un sistema de «tipado» Java.

La diferencia fundamental con GWT es que al ser un «transpiler» hace una traducción directa entre Java y JavaScript posicionando el código a un lado o al otro del cliente y el servidor. Esto, claramente, tiene sus ventajas y sus inconvenientes dependiendo del uso que se le quiera dar.

Sin embargo descartamos JSweet por varias razones. En primer lugar porque su traducción de Java a JavaScript es muy literal y no es lo que buscábamos ya que de esa forma perderíamos mucha información. No reconoce las librerías utilizadas en Thoth. Nos pareció más útil utilizar un cliente servidor en GWT en el cual se pudiera comunicar vía RPC el cálculo del parser. Es decir, con JSweet deberíamos haber reescrito el parser de una forma muchísimo más compleja para que JSweet lo pudiera interpretar y eso hubiera llevado mucho más tiempo. Está mucho más centrado en el desarrollo en JavaScript en vez de en Java, lo que no nos serviría para poder aprovechar el trabajo de Thoth ya hecho. Recordemos que Thoth esta compuesto por dos versiones, hechas en grupos de dos alumnos y rehacer la aplicación completa en JavaScript sería una tarea muy costosa.

GWT en comparación, cuenta con una comunidad y un soporte bastante más completo, y pensado para desarrolladores Java, con una gran documentación acerca de su uso e implementación.

4.4. DukeScript

Se define como una tecnología para la creación de aplicaciones Java «multi-plataforma» que internamente hacen uso de tecnologías HTML5 y JavaScript para el renderizado⁶. Al igual que en los casos anteriores «sólo» se necesita desarrollar la aplicación en Java para después transformarla. Y digo «sólo» porque eso es en la teoría ya que como hemos podido ver, y en parte es lógico, la traducción suele requerir, por lo menos, realizar ajustes del lenguaje para un buen funcionamiento.

DukeScript se centra sobre todo en el desarrollo más que en el paso de Java a JavaScript. Da la posibilidad de que alguien con conocimientos, digámoslo así, en Java pueda llevar a cabo un proyecto en lenguajes pensados para aplicaciones móviles o web. Esto no quita que se puedan realizar aplicaciones de escritorio con JavaScript.

⁵<http://http://www.jsweet.org/>

⁶<https://dukescript.com/>

4.5. Vaadin

Vaadin es un *framework* de Java de código abierto, para crear aplicaciones web⁷. Se programa en Java o cualquier otro lenguaje de JVM o Java Virtual Machine. Lo más destacado de Vaadin es que está construido sobre una base de GWT, por ello es una de las grandes alternativas a este último. La forma de trabajar con Vaadin es mediante el lenguaje Java e incorpora un lado cliente y otro servidor, en el cual irán las funcionalidades más complejas y su programación es dirigida por eventos. Es decir, hasta aquí es igual a GWT.

Las mejoras con respecto a GWT son varias, pero voy a mencionar solo aquellas que son más relevantes para este proyecto. Cuenta sobre todo con muchos elementos visuales, mejorados y con diseños más actuales. La parte visual es tan importante en Vaadin que incluyen un «diseñador»⁸ o *designer* en inglés, como complemento de Eclipse que facilita mucho la creación de la parte visual, ya que da la posibilidad de hacer el diseño de forma visual.

En realidad, la mayor parte de los elementos visuales, menús, *boards*, diagramas estadísticos, iconos etc, están pensados sobre todo para un uso comercial orientado sobre todo para empresas. Por ello, el problema principal es que para poder hacer uso de su potencial se necesitan licencias de pago.

Aunque cuenta con un núcleo de elementos gratuitos y periodos de prueba también gratuitos, decidimos seguir nuestro camino, por llamarlo así, con GWT y hacerlo completamente de esta forma.

