



Trabajo de Fin de Máster Sistemas y Tecnologías Web Aplicadas (SyTWA)

Shell para corrección automática de repositorios de GitHub

*CLI tool for automatic correction of GitHub's
repositories .*

Juan José Labrador González

La Laguna, 1 de julio de 2017

D. **Casiano Rodríguez León**, con DNI número 42.020.072-S profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

“Sistemas y Tecnologías Web Aplicadas. Shell para corrección automática de repositorios de GitHub.”

ha sido realizada bajo su dirección por D. **Juan José Labrador González**, con DNI número 78.729.778-L.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 1 de julio de 2017

Agradecimientos

La realización de esta asignatura de Trabajo de Fin de Máster no hubiera sido posible sin la ayuda de la Sección de Ingeniería Informática de la Escuela Superior de Ingeniería y Tecnología, que ha llevado a cabo todos los trámites necesarios.

Mención especial para mi familia, pareja y amigos, quienes me han alentado para no rendirme y lograr mis objetivos pese a las dificultades y contratiempos encontrados durante la realización de este Trabajo de Fin de Máster.

Y por último, especialmente agradecer a Casiano Rodríguez León su labor como tutor del Trabajo de Fin de Máster. Además de aprender muchísimo junto a él, me ha aconsejado, animado y resuelto mis dudas de manera incansable en la realización de este trabajo. Estoy seguro de que la experiencia y conocimientos adquiridos gracias a él, me ayudarán en mis próximos retos profesionales y personales.

Licencia



© Esta obra está bajo una licencia de Creative Commons
Reconocimiento-NoComercial-CompartirIgual 4.0
Internacional.

Resumen

El objetivo de este Trabajo de Fin de Máster ha sido integrar los conocimientos adquiridos durante los estudios del Máster y, en especial, del itinerario de Tecnologías de la Información, aproximando al alumno a la resolución de problemas de aplicaciones Web y favoreciendo el desarrollo de destrezas propias de la Ingeniería Web: se centra en el aprendizaje y puesta en práctica de metodologías, aproximaciones, técnicas y herramientas para abordar la creciente complejidad de este tipo de aplicaciones en el marco de las metodologías ágiles. Cada vez ésta cobra más importancia, siendo constante el aumento del número de aplicaciones de escritorio, smartphones y tablets.

En este Trabajo de Fin de Máster se propone el desarrollo de un paquete Node.js (NPM) que facilite la descarga y corrección de repositorios GitHub de alumnos. Existe un buen número de herramientas de Control de Versiones que permiten alojar proyectos software y agruparlos en organizaciones lógicas, pero carecen de mecanismos para automatizar funciones de uso cotidiano como la descarga de los mismos, la preparación del entorno de cada proyecto o la ejecución de pruebas.

En nuestra propuesta, se ha realizado una primera aproximación a la automatización de descargas y correcciones de repositorios, recopilando todos los datos inherentes de estas acciones y generando los informes correspondientes en formato PDF y HTML. Todo ello mediante un sencillo uso y sentando las bases para proporcionar más funcionalidades a la herramienta en un futuro próximo.

Palabras clave: Consola, CLI, Shell, Terminal, Node.js, GitHub, Corrección, Automatización

Abstract

Insert here an abstract in an EU language (preferably English)

Keywords: *Console, CLI, Shell, Terminal, Node.js, GitHub, Correction, Automation*

Índice general

1. Introducción	1
1.1. Antecedentes	1
1.2. Estado actual del arte	1
1.3. Objetivos y actividades a realizar	2
1.4. Tecnología usada	3
2. Desarrollo	4
2.1. Metodología usada	4
2.1.1. GitHub	4
2.1.2. Travis-CI	7
2.1.3. Experiencia de usuario	7
3. Resultados	9
3.1. Funcionalidades requeridas	9
3.1.1. Autenticación con GitHub	9
3.1.2. Listar organizaciones, asignaciones y repositorios de GitHub del usuario	12
3.1.3. Automatizar la descarga de repositorios	13
3.1.4. Automatizar la ejecución de scripts en los repositorios . .	14
3.1.5. Recopilar la información obtenida de la automatización de tareas	15
3.2. Funcionalidades extra	17
3.3. Problemas encontrados y soluciones	17
3.3.1. Asincronía	17
3.3.2. Autocompletado de comandos	18
3.4. Perfil del usuario de ghshell	18
4. Conclusiones y líneas futuras	19
5. Summary and Conclusions	21
5.1. First Section	21
6. Presupuesto	22
6.1. Introducción y coste por hora	22

6.2. Funcionalidades requeridas	23
6.3. Funcionalidades extra	23
6.4. Coste y duración total	24
A. Glosario	25
B. Guía de uso	30
B.1. Instalación	30
B.1.1. Requisitos	30
B.1.2. Dependencias	30
B.1.3. Instalación	30
B.2. Ejecución	30
B.2.1. Otras consideraciones	30
Índice alfabético	31
Bibliografía	32

Índice de Figuras

2.1. Captura del repositorio del paquete NPM en GitHub	5
2.2. Ramas del repositorio	6
2.3. Apartado de issues	7
2.4. Herramienta de integración continua	7
3.1. Página del gestor de paquetes NPM	9
3.2. Login de usuario	10
3.3. Usuario autenticado	10
3.4. Token personal en GitHub	11
3.5. Login automático una vez generado el token	11
3.6. Logout de usuario	11
3.7. Lista de organizaciones del usuario	12
3.8. Lista de repositorios de una organización	12
3.9. Asignaciones dentro de otra organización	12
3.10. Acceso a un repositorio de una organización	13
3.11. Clonado del repositorio donde nos encontramos	13
3.12. Clonado de asignaciones que coinciden con una expresión regular	13
3.13. Resultado del clonado	14
3.14. Ejecución del script 'install.sh' en el repositorio actual	14
3.15. Ejecución del script 'install.sh' en asignaciones que coinciden con una expresión regular	15
3.16. Resultado de la ejecución del script 'install.sh'	15
3.17. Creación del Gitbook en el repositorio actual	16
3.18. Creación del Gitbook en asignaciones que coinciden con una ex- presión regular	16
3.19. Resultado de la creación del Gitbook	16

Índice de Tablas

6.1. Tabla de actividades, duración y precios de las funcionalidades requeridas	23
6.2. Tabla de actividades, duración y precios de las funcionalidades extra	23
6.3. Precio y duración total	24

Capítulo 1

Introducción

1.1. Antecedentes

La World Wide Web está sujeta a un cambio continuo. La llegada de HTML5, la creciente importancia de AJAX y de la programación en el lado del cliente, las nuevas fronteras de la Web Semántica, y la explosión de las redes sociales son ejemplos de esta tendencia general.

