



# Tabla de contenidos

[**Tabla de contenidos**](#_6gru9gvpu1m2) **2**

[**Informe de evaluación heurística**](#_xypqxujddtnc) **3**

[Producto a evaluar](#_ez7malg6ihqk) 3

[Método de evaluación](#_8xvt2f45h38f) 4

[Principios de Usabilidad](#_bpvjqokdxetm) 4

[Resultados de las evaluaciones](#_iomaoty0kksd) 5

[Problemas encontrados](#_hrmcebqubrmu) 5

[Principios incumplidos](#_2ysjsthqdwp5) 7

[Notas por evaluador](#_6jx8hsx07jh4) 7

[**Encuestas**](#_dia0ss5xfmn) **8**

[**Machine Learning**](#_k72ca1ptexxc) **9**

[Flujo del experimento](#_l4t7wn2r09vg) 9

[Preprocesamiento de datos](#_i6gagyiatj21) 9

[Ejecución del algoritmo](#_19ueo3eeum3y) 10

[Reinforcement Learning](#_dqjwxvouqevt) 10

[Supervised Learning](#_u0saks5y9fd8) 10

[Métricas](#_sflz4t838lf9) 10

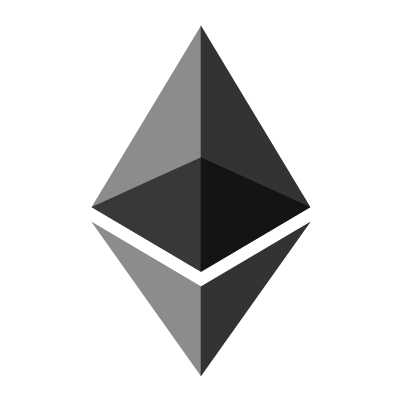
[Script final](#_tkizcv2qpfss) 10

[**Conclusiones**](#_6mw3inhk7ju) **11**

# 

# Informe de evaluación heurística

## Producto a evaluar



Ethereum [4] es una plataforma de blockchain que permite almacenar tanto la lógica de negocio y los datos de una aplicación en una red distribuida, lo cual garantiza la seguridad, inmutabilidad y transparencia de los datos, además de una potencial automatización. El cliente es el programa que implementa todos los aspectos de una blockchain. El cliente Geth [2], escrito en go, implementa la especificación de la red Ethereum descrita en el Yellow Paper [3]. Geth, junto con Parity [5], es una de la implementaciones más populares de la red. Sin embargo, Geth no posee una interfaz gráfica, sino una interfaz de línea de comandos (CLI). El objetivo de este informe será evaluar la usabilidad de la interfaz del cliente Geth.

## 

## Método de evaluación

El cliente Geth será evaluado a través de una lista de heurísticas o principios de usabilidad orientados a programas cuya única interfaz para con el usuario sea a través de la línea de comandos [1]. Se analizarán los comandos de Geth usando dichas heurísticas y se detallará una lista de problemas encontrados con evaluaciones por juicio de expertos.

### Principios de Usabilidad

|  |  |
| --- | --- |
| ID principio | Principio de usabilidad |
| HU1 | Balancear poder y simplicidad |
| HU2 | Usar una estructura jerárquica |
| HU3 | Definir reglas y usarlas |
| HU4 | Modelar el inglés |
| HU5 | Modelar los estándares de la industria |
| HU6 | Usar palabras significativas |
| HU7 | Usar comandos cortos |
| HU8 | Permitir comandos con palabras completas |
| HU9 | Usar palabras distintas |
| HU10 | Abreviar con truncado |
| HU11 | Ser case-sensitive |
| HU12 | Evitar shift keys |
| HU13 | Dar retroalimentación positiva |
| HU14 | Pedir confirmación |
| HU15 | Usar mensajes de error informativos |
| HU16 | Proveer ayuda útil |