Las diferencias con GWT radican sobre todo en el lado de ejecución de la mayor parte de la aplicación. Con GWT, la mayor parte de las operaciones se encuentran en el lado del cliente, mientras que en Vaadin, el lado del cliente se comunica casi con cada interacción con el servidor. Como consecuencia, GWT es más rápido y se percibe sobre todo en condiciones en las que la conexión es mala o está sobrecargada. Por otro lado, Vaadin al ejecutarse sobre todo en el servidor, las operaciones son más seguras, ya que se encuentran más ocultas de cara al usuario. Además, al ser las operaciones en el servidor puede usar cualquier librería estándar de Java, GWT sin embargo, al estar casi todo en el cliente, debe limitarse a unas pocas librerías⁹.

Para aclarar un poco las diferencias que hay entre Vaadin y GWT utilizaré una tabla comparativa 4.1.

⁷<https://vaadin.com/home>

⁸<https://vaadin.com/designer>

⁹Véase <http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsCompatibility.html>

Características	Vaadin	GWT
Lenguaje	Cualquier JVM	Java con limitaciones de librerías en la parte del cliente.
Ejecución	Cliente o Servidor indistintamente	Se debe diferenciar que partes del código van en el cliente y cuales en el servidor. Esto depende de lo que admita el cliente.
Componentes gráficos	Librería UI con componentes modernos y mucha variedad	Librería algo anticuada y con fallos.
Herramientas de diseño GUI	Incluye <i>Designer</i>	Manual. Existen herramientas externas de uso comercial con coste.
Tipo de licencia	Comercial para la mayoría de los componentes	Gratuita

Tabla 4.1: Tabla comparativa entre Vaadin y GWT.

4.6. Herramientas para el cifrado de contraseñas

En este programa utilizamos una técnica muy simple, que mejora un poco este aspecto. Sabemos que existen técnicas más avanzadas que lo que aumentan mucho los tiempos de procesado en los ataques por fuerza bruta, por ejemplo. Pero no queríamos centrarnos mucho en ese tema además de que percibimos un ralentizado a la hora de registrar a un usuario ya que cifrar la contraseña requiere un mayor tiempo de ejecución.

La técnica que empleamos se conoce como el cifrado hash con sal o semilla (según su traducción del inglés, *hashing with salt*). Consiste en añadir un conjunto de caracteres aleatorio y concatenarlo a la contraseña y una vez hecho esto, cifrarlo con la función *hash*. De esta manera se consigue que la contraseña sea mucho más aleatoria que la que inicialmente ha introducido el usuario. Además de forma transparente para él.

Para generar el *salt* en Java contamos con el paquete «security» y la clase «SecureRandom»¹⁰ con la que nos aseguramos de crear una cadena de bytes aleatoria. Suficientemente aleatoria para nuestro propósito. Podemos especi-

¹⁰<https://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html>

ficar el tamaño en bytes del salt. Para almacenarlo, lo pasamos a formato «String».

Por otro lado, para hacer la autenticación del usuario se debe almacenar en la base de datos tanto el *salt* como el *hash* generado. Cuando el usuario introduce la contraseña para iniciar sesión, internamente se añade el salt a la contraseña introducida, se cifra con el mismo método y se comparan. Si coinciden, accederá a la aplicación, sino, deberá introducir de nuevo los datos necesarios.

4.7. Plataforma *Google Cloud*

La aplicación cuenta con una base de datos en la que se guarda el registro de los usuarios. A la base de datos se puede acceder desde la plataforma *Google Cloud* con una cuenta creada para ese propósito.

Esta plataforma sirve para alojar aplicaciones realizadas con alguna de las herramientas de desarrollo web de Google. Se ofrecen varios servicios interesantes, como una base de datos SQL, necesaria para hacer el registro e inicio de sesión.

De esta forma, gracias a la SDK de App Engine podemos gestionar una pequeña base de datos con el *datastore* de Google. Se trabaja con entidades y para este proyecto se han creado 3 tipos de entidades.