Las aplicaciones web parecen evolucionar hacia entornos cada vez más ricos y flexibles en los que los usuarios pueden acceder con facilidad a los documentos, publicar contenido, escuchar música, ver vídeos, realizar dibujos e incluso jugar usando un navegador. Esta nueva clase de software ubicuo no cesa de ganar momentum y promueve nuevas formas de interacción y cooperación.

Ante la rápida evolución del software, los sistemas de control de versiones han adquirido una mayor importancia dentro de la metodología del desarrollo del software: la gestión de las versiones del propio software se ha convertido en una actividad crítica. Estos sistemas han evolucionado a la par que el software, proporcionando nuevas funcionalidades y orientándose hacia la colaboración.

1.2. Estado actual del arte

Actualmente, hay numerosos sistemas de control de versiones. Todos ellos proporcionan mecanismos de almacenamiento del código, de modificación y de consulta histórica del mismo, a la vez que proporcionan un entorno colaborativo en el que los usuarios pueden colaborar e interactuar entre sí.

En el caso particular de GitHub, además de proporcionar lo mencionado anteriormente, observando el creciente número de estudiantes que utiliza la plataforma, ha creado herramientas específicas para facilitar sus desarrollos (ej: Student Developer Pack) y provee a profesores de herramientas para gestionar dichos desarrollos (ej: GitHub Classrooms).

Sin embargo, estas herramientas de gestión de desarrollos requieren una administración interactiva por parte del profesor. No cuentan aún con funcionalidades de automatización de tareas.

1.3. Objetivos y actividades a realizar

En este proyecto se persigue integrar los conocimientos adquiridos durante los estudios del Máster y, en especial, del itinerario de Tecnologías de la Información para solucionar problemas actuales de aplicaciones y servicios Web.

Los objetivos propuestos para alcanzar en este Trabajo de Fin de Máster ha sido los siguientes:

- Conocer, dominar y practicar con lenguajes y herramientas de desarrollo de aplicaciones Web en el servidor (Node.js, npm).
- Conocer, practicar y dominar de herramientas de Desarrollo Dirigido por Pruebas en entornos web y la Integración Continua.
- Conocer, practicar y familiarizarse con diferentes mecanismos de seguridad, autenticación y autorización.
- Conocer, practicar y dominar diferentes herramientas colaborativas y de control de versiones.
- Conocer, practicar y dominar Metodologías Ágiles de desarrollo de software.

Y las actividades a realizar en el mismo son las que se describen a continuación:

- Revisión bibliográfica y estado del arte.
- Desarrollar una herramienta de línea de comandos escrita en Node.js que permita automatizar tareas relacionadas con repositorios de GitHub:
 - Autenticación con GitHub.
 - Listar organizaciones, asignaciones y repositorios de GitHub del usuario.
 - Automatizar la descarga de repositorios.
 - Automatizar la ejecución de scripts en los repositorios (TDD, creación de entorno, evaluación de código...).
 - Recopilar la información obtenida de la automatización de tareas y presentarla al usuario (PDF, HTML...).

- Redacción de la memoria.
- Preparación de la presentación oral.

1.4. Tecnología usada

Para llevar a cabo el desarrollo de esta herramienta se planteó realizar el desarrollo en **Node.js**, creando una librería modular que se pudiese instalar mediante el gestor de paquetes de Node.js (NPM).




Además, se ha hecho uso de otras tecnologías enumeradas a continuación:

- NPM 

- GitHub 

- GitBook  **GitBook**

- Travis-CI  **Travis CI**

Capítulo 2

Desarrollo

En el capítulo anterior se ha descrito el estado del arte actual y se ha definido el Trabajo de Fin de Máster, especificado los objetivos, actividades a desarrollar y las tecnologías empleadas para su desarrollo. A continuación, se describirá la metodología de trabajo seguida.

2.1. Metodología usada

Se ha llevado a cabo una metodología de trabajo ágil, común en el campo de la Ingeniería Informática, con reuniones periódicas en las que se definían una serie de tareas u objetivos (iteración) y que se presentaban la siguiente reunión. De este modo, con la entrega de prototipos funcionales de la aplicación, se han ido testeando, corrigiendo y mejorando las funcionalidades, al mismo tiempo que detectando problemas no contemplados en las fases previas de diseño.

Esta metodología, además, ha propiciado la generación de ideas que se han traducido en nuevas características.

2.1.1. GitHub

Para llevar a cabo esta metodología, se ha usado GitHub como herramienta de Control de Versiones (CVS). Todo el código implementado se alojaba en dicha herramienta, permitiendo así su cómoda modificación y actualización.