## 

## Resultados de las evaluaciones

### Problemas encontrados

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Definición del problema | Comentarios sobre el problema | Ejemplos de ocurrencia del problema | Principios incumplidos |
| P1 | geth js | El comando js no es sufucientemente explícito con respecto a su función. El símbolo js no es imperativo. El símbolo js no es una palabra completa (javascript). | Al momento de ejecutar javascript scripts | H4, H6, H8 |
| P2 | geth --config value | La opción --config no contiene la información con respecto a los argumentos que recibe ni el formato del archivo de configuración | Al momento de usar un archivo TOML como configuración personalizada del cliente | H4 |
| P3 | geth bug | El comando bug no sigue una jerarquía de acción (geth report bug). | Al momento de reportar un bug | H2 |
| P4 | geth init | El comando geth init tiene confusiones semánticas con el comando geth. | Al momento de iniciar una blockchain con un bloque genesis personalizado | H6 |
| P5 | geth attach | El comando geth attach no provee ayuda significativa sobre su uso. El comando geth no sigue una jerarquía de acción (geth attach console). | Al momento de inicializar una consola remota con un nodo geth | H2, H16 |
| P6 | geth dump | El geth geth dump no es claro con sus intenciones y no sigue una jerarquía (geth dump block). | Al momento de imprimir un bloque en específico. | H2, H6 |
| P7 | geth dumpconfig | El comando dumpconfig no sigue una jerarquía (geth dump config). La palabra config no está truncada (cfg) o completa (configuration). | Al momento de imprimir la configuración de un nodo. | H2, H8, H10 |
| P8 | geth --mine | La opción --mine es una acción, y debería ser un comando aparte. | Al momento de empezar el proceso de minado. | H3 |
| P9 | geth removedb | El comando debe seguir una jerarquía (geth remove db) y no existe una palabra completa para db (database). | Al momento de eliminar la base de datos. | H2, H8 |
| P10 | geth account new | La creación de cuentas no sigue una jerarquía de acción y tampoco sigue los estándares de la industria (geth create/add account) | Al momento de crear una nueva cuenta | H2, H5 |
| P11 | geth --mine | El estándar de geth para las opciones son sustantivos. --mine es un verbo, reservado para los comandos | Al momento de iniciar la minería en el nodo | H3 |
| P12 | geth --unlock | La opción --unlock es un verbo, reservado para los comandos. El comando no especifica que el objeto a desbloquear sea una cuenta. | Al momento de desbloquear una cuenta | H3, H8 |
| P13 | geth --gpoblocks | La opción --gpoblock debería ser un comando que siga una jerarquía (search blocks --gasprice) . El nombre de la opción no es simple y la abreviación no es un acrónimo reconocible. | Al momento de buscar los gasprice de los últimos bloques. | H1, H2, H6, H10 |
| P14 | geth makedag | El comando geth makedag no sigue una jerarquía (geth generate dag) | Al momento de generar el DAG de la minería con ethash | H2 |
| P15 | geth makecache | El comando geth makecache no sigue una jerarquía (geth generate cache) | Al momento de generar el cache de verificación de ethash | H2 |
| P16 | geth bug | El comando geth bug no tiene una palabra que indique su funcionalidad (report). | Al momento de reportar un bug. | H6 |
| P17 | geth js | El comando js no es un verbo ni una palabra completa. | Al momento de ejecutar un archivo javascript | H2, H8 |
| P18 | geth removedb | El comando geth removedb no pide confirmación ni usa palabras completas (database) | Al momento de elminar una base de datos | H8, H14 |
| P19 | geth account new | El comando account no es un verbo ni presenta una jerarquía de acción. | Al momento de crear una cuenta | H2, H3 |
| P20 | geth init | El comando geth init no es claro con su real propósito (geth init genesisblock) | Al momento de inicializar una blockchain con un génesis block personalizado | H2, H4 |