- **User.** En ella se almacenan los usuarios registrados con un nombre, *email*, fecha de registro y contraseña (*password*). Además, como ya hemos hablado en la sección Herramientas para el cifrado de contraseñas, también una *salt* para hacer el inicio de sesión (ver figura 4.4).
- **UserAction.** Se muestra la fecha de la última sesión de cada usuario. Con esta entidad pretendemos ver que usuarios son más activos, cuando se ha utilizado etc. Es cierto que esto se puede analizar desde el administrador de *Google Cloud*, pero decidimos introducir esta entidad antes de alojarla ahí.
- **GrammarUsed.** Entidad que sirve para almacenar las gramáticas comprobadas por un usuario.

La aplicación se puede gestionar fácilmente desde esta plataforma, el problema principal es que es de pago, pero cuenta con una versión gratuita de un año completo, con un gasto máximo de 300 euros. Esto quiere decir que puedo utilizar elementos con coste adicional hasta llegar al cupo de los 300 euros.

Nombre/ID	Email	Name	Password	RegisterDate	Salt	UserID
<input type="checkbox"/> nombre-admin@ubu.es	admin@ubu.es	admin	19cb5251beebd15ee9b4d6cf6e1d199a	8:48 27/5/2017	tvVp9jUlgYCuAMh1aMwh1QJPClgr-	admin@ubu.es
<input type="checkbox"/> nombre-javier@ubu.es	javier@ubu.es	Javier Paramo	47eef5258b86c7ee3be126458ac565224	23:13 26/5/2017	wj0DG3gwyfIPs2OoglnB3H1PN+	javier@ubu.es

Figura 4.4: Base de datos en *Google Cloud*. Se puede apreciar la entidad «User» que muestra la base de datos.

Aspectos relevantes del desarrollo del proyecto

Lo más relevante de este proyecto es el transformar la última versión de Thoth a una aplicación web hecha por medio de GWT lo que supone comprender el funcionamiento interno del Thoth original, «desglosándolo» para poder adaptarlo a las condiciones de un proyecto hecho con GWT. Estas condiciones limitan un poco la aplicación y nos obligan a reformar partes que antes eran más sencillas.

Además de condicionar Thoth a GWT, también hemos añadido nuevas funcionalidades de las que hablaremos y que completan la aplicación.

5.1. Internacionalización

La aplicación cuenta con la funcionalidad de la internacionalización. Dentro del menú se pueden elegir entre varios idiomas a los que se traducirán los diferentes elementos. Los idiomas en los que está disponible la versión web de Thoth son: alemán, castellano o español, francés y por supuesto inglés. Consideramos que esta funcionalidad es muy importante para poder llegar a diferentes países en el caso de que fuera necesario.

La internacionalización de la aplicación es un poco diferente a la utilizada en la versión de escritorio de Thoth. En primer lugar es necesario incluir una interfaz con los métodos para la internacionalización y los mensajes «por defecto» asociados a cada uno. Cada vez que queramos hacer uso de esos mensajes hay que hacer una llamada al método de la interfaz. Esa interfaz se encuentra en el directorio «client.gui.utils» donde se encuentran también los ficheros «properties» asociados, donde se encuentran las diferentes traducciones según el mensaje. Estos mensajes son los mismos que los utilizados para la internacionalización de Thoth V2.

Para poder realizar el cambio de idioma es necesario hacer uso de las propiedades las clases «xml» y «html» de GWT, en concreto «locale» que es la que especificará la localidad, que determina el idioma. Por ello cada vez que elegimos un idioma, la aplicación se redirige a una nueva «URL» (llevando a cabo una nueva compilación) con el atributo `locale=` seguido de las siglas del idioma al que se quiere traducir.