ULL-ESIT-GRADOII-TFG / ghshell

Unwatch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Trabajo de Fin de Máster - Sistemas y Tecnologías Web Aplicadas (SyTWA) Edit

nodejs npm npm-package cli shell github Manage topics

53 commits 4 branches 1 release 1 contributor MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Commit Message	Time Ago
lib	GitBook option implemented	2 days ago
scripts	Example script added	4 days ago
test	Tests in pending state by the moment	3 months ago
.gitignore	.gitignore updated	3 months ago
.nvmrc	Using NodeJS v8.1.2	4 days ago
.travis.yml	Node version edited in .travis.yml	2 days ago
LICENSE	Initial commit	3 months ago
README.md	Fixed Travis-CI Badge	a day ago
index.js	GitBook option implemented	2 days ago
package-lock.json	README.md updated	2 days ago
package.json	Set Node engine to >= 8.0.0	2 days ago

Figura 2.1: Captura del repositorio del paquete NPM en GitHub

El trabajo se dividía en ramas, de modo que la versión estable de la aplicación (rama **master**) quedara aislada de la versión en desarrollo (rama **develop**) y de la rama experimental (rama **test**).

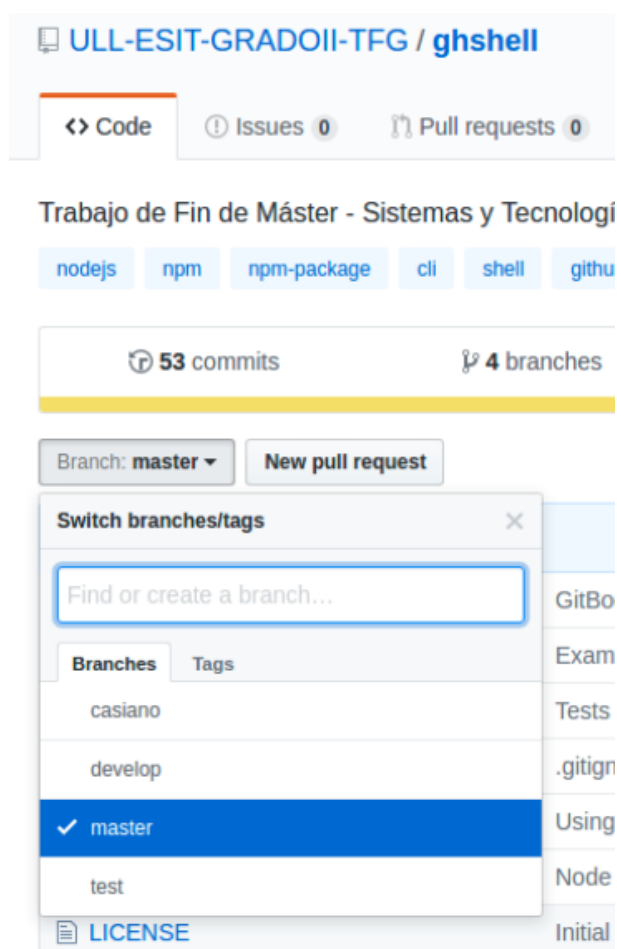


Figura 2.2: Ramas del repositorio

La documentación adicional para llevar a cabo los desarrollos de cada iteración, así como los problemas detectados, se anotaban en el apartado de **issues** con el fin de que quedara constancia de ello y se reflejara el estado en el que se encontraba cada uno.

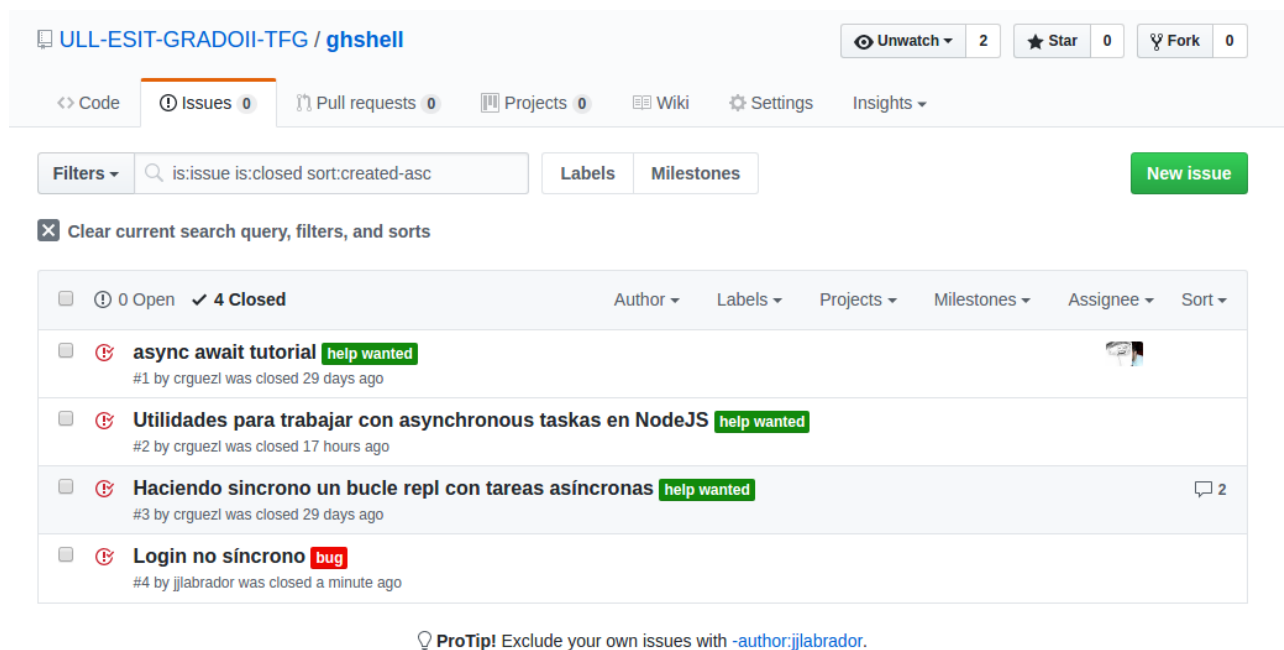


Figura 2.3: Apartado de issues

2.1.2. Travis-CI

Como herramienta de integración continua, se ha utilizado Travis-CI, con el fin de asegurarnos el despliegue de la aplicación era satisfactorio tras cada cambio subido a la herramienta de control de versiones (GitHub).