### Principios incumplidos

|  |  |  |
| --- | --- | --- |
| ID principio | Principio de usabilidad | Problemas que incumplen el principio |
| HU1 | Balancear poder y simplicidad | P13 |
| HU2 | Usar una estructura jerárquica | P3, P5, P6, P7, P9, P10, P13, P14, P15, P17, P19, P20 |
| HU3 | Definir reglas y usarlas | P8, P11, P12, P19 |
| HU4 | Modelar el inglés | P1, P2, P20 |
| HU5 | Modelar los estándares de la industria | P10 |
| HU6 | Usar palabras significativas | P1, P6, P13, P16 |
| HU7 | Usar comandos cortos |  |
| HU8 | Permitir comandos con palabras completas | P1, P7, P9, P12, P17, P18 |
| HU9 | Usar palabras distintad |  |
| HU10 | Abreviar con truncado | P7, P13 |
| HU11 | Ser case-sensitive |  |
| HU12 | Evitar shift keys |  |
| HU13 | Dar retroalimentación positiva |  |
| HU14 | Pedir confirmación | P18 |
| HU15 | Usar mensajes de error informativos |  |
| HU16 | Proveer ayuda útil | P11 |

### Notas por evaluador

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Evaluador 1 | | | Evaluador 2 | | | Promedio | | | Desviación Estándar | | |
| ID | Definición del problema | S | F | C | S | F | C | S | F | C | S | F | C |
| P1 | geth js | 1 | 2 | 3 | 4 | 3 | 7 | 2,5 | 2,5 | 5,0 | 2,12 | 0,71 | 2,83 |
| P2 | geth --config value | 3 | 3 | 6 | 3 | 1 | 4 | 3,0 | 2,0 | 5,0 | 0,00 | 1,41 | 1,41 |
| P4 | geth init | 4 | 4 | 8 | 3 | 4 | 7 | 3,5 | 4,0 | 7,5 | 0,71 | 0,00 | 0,71 |
| P5 | geth attach | 0 | 3 | 3 | 3 | 0 | 3 | 1,5 | 1,5 | 3,0 | 2,12 | 2,12 | 0,00 |
| P6 | geth dump | 4 | 2 | 6 | 4 | 1 | 5 | 4,0 | 1,5 | 5,5 | 0,00 | 0,71 | 0,71 |
| P7 | geth dumpconfig | 1 | 4 | 5 | 1 | 2 | 3 | 1,0 | 3,0 | 4,0 | 0,00 | 1,41 | 1,41 |
| P10 | geth account new | 4 | 3 | 7 | 3 | 2 | 5 | 3,5 | 2,5 | 6,0 | 0,71 | 0,71 | 1,41 |
| P11 | geth --mine | 0 | 4 | 4 | 1 | 1 | 2 | 0,5 | 2,5 | 3,0 | 0,71 | 2,12 | 1,41 |
| P12 | geth --unlock | 3 | 1 | 4 | 1 | 2 | 3 | 2,0 | 1,5 | 3,5 | 1,41 | 0,71 | 0,71 |
| P13 | geth --gpoblocks | 4 | 2 | 6 | 0 | 4 | 4 | 2,0 | 3,0 | 5,0 | 2,83 | 1,41 | 1,41 |
| P14 | geth makedag | 1 | 2 | 3 | 0 | 1 | 1 | 0,5 | 1,5 | 2,0 | 0,71 | 0,71 | 1,41 |
| P15 | geth makecache | 4 | 2 | 6 | 2 | 2 | 4 | 3,0 | 2,0 | 5,0 | 1,41 | 0,00 | 1,41 |
| P16 | geth bug | 0 | 0 | 0 | 0 | 1 | 1 | 0,0 | 0,5 | 0,5 | 0,00 | 0,71 | 0,71 |
| P18 | geth removedb | 4 | 3 | 7 | 3 | 3 | 6 | 3,5 | 3,0 | 6,5 | 0,71 | 0,00 | 0,71 |

# Encuestas

Previo a realizar las encuestas se generó un manual para la interacción con el cliente geth de ethereum. El manual fue el siguiente:

**# Crear una red de desarrollador y transferir ethers entre dos cuentas**

**## Abrir una terminal y levantar una red de desarrollador.**

La direccion de la carpeta es data

La direccion del canal ipc es ipc

```

geth --datadir data --ipcpath ipc --dev

```

**## Abrir una terminal y crear una cuenta adicional**

La direccion de la carpeta es data

El passphrase es 123exp

```

geth --datadir data account new

Passphrase: 123exp