5.2. Google Web Toolkit

Para trabajar con GWT hemos utilizado el entorno de desarrollo de eclipse y con ayuda del *plugin* de Google, se crea un proyecto. Al crear un proyecto con él, se crean unos paquetes por defecto:

- Client: las clases dentro de este paquete se traducirán a lenguaje JavaScript. Es la parte fundamental de un proyecto en GWT y en ella se encuentra la clase principal que implementa «EntryPoint» (indicativo de que es la clase en la que se encuentra la función principal) y la función principal «onModuleLoad». Todo proyecto tiene al menos una clase en el cliente con estas características.
- Server: dentro, las clases no se traducirán a JavaScript y se puede programar en Java de forma completamente normal. Para realizar la comunicación se utilizan llamadas de procedimiento remoto (Remote Procedure Calls). Incluye servelets, bases de datos etc.
- Shared: mezcla de las dos anteriores. Se traduce también a JavaScript y puede ser utilizada en ambos lados de la aplicación. Suelen ser clases de ayuda para las acciones de un lado al otro de la aplicación.

Módulos en GWT

Las unidades individuales de configuración en GWT se denominan módulos. Un módulo agrupa todos los parámetros de configuración que necesita su proyecto GWT:

- Módulos heredados.
- Un nombre de clase como punto de entrada. Estos son opcionales, aunque cualquier módulo referido en HTML debe tener al menos una clase de punto de entrada especificada.
- Entradas de ruta de origen
- Entradas de ruta pública

- Reglas de vinculación diferida, incluyendo proveedores de propiedad y generadores de clase

Los módulos están definidos en archivos con extensión «.gwt.xml». En una aplicación con varios módulos como la nuestra, hay dos formas de enfocarlo: compilando cada módulo por separado o creando una definición XML de módulo de nivel superior que incluya todos los módulos y compilar el módulo de nivel superior.

Comunicación RPC

Como ya hemos hablado, RPC se utiliza para transferir objetos Java entre el cliente y el servidor a través de HTTP. Cada vez que se transfieran objetos a través de RPC, deben ser serializados. Los pasos para establecer la comunicación consisten en crear lo siguiente:

Hay que tener en cuenta que por defecto se crea un archivo dentro del directorio «War» denominado «web.xml». En él se configura el lugar donde se alojan los servlets, que contienen la implementación del servicio.

Aspectos del diseño con GWT

El diseño en GWT es uno de los apartados donde más se ha notado lo anticuado que está. Los elementos o widgets tienen diseños algo desactualizados. En realidad, esto no es un problema, ya que por medio de estilos «css» podemos configurar el diseño que queramos, pero es una clara desventaja en comparación con otros frameworks.

En nuestro caso, hemos hecho uso de un tema que proporciona GWT. Sirve para que al incorporarlo a un módulo con la sentencia «*inherits*», da un mínimo de aspecto a los elementos que se incorporen. Es el caso del módulo «GramaticaCS» que incorpora un tema denominado «Clean». Como se puede apreciar en la siguiente imagen 5.5, el estilo Clean otorga unos valores al elemento del menú, pero yo los cambio por medio de mi estilo «GrmaticaCS.css». Digamos que los sobrescribo. En este caso no utilizo por tanto Clean pero para otros elementos sí.

Es importante a la hora de diseñar, ya que la jerarquía, se aplica de más concreto a más amplio. Así si un botón está, por ejemplo, en un panel vertical y el panel vertical tiene un formato de texto *stilo1* y el botón tiene otro denominado *stilo2*, se aplicará el *stilo2*. En el caso de haber un tema heredado por un módulo, hay que tener en cuenta esto para ver por qué no se ha aplicado el *stilo1* a mi botón cuando yo le he dado uno al panel que lo engloba. Por defecto, al crear una aplicación con GWT, nuestro módulo hereda el estilo Standard.