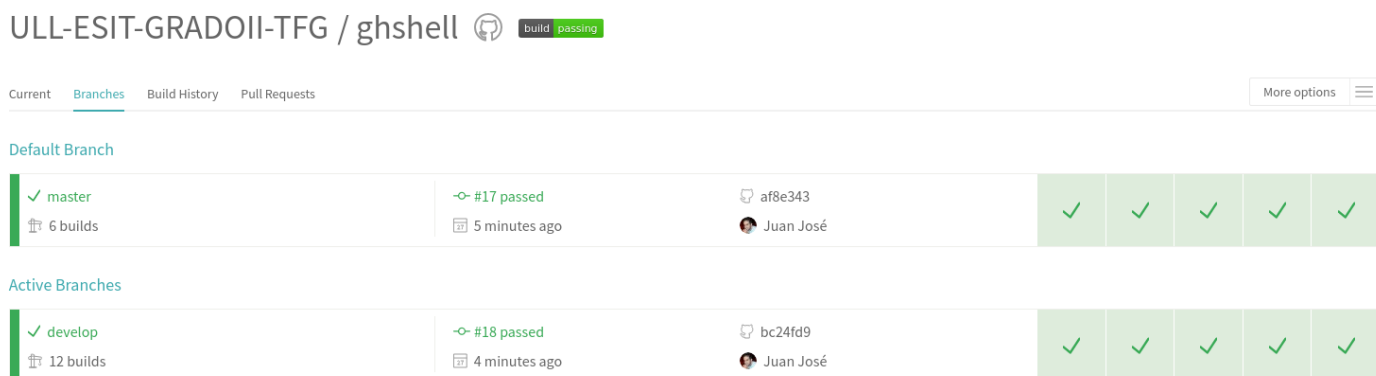


Figura 2.4: Herramienta de integración continua

2.1.3. Experiencia de usuario

Por otra parte, el tutor del Trabajo de Fin de Máster ha hecho pruebas reales con el resultado de cada iteración, actuando como *Product Owner*. De este modo, se comprobaba el funcionamiento de la aplicación en un entorno real

y se recibía un valioso feedback para corregir problemas o hacer mejoras en las siguientes iteraciones.

Capítulo 3

Resultados

Finalizada la etapa de desarrollo del Trabajo de Fin de Máster, se procede a describir la herramienta implementada.

La herramienta se ha denominado **ghshell**, abreviatura de 'GitHub Shell'. Se ha publicado en NPM[1] para su fácil distribución e instalación:



Figura 3.1: Página del gestor de paquetes NPM

Las funcionalidades implementadas en **ghshell**, se describen a continuación.

3.1. Funcionalidades requeridas

3.1.1. Autenticación con GitHub

Una vez que el usuario se autentifica con GitHub, se genera un token personal, que se usa posteriormente para acceder a la API de Github. Este token se almacena cifrado en el equipo del usuario, por lo que las siguientes ocasiones que utilice la herramienta no hará falta que vuelva a iniciar sesión:

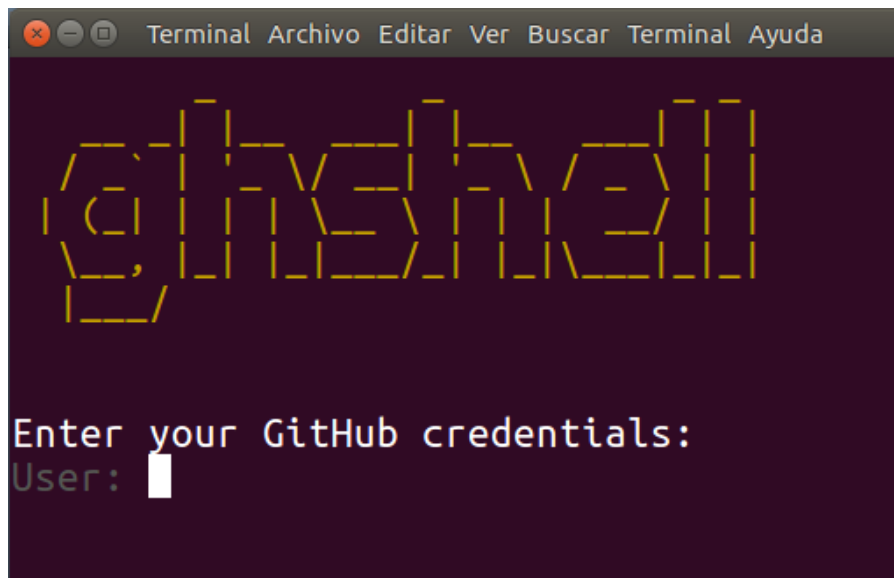


Figura 3.2: Login de usuario

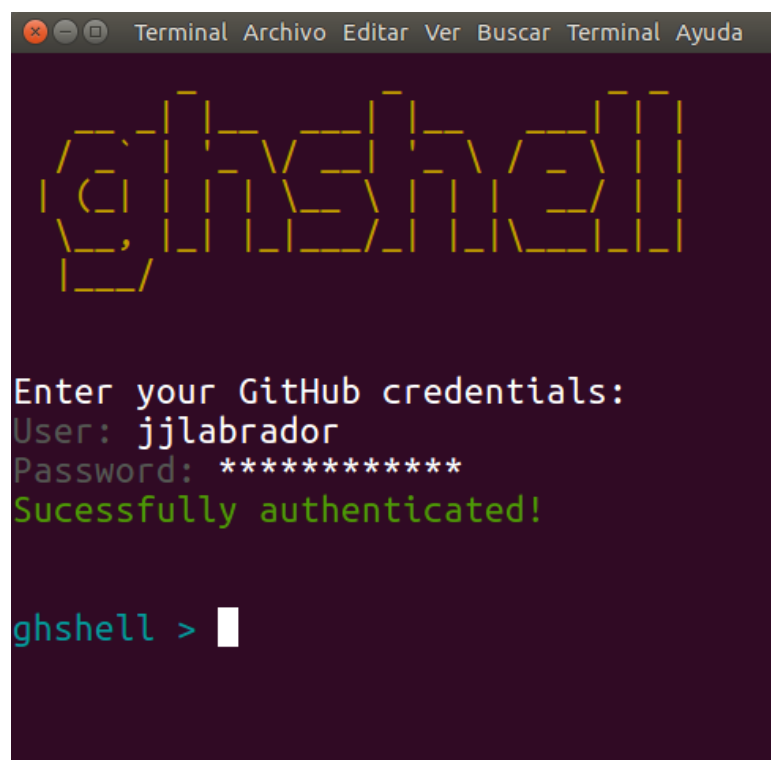


Figura 3.3: Usuario autenticado

Personal access tokens

Generate new token

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

ghshell, the CLI tool for automatic corrections and executions of GitHub's repositories Never used

Edit

Delete

— public_repo, read:org, read:user

Figura 3.4: Token personal en GitHub

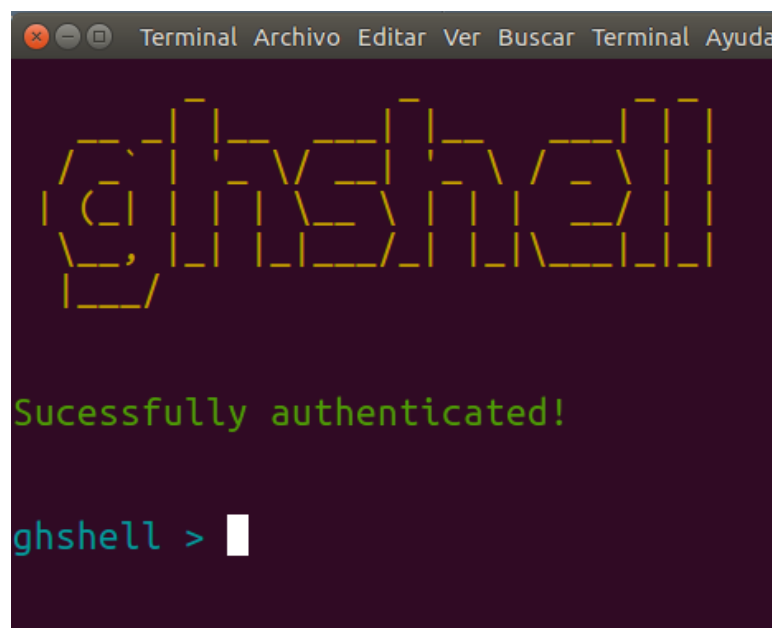


Figura 3.5: Login automático una vez generado el token

Si el usuario cierra sesión en la herramienta, se eliminará el token en GitHub y en el equipo:

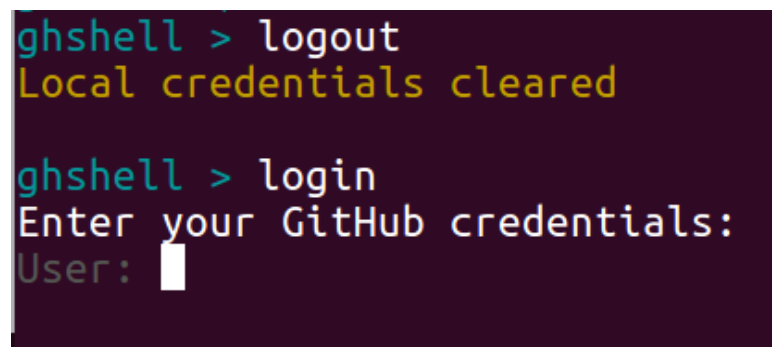


Figura 3.6: Logout de usuario

3.1.2. Listar organizaciones, asignaciones y repositorios de GitHub del usuario

Con el comando `orgs -l`, se puede listar las organizaciones del usuario y usando `repos -l`, se listarán los repositorios del usuario. También se puede acceder 'virtualmente' a las organizaciones y listar los repositorios que contiene, así como las asignaciones.

NOTA: se puede consultar toda la información referente a los comandos del programa en el Apéndice 2.

```
ghshell > orgs -l
DSI-ETSII-ULL    ULL-ESIT-GRADOII-TFG    ULL-ESIT-TFM-test-evaluation-shell
ghshell > █
```

Figura 3.7: Lista de organizaciones del usuario

```
ghshell > orgs
Select organization (left empty for cancel the action): ULL-ESIT-GRADOII-TFG
ghshell (ULL-ESIT-GRADOII-TFG) > repos -l
teachers_pet  plugin-exercises  ruql  ghedsh  gitbook-plugin-jazer  rege
xp-gbp  tott-gulpjs  rudolf-cicko-17  Memoria-TFG-Cicko  TFG-Eleazar-17
TFG-Memoria-Eleazar-17  ghshell
ghshell (ULL-ESIT-GRADOII-TFG) > █
```

Figura 3.8: Lista de repositorios de una organización

```
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > assignments /evaluar/
evaluar-rutas-jjlabrador
evaluar-rutas-crguezl
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > █
```

Figura 3.9: Asignaciones dentro de otra organización

Las asignaciones son un conjunto de repositorios que corresponden a una tarea asignada por los profesores a los alumnos.