```

**## Abrir una terminal y agregar una consola de javascript al nodo**

```

geth attach data/ipc

```

**## Revisar el balance de las dos cuentas**

```

web3.fromWei(eth.getBalance(eth.account[0]))

web3.fromWei(eth.getBalance(eth.account[1]))

```

**## Transferir 5 ethers de la cuenta 1 a la cuenta 2**

```

var sender = eth.accounts[0];

var receiver = eth.accounts[1];

var amount = web3.toWei(5, "ether")

eth.sendTransaction({from:sender, to:receiver, value: amount})

```

**# Verificación**

**## Revisar el balance de la segunda cuenta**

```

web3.fromWei(eth.getBalance(eth.account[1]))

```

El rango de análisis para encuesta tomada fue de 1 a 5; siendo 1 muy en desacuerdo y 5 totalmente de acuerdo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **Persona** | | | | | | | | | |  |  |
|  | **Preguntas** | | | | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **TOTAL** | **MEDIA** |
| 1 | Entiendo con fácilidad qué es un nodo en blockchain | | | | | 2 | 2 | 2 | 1 | 4 | 1 | 2 | 2 | 1 | 4 | 21 | 2.1 |
| 2 | La interacción mediante consola(cmd) me parece cómoda | | | | | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 2 | 1 | 5 | 17 | 1.7 |
| 3 | Considero que levantar un nodo desarrollador con el cliente geth es fácil | | | | | 3 | 2 | 2 | 3 | 4 | 3 | 2 | 3 | 2 | 4 | 28 | 2.8 |
| 4 | El comando "help" del cliente geth contiene la información necesaria para entender el uso del cliente | | | | | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 12 | 1.2 |
| 5 | Crear una cuenta(address) en el cliente geth es fácil | | | | | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 4 | 32 | 3.2 |
| 6 | Revisar el balance de las cuentas no es complicado | | | | | 2 | 2 | 2 | 3 | 4 | 3 | 3 | 3 | 3 | 4 | 29 | 2.9 |
| 7 | Transferir ethers de una cuenta a otra es fácil | | | | | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 4 | 23 | 2.3 |
| 8 | Verificar si una transacción se ha realizado no es complicado | | | | | 1 | 2 | 2 | 2 | 4 | 2 | 2 | 2 | 2 | 4 | 23 | 2.3 |
| 9 | Pude convertir weis a ethers con facilidad | | | | | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 5 | 31 | 3.1 |
| 10 | Pude agregar una consola de javascript a mi nodo levantado con facilidad | | | | | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 2 | 4 | 27 | 2.7 |
|  |  |  |  |  | **TOTAL** | 21 | 20 | 21 | 21 | 33 | 21 | 22 | 24 | 20 | 40 |  |  |

# Machine Learning

El cliente Geth, de acorde a la especificación en el Yellow Paper, implementa un lenguaje máquina (o bytecode) el cual está conformado por un número finito de instrucciones, cada una de las cuales posee un peso (o gas). Estos valores sirven para calcular el costo total de una transacción, y así evitar recursividades infinitas producto de ataques o errores que bloqueen los nodos en la red.

Debido a que la complejidad de la programación en el lenguaje máquina de Ethereum es similar al del lenguaje ensamblador, la comunidad creó diversos lenguajes de programación de alto nivel. Uno de estos es Solidity.

El estado actual del desarrollo de contratos inteligentes es el de tener un ecosistema con varios contratos interactuando entre sí. Iniciativas como OpenZeppelin o uPort muestran una clara tendencia a que los proyectos contengan un gran número de contratos incluidos desde su concepción. Mientras que la inclusión de librerías en el desarrollo tradicional no representa mayor inconveniente que el de aumento de espacio o una ligera disminución en el perfomance, en el caso de los contratos inteligentes, mientras más código exista en el proyecto, mayor será el costo de despliegue y de uso.