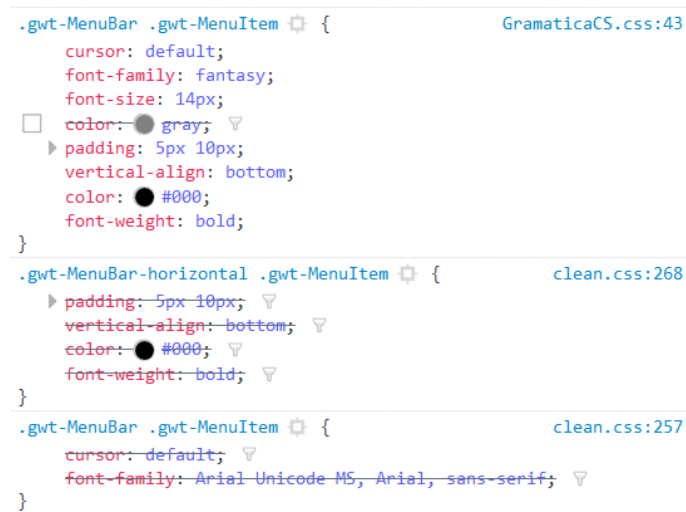


Figura 5.5: Vista de los estilos aplicados sobre un botón del menú desde el navegador.

Mencionar también que algunos elementos tienen fallos que no están resueltos y como hemos visto en foros, llevan tiempo sin resolver como es el caso del `TabPanel` o el `TabLayoutPanel`. Al tratar de crear «pestañas» e insertar en ellas un panel con ayuda de la función `insert`, la vista fallaba y al añadir una pestaña nueva, borraba la anterior. Estuvimos intentándolo durante dos *sprints* pero hasta que vimos que no se podía hacer de esa forma.

5.3. Registro e inicio de sesión de usuarios

En un primer momento, la aplicación se ejecutó en modo local desde Eclipse, de forma que para ello, como se explica en los anexos, necesitábamos hacer uso de la ejecución en modo *Super Dev*. La base de datos era local y se podía acceder a ella añadiendo a la url local «`_ah/admin`».

Más tarde, una vez se consiguió hacer todo en modo local, la aplicación se despliega en App Engine, así como todo lo que concierne a esta, es decir, base de datos, comunicación RPC, inicio de sesión etc. Todo esto se explica con más detalle en el Anexo del manual de programador.

Sin duda la parte más importante en el registro e inicio de sesión está en el servidor. Al hacer el registro tuvimos que aprender de forma detallada la relación que hay entre los módulos y el servidor. Para ello hay que meditar sobre cómo debe ser la arquitectura de acuerdo a nuestro objetivo. En esta aplicación, tenemos un módulo sólo para el registro e inicio de sesión, que se conecta a un servicio en el *server*. Este servicio debe apoyarse en otro, que

controlará la sesión para acceder al otro módulo que es el que contiene la gramática.

Realizar esto es una tarea ardua ya que para probar los cambio debíamos recompilar el proyecto cada vez, ejecutarlo en el navegador y comprobar los cambios añadiendo salidas por pantalla o mensajes de error.

Con todo esto he podido aprender la forma de trabajar desde un lado, digamos, más común como sería el cliente, y comunicarlo con el servidor, que necesita ayudarse de interfaces y clases dedicadas únicamente a la comunicación entre ambos.

El despliegue también ha sido una parte importante, que personalmente me llevó tiempo, pero que al final logramos encontrar y estamos bastante contentos con ello.

Trabajos relacionados

En este apartado vamos a ver otros proyectos, estudios o trabajos que están relacionados con este trabajo. Explicaremos en qué consisten y qué tienen que ver con este proyecto.

- **Noam**: Es una biblioteca JavaScript para trabajar con autómatas y gramáticas formales para lenguajes regulares y sin contexto.
- **JSFLAP**¹¹: Se trata de una aplicación web en desarrollo, programada en JavaScript, que sirve para visualizar lenguajes formales y teoría de los autómatas. Actualmente cuenta con una versión beta, aunque su desarrollo parece que se encuentra parado. Es un programa muy visual, permite exportar un autómata en L^AT_EX, en texto o en imagen o configurar el aspecto visual. Es muy útil y fácil de usar.
- **Compiler Construction Toolkit**¹²: Se trata de una herramienta web con la que podemos construir componentes de un compilador. Sus herramientas se pueden clasificar en
 - Herramientas de teoría de compiladores.
 - Herramientas de diseño de compiladores.
 - Herramientas para generar un parser.