También es posible acceder 'virtualmente' a los repositorios y realizar acciones sobre ellos:

```
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > repos
Select repository (left empty for cancel the action): evaluar-rutas-jjlabrador
ghshell (ULL-ESIT-TFM-test-evaluation-shell ~> evaluar-rutas-jjlabrador) > █
```

Figura 3.10: Acceso a un repositorio de una organización

3.1.3. Automatizar la descarga de repositorios

En función del contexto dónde nos encontremos dentro de la herramienta, podremos:

- Clonar el repositorio en el que nos encontremos.
- Clonar un repositorio determinado.
- Clonar todos los repositorios que coincidan con una determinada expresión regular.
- Clonar todos los repositorios de una asignación que coincidan con una determinada expresión

El clonado se realiza de manera asíncrona, por lo que podemos seguir trabajando mientras se clona(n) el/los repositorio(s). Se puede observar el estado de la clonación revisando el fichero de log que se genera cuyo nombre sigue el formato: `nombre-repositorio -clone.log`.

```
ghshell (ULL-ESIT-TFM-test-evaluation-shell ~> evaluar-rutas-jjlabrador) > clone
Cloning evaluar-rutas-jjlabrador... (see evaluar-rutas-jjlabrador-clone.log for more information)
ghshell (ULL-ESIT-TFM-test-evaluation-shell ~> evaluar-rutas-jjlabrador) > █
```

Figura 3.11: Clonado del repositorio donde nos encontramos

Si clonamos repositorios que pertenecen a una organización, se creará una carpeta con el nombre de la organización y en su interior se guardarán los repositorios clonados.

Además, si clonamos repositorios que pertenecen a una asignación, también se creará una carpeta con el nombre de la asignación que los contendrá.

```
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > assignments /evaluar/ clone
Cloning evaluar-rutas-jjlabrador... (see evaluar-rutas-jjlabrador-clone.log for more information)
Cloning evaluar-rutas-crguezl... (see evaluar-rutas-crguezl-clone.log for more information)
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > █
```

Figura 3.12: Clonado de asignaciones que coinciden con una expresión regular

```
juanjose@Sobremesa:/tmp/TFM$ tree -L 3
.
├── install.sh
├── ULL-ESIT-TFM-test-evaluation-shell
│   └── evaluar-rutas
│       ├── evaluar-rutas-crguezl
│       ├── evaluar-rutas-crguezl-clone.log
│       ├── evaluar-rutas-jjlabrador
│       └── evaluar-rutas-jjlabrador-clone.log
4 directories, 3 files
juanjose@Sobremesa:/tmp/TFM$
```

Figura 3.13: Resultado del clonado

3.1.4. Automatizar la ejecución de scripts en los repositorios

En función del contexto donde nos encontremos dentro de la herramienta, podremos:

- Ejecutar un script en el repositorio en el que nos encontremos.
- Ejecutar un script en un determinado repositorio.
- Ejecutar un script en todos los repositorios que coincidan con una determinada expresión regular.
- Ejecutar un script en todos los repositorios de una asignación coincidan con una determinada expresión regular.

La ruta del fichero del script puede ser absoluta o relativa. Estos scripts puede ser de cualquier tipo: TDD, creación de entorno, evaluación de código...

La ejecución de cada script se ejecuta en un proceso hijo independiente pero, a diferencia del clonado, el script se ejecuta línea a línea de manera síncrona. Se puede observar el estado de la ejecución del script y los resultados revisando el fichero de log que se genera: <nombre-repositorio >- <nombre-script >.log

```
ghshell (ULL-ESIT-TFM-test-evaluation-shell ~> evaluar-rutas-jjlabrador) > script install.sh
Execution of install.sh in evaluar-rutas-jjlabrador has finished
ghshell (ULL-ESIT-TFM-test-evaluation-shell ~> evaluar-rutas-jjlabrador) >
```

Figura 3.14: Ejecución del script 'install.sh' en el repositorio actual


```
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > assignments /evaluar/ script install.sh
Execution of install.sh in evaluar-rutas-jjlabrador has finished
Execution of install.sh in evaluar-rutas-crguezl has finished
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > █
```

Figura 3.15: Ejecución del script 'install.sh' en asignaciones que coinciden con una expresión regular

```
juanjose@Sobremesa:/tmp/TFM$ tree -L 3
.
├── install.sh
├── ULL-ESIT-TFM-test-evaluation-shell
│   └── evaluar-rutas
│       ├── evaluar-rutas-crguezl
│       ├── evaluar-rutas-crguezl-clone.log
│       ├── evaluar-rutas-crguezl-install.sh.log
│       ├── evaluar-rutas-jjlabrador
│       ├── evaluar-rutas-jjlabrador-clone.log
│       └── evaluar-rutas-jjlabrador-install.sh.log
4 directories, 5 files
juanjose@Sobremesa:/tmp/TFM$ █
```

Figura 3.16: Resultado de la ejecución del script 'install.sh'

3.1.5. Recopilar la información obtenida de la automatización de tareas

Una vez ejecutados los scripts necesarios para evaluar un determinado repositorio, es posible generar un GitBook con el resultado de la ejecución de los mismos. Este libro se genera en formato PDF y en HTML.

En función del contexto dónde nos encontremos dentro de la herramienta, podremos:

- Crear un GitBook en el repositorio en el que nos encontremos.
- Crear un GitBook en un determinado repositorio.
- Crear un GitBook en todos los repositorios que coincidan con una determinada expresión regular.
- Crear un GitBook en todos los repositorios de una asignación coincidan con una determinada expresión regular.

```
ghshell (ULL-ESIT-TFM-test-evaluation-shell ~> evaluar-rutas-jjlabrador) > book
Book evaluar-rutas-jjlabrador created successfully!
Book evaluar-rutas-jjlabrador exported to PDF successfully!

ghshell (ULL-ESIT-TFM-test-evaluation-shell ~> evaluar-rutas-jjlabrador) > █
```

Figura 3.17: Creación del Gitbook en el repositorio actual

```
ghshell (ULL-ESIT-TFM-test-evaluation-shell) > assignments /evaluar/ book
Book evaluar-rutas-jjlabrador created successfully!
Book evaluar-rutas-jjlabrador exported to PDF successfully!
Book evaluar-rutas-crguezl created successfully!
Book evaluar-rutas-crguezl exported to PDF successfully!

ghshell (ULL-ESIT-TFM-test-evaluation-shell) > █
```