Es aquí donde radica la relevancia de los algoritmos de aprendizaje automático, en la optimización del proceso de compilación hacia el lenguaje máquina de Ethereum de tal forma que los costos sean minimizados.

El impacto de la disminución tanto en el coste del despliegue de contrato como en los subsecuentes costos por la ejecución de cada función individual podría aumentar la viabilidad de aplicaciones mucho más complejas, lo que añadiría más valor a la red Ethereum y las plataformas que ya están implementadas dentro de ella.

## Flujo del experimento

Debido a que el objetivo es la reducción del gas al momento de compilación, el flujo o ciclo de vida que debería seguir el experimento es similar para los algoritmos de aprendizaje automático.

### Preprocesamiento de datos

Es necesario tener un documento donde se especifique los pesos en gas por cada instrucción. Este documento debe ser fácilmente parseable para poder armar un estructura de datos óptima para los algoritmos de aprendizaje.

Es necesario tener contratos de ejemplo, los cuales pueden ser implementaciones de estándares como los ERC20 o ERC721. Estos contratos deben ser compilados y etiquetados. La compilación dará como resultado bytecodes que pueden ser contrastados con la lista de instrucciones para dar con el costo total de despliegue, lo cual vendría a ser la etiqueta para cada contrato. Asimismo, cada función presente en los contratos debe ser ejecutada y el costo total de esas instrucciones guardados en un mismo dataset. Con esto se tendrían todos los datos suficientes para las siguientes etapas.

### Ejecución del algoritmo

Los algoritmos de machine learning son aplicados en la automatización de las heurísticas usadas por los compiladores.

#### Reinforcement Learning

El algoritmo usado se llama **policy search**, implementado con programación genética [7]. El compilador crea opciones legales al momento de analizar el código y luego usa políticas para escoger el subconjunto de esas opciones que dé un mejor rendimiento. Policy search se aplica en el segundo proceso para automatizar la producción de políticas. Las heurísticas generadas se aplican para varios casos en el proceso de compilación, ya sea para definir el orden instrucciones o para identificar el flujo de control de hiper-bloques de código.

#### Supervised Learning

El caso de estudio es el loop unrolling, que es tema particularmente relevante para un compilador de smart contracts debido a que muchas lógicas de negocio hacen uso de iteraciones o loops, y este tipo de código suelen ser de los que usan más gas. Los algoritmos usados son el de **K-nearest neighbors** [7] y el de **Support Vector Machines** (SVM) [7]. Para construir el vector de características se usan datos del código dentro de los loops tales como el número de instrucciones o el tipo de estas. El gas usado por el código del bloque también podría ser usado como dimensión adicional. Los dos modelos de aprendizaje supervisado usan estas características para el entrenamiento.

Ambos algoritmos usan el modelo de poder, energía o gas obtenido en el preprocesamiento de datos.

### Métricas

La única métrica usada para el experimento es la cantidad de gas que toma cada contrato al ser desplegado y por cada una de sus funciones.

### Script final

El script final debe ser capaz de preprocesar los datos y ejecutar los modelos de aprendizaje automático por separado. Una pequeña tabla comparativa de los precios finales debe ser mostrada al final.

# Conclusiones

Usando prácticas de ingeniería con respecto a las evaluaciones por medio de heurísticas se llega a la conclusión de que el cliente Geth no posee una estructura jerárquica consistente, al ser este el principio de usabilidad predominante en la lista de problemas encontrados. Este problema es relevante ya que impide que los usuarios puedan tener una experiencia intuitiva con el programa, y, siendo el tema de blockchain uno complejo, puede ocasionar una migración a otro tipo de clientes que tengan interfaces más intuitivas o de diferente naturaleza, como la interfaz gráfica de Parity.

Por el lado de los algoritmos de Machine Learning, ciertamente es viable realizar pruebas que busquen optimizar el gas consumido por los contratos a través del proceso de compilación, puesto que el sistema es similar a los modelos por energía de las investigaciones para los compiladores tradicionales. Incluso se puede decir que es un modelo simplificado, puesto que el peso a cada instrucción ya está asignado de antemano.