Es algo más difícil de utilizar que JSFLAP, pero cuenta con un mayor número de herramientas. Genera código en lenguaje «Ruby».

- **BURGRAM**¹³: Consiste en un programa para la generación y simulación de tablas de análisis sintáctico, Dirigido por el Dr. César Ignacio García Osorio y desarrollado por Carlos Gómez Palacios.

¹¹<http://jsflap.com/>

¹²<http://hackingoff.com/compilers>

¹³<http://cgosorio.es/BURGRAM/>

Conclusiones y Líneas de trabajo futuras

En este apartado, mencionaremos algunas conclusiones a las que hemos llegado después de realizar el proyecto además de posibles ideas relacionadas con futuros proyectos.

7.1. Conclusiones

Si bien es verdad que la dirección tomada en un principio sobre qué herramienta utilizar para desarrollar el proyecto, podría haber sido más acertada, hemos aprendido mucho sobre el entorno de GWT. *WebSwing* nos hubiera facilitado mucho el trabajo de transformar a HTML5 el código en Java. Pero por otro lado, hemos visto la forma en la que se desarrollan cientos de aplicaciones con GWT, algunas tan grandes como Cloudorado¹⁴, BookedIN¹⁵, Gae-Studio¹⁶ o Sigmah¹⁷.

En el caso de que hubiéramos hecho el proyecto con herramientas más fáciles nos hubiéramos centrado sobre todo en añadir mejoras de Thoth, sin perder tanto tiempo en adaptar el código Java para GWT.

En cuanto a mi, en líneas generales, he podido aprender mucho sobre el desarrollo web, la arquitectura que utiliza y como se distribuye el programa entre el cliente y el servidor o la forma en la que se despliega una aplicación de estas características.

¹⁴<https://www.cloudorado.com/>

¹⁵<https://bookedin.com/>

¹⁶<https://dev.arcbees.com/gaestudio/>

¹⁷<http://www.sigmah.org/EN.html>

7.2. Líneas de trabajo futuras

Como posibles mejoras sobre este proyecto, hemos encontrado varios detalles que se nos han escapado, ya sea por falta de tiempo, o por no habernos dado cuenta antes.

- Creemos que es interesante almacenar en la base de datos la configuración del usuario, para que una vez que cambie el idioma de la aplicación, por defecto, al iniciar sesión lo haga con el idioma elegido anteriormente.
- Añadir algún método para guardar y cargar gramáticas, ya sea de forma local, en ordenador del usuario, o en la base de datos.
- poder añadir comentarios al introducir un gramática, de forma que no se tengan en cuenta a la hora de comprobar la gramática. Nos dimos cuenta tarde, al hacer varias pruebas finales.
- Este proyecto se podría haber hecho de forma parecida pero con *WebS-wing* y centrándose sobre todo en la base de datos, en la accesibilidad desde distintos dispositivos etc.
- Crear perfiles de usuario, en que pueda configurar un avatar, añadir gramáticas favoritas entre otros.
- Añadir un panel de construcción de autómatas como en otras versiones de Thoth.

Bibliografía

- [1] Alfred V Aho, Ravi Sethi, and Jeffrey D Ullman. *Compilers, Principles, Techniques*. Addison wesley Boston, 1986.
- [2] Álgvar Arnaiz-González, José Francisco Díez-Pastor, Ismael Ramos-Pérez, and César García-Osorio. Herramienta docente en línea para facilitar la enseñanza de teoría de autómatas. *Jornadas Universitarias de Tecnología Educativa*, 2017.
- [3] César García-Osorio, Andrés Arnaiz-Moreno, and Álgvar Arnaiz-González. Enseñanza asistida de teoría de autómatas y lenguajes formales mediante el uso de thoth, 2007.
- [4] Ecma International. Ecma international , dedicated to the standardization of information and communication systems., 2017.