Figura 3.18: Creación del Gitbook en asignaciones que coinciden con una expresión regular

```
juanjose@Sobremesa:/tmp/TFM$ tree -L 3
.
├── install.sh
├── ULL-ESIT-TFM-test-evaluation-shell
│   └── evaluar-rutas
│       ├── evaluar-rutas-crguezl
│       ├── evaluar-rutas-crguezl-clone.log
│       ├── evaluar-rutas-crguezl_gitbook
│       ├── evaluar-rutas-crguezl-gitbook_build.out
│       ├── evaluar-rutas-crguezl-gitbook_pdf.out
│       ├── evaluar-rutas-crguezl-install.sh.log
│       ├── evaluar-rutas-crguezl.pdf
│       ├── evaluar-rutas-jjlabrador
│       ├── evaluar-rutas-jjlabrador-clone.log
│       ├── evaluar-rutas-jjlabrador_gitbook
│       ├── evaluar-rutas-jjlabrador-gitbook_build.out
│       ├── evaluar-rutas-jjlabrador-gitbook_pdf.out
│       ├── evaluar-rutas-jjlabrador-install.sh.log
│       └── evaluar-rutas-jjlabrador.pdf
6 directories, 11 files
juanjose@Sobremesa:/tmp/TFM$ █
```

Figura 3.19: Resultado de la creación del Gitbook

3.2. Funcionalidades extra

Además de las funcionales solicitadas en este Trabajo de Fin de Máster, se han añadido una serie de funcionalidades extra que, a pesar de no ser requeridas, brindan al usuario de una mejor experiencia de uso del programa:

- Autocompletado de los comandos disponibles en función del contexto donde nos encontremos (nivel principal, organización o repositorio).
- Opción de ayuda que muestra la descripción de los comandos y cómo se utilizan. Esta ayuda varía dependiendo del contexto donde nos encontremos.
- Opción de visualizar el directorio de trabajo donde se ha ejecutado el programa. Útil para determinar rutas relativas de los scripts que se desean ejecutar.
- Opción para conocer el propietario de cada repositorio. En el caso de que el repositorio pertenezca a una organización, mostrará los contribuyentes de ese repositorio.

NOTA: se puede consultar toda la información referente a los comandos del programa en el Apéndice 2.

3.3. Problemas encontrados y soluciones

A continuación se detallan los problemas encontrados durante la implementación de la herramienta y las soluciones encontradas para los mismos:

3.3.1. Asincronía

Una de las características más importantes del lenguaje Node.js es la asincronía. Usa un modelo de operaciones de entrada/salida sin bloqueo y orientado a eventos, que lo hace ligero y eficiente. Sin embargo, algunas acciones que debía realizar esta herramienta debían de ser síncronas. Ej: login del usuario y ejecución de scripts.

Solución

La solución a este comportamiento pasó por realizar un amplio estudio de la documentación para usar mecanismos que permitieran bloquear la ejecución de la herramienta en las partes que deseábamos. Los mecanismos usados han sido:

- Funciones síncronas del propio lenguaje.

- Promesas
- Métodos `async/await`
- Librerías con métodos implementados de manera síncrona.

3.3.2. Autocompletado de comandos

Para el manejo de los flujos de lectura y escritura de la herramienta, se ha utilizado la interfaz nativa de Node.js (Readline). Esta interfaz provee de una función de autocompletado para el texto que escribe el usuario.

Sin embargo, sólo funciona con la primera palabra (comando) que escribe. Tras investigar al respecto y buscar posibles librerías alternativas, no existía ninguna solución que corrigiera este comportamiento.

Solución

Realizando numerosas pruebas, se halló una manera propia de conseguir completar más de un comando en la misma línea. Cuando realice los test de aceptación pertinentes requeridos por la comunidad de Node, solicitaré un Pull Request a su repositorio con esta mejora.

3.4. Perfil del usuario de ghshell

El uso de ghshell está especialmente dirigido a un determinado grupo de profesores: nos referimos al perfil de un profesor, principalmente docente en alguna rama de Ingeniería, con conocimientos avanzados en programación y en herramientas de control de versiones.

No obstante, ya que la curva de aprendizaje de ghshell no es excesiva y dado que el uso de las herramientas de control de versiones no se limita exclusivamente a repositorios de código fuente, se puede extender su uso para el resto de profesorado y usuarios con otros roles. Basta con tener claras unas nociones básicas de informática, junto con la lectura y asimilación previa de la documentación de la herramienta.

Capítulo 4

Conclusiones y líneas futuras

Desde hace unos años hasta ahora, ha tenido lugar un enorme crecimiento de las herramientas de control de versiones. Se han convertido en una herramienta imprescindible en la metodologías de desarrollo del software y las instituciones de enseñanza saben que incorporarlas a sus sistemas educativos es clave para ofrecer un servicio puntero y de calidad.

Ésto es lo que se pretende con la herramienta obtenida tras la realización de este Trabajo de Fin de Máster: que sea posible su implantación dentro del marco académico de la Universidad de La Laguna, partiendo de la premisa de que, actualmente, el desarrollo de un proyecto software sin tener detrás un sistema de control de versiones, no es viable.

La automatización de las tareas de clonado y ejecución de pruebas facilitaría al profesor, en primera instancia, la corrección de las prácticas y proyectos de los alumnos. El ahorro de tiempo de ejecutar estas tareas manualmente es considerable, teniendo en cuenta el número de prácticas que realiza cada alumno por asignatura. Esta enorme carga de trabajo del profesor puede ser aprovechada en otros ámbitos docentes.

Por otra parte, esta herramienta sienta las bases a posibles desarrollos futuros, ampliando las funcionalidades de la misma. Se ha desarrollado pensando en su posible escalabilidad y ya que cuenta con toda la estructura base creada (autenticación de usuarios, clonado, ejecución y reporte de resultados), se pueden añadir funcionalidades sin demasiado esfuerzo.

Para concluir, podemos afirmar que los objetivos marcados al comienzo de este Trabajo de Fin de Máster han sido cumplidos y las principales líneas de desarrollo a continuar podrían ser las enumeradas a continuación:

- Dotar de más funcionalidad de GitHub a la herramienta:
 - Subir cambios a los repositorios (git push).
 - Crear issues.
 - Gestionar Pull Requests.
 - Buscar repositorios.
 - Gestión de permisos de usuarios a repositorios y organizaciones.
 - Gestionar Classrooms.
- Enriquecer el formato de la documentación generada.
- Realizar despliegues locales de aplicaciones web (como procesos hijos de la herramienta).

Capítulo 5

Summary and Conclusions

This chapter is compulsory. The memory should include an extended summary and conclusions in english.

5.1. First Section

Capítulo 6

Presupuesto

En este capítulo se especifica un presupuesto que indica cuánto costaría realizar este Trabajo de Fin de Máster si se tratase de un trabajo encargado por un cliente.

6.1. Introducción y coste por hora

Se definirá una tabla con la lista de actividades realizadas en este Trabajo de Fin de Máster. Otra columna indicará la duración en horas que se han empleado para dicha actividad junto con el precio por hora calculado.

El precio por hora que se considerará en este presupuesto es de 30€/hora.

6.2. Funcionalidades requeridas

Actividad	Duración	Precio
Autenticación con GitHub	xx horas	xx €
Listar organizaciones, asignaciones y repositorios	xx horas	xx €
Automatizar la descarga de repositorios	xx horas	xx €
Automatizar la ejecución de scripts en los repositorios	xx horas	xx €
Exportar la información obtenida de la automatización de tareas	xx horas	xx €
Subtotal	xx horas	xx €

Cuadro 6.1: Tabla de actividades, duración y precios de las funcionalidades requeridas

6.3. Funcionalidades extra

Actividad	Duración	Precio
Autocompletado de comandos según contexto	xx horas	xx €
Opción de ayuda según contexto	xx horas	xx €
Visualización del directorio de trabajo actual	xx horas	xx €
Opción para conocer propietarios del repositorio	xx horas	xx €
Subtotal	xx horas	xx €

Cuadro 6.2: Tabla de actividades, duración y precios de las funcionalidades extra

6.4. Coste y duración total

Actividad	Duración	Precio
Funcionalidades requeridas	xx horas	xx €
Funcionalidades extra	xx horas	xx €
Total	xx horas	xx €

Cuadro 6.3: Precio y duración total

Apéndice A

Glosario

A

AJAX : acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML). Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

API : (*Application Programming Interface* o Interfaz de Programación de Aplicaciones). Conjunto de funciones y procedimientos o métodos que ofrece cierta librería para ser utilizados por otro software como una capa de abstracción.

Asíncrono :

Asignación :

Async/Await :

C

CVS : (*Control Versioning System* o Sistema de Control de Versiones). Aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.

G

GitBook :

GitHub : forja para alojar proyectos utilizando el Sistema de Control de Versiones **Git**. Para más información, visitar <https://github.com>.

GitHub Classroom :

H

HTML5 : (*HyperText Markup Language*). Lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web definiendo una estructura básica y un código para la definición del contenido de la misma.

J

JavaScript : lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

M

Metodologías ágiles : conjunto de métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizados y multidisciplinarios. Se caracterizan además por la minimización de riesgos desarrollando software en iteraciones cortas de tiempo.

N

Node.js :

NPM :

O

Organización :

P

Promesa :

R

Repositorio :

S

Student Developer Pack :

Síncrono :

T

Travis-CI :

TDD : (*Test-Driven Development* o Desarrollo Dirigido por Pruebas). Práctica de programación que involucra otras dos prácticas: escribir las pruebas primero (*Test First Development*) y Refactorización de código (*Refactoring*).

Token :

W

Web semántica : idea de añadir metadatos semánticos y ontológicos a la World Wide Web. Esas informaciones adicionales, que describen el contenido, el significado y la relación de los datos, se deben proporcionar de manera formal, para que sea posible evaluarlas automáticamente por máquinas de procesamiento. El objetivo es mejorar Internet ampliando la interoperabilidad entre los sistemas informáticos usando **agentes inteligentes**, es decir, programas en las computadoras que buscan información sin necesidad de interacción humana.

World Wide Web : (WWW). Sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y navega a través de esas páginas usando hiperenlaces.

Apéndice B

Guía de uso

El objetivo de esta guía de usuario es proporcionar a los usuarios un ejemplo para la puesta a punto y ejecución de las funcionalidades implementadas en el paquete NPM ghshell durante el Trabajo de Fin de Máster.

B.1. Instalación

B.1.1. Requisitos

Node.js ≥ 8

B.1.2. Dependencias

Gitbook Calibre

B.1.3. Instalación

Para instalar el paquete, basta con ejecutar el siguiente comando:

```
[~]$ npm install ghshell -g
```

B.2. Ejecución

```
1 var foo = function(){  
2   console.log('foo');  
3 }  
4 foo();
```

B.2.1. Otras consideraciones

Para que

Índice alfabético

Node.js, 3

Bibliografía

- [1] NPM. <https://www.npmjs.com/>.
- [2] FACOM OS IV SSL II USER'S GUIDE, 99SP0050E5. Technical report, 1990.
- [3] D. H. Bailey and P. Swarztrauber. The fractional Fourier transform and applications. *SIAM Rev.*, 33(3):389–404, 1991.
- [4] A. Bayliss, C. I. Goldstein, and E. Turkel. An iterative method for the Helmholtz equation. *J. Comp. Phys.*, 49:443–457, 1983.
- [5] C. Darwin. *The Origin Of Species*. November 1859.
- [6] C. Goldstein. Multigrid methods for elliptic problems in unbounded domains. *SIAM J. Numer. Anal.*, 30:159–183, 1993.
- [7] P. Swarztrauber. *Vectorizing the FFTs*. Academic Press, New York, 1982.
- [8] S. Taásan. *Multigrid Methods for Highly Oscillatory Problems*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1984.