**SAMUEL CESAR SANTAN**

**CARLOS EVANGELISTA CURI**

**JONATAN MORENO LAZARO**

**MARIA BENAVIDES BOLAÑOS**

lenguajes de programacion go-rust

Desarrollo de Tecnologías de Emergentes

Contenido

[**1. Autores del trabajo, planificación y entrega** 2](#_Toc479023807)

[**1.1 Autores** 2](#_Toc479023808)

[**1.2 Planificación** 2](#_Toc479023809)

[**1.3 Entrega** 2](#_Toc479023810)

[**2. Descripción de las tecnologías** 3](#_Toc479023811)

[**2.1 Descripción de la tecnología Go** 3](#_Toc479023812)

[**2.2 Descripción de la tecnología Rust** 3](#_Toc479023813)

[**3. Criterios de comparación** 4](#_Toc479023814)

[**3.1 Categoría A: Criterios Generales** 4](#_Toc479023815)

[**3.1.1 Criterio A.1: Orientado a objetos GO** 4](#_Toc479023816)

[**3.2 Categoría B: Funcionales** 5](#_Toc479023817)

[**3.3 Categoría C: Criterios Software** 6](#_Toc479023818)

[**3.3.2 Criterio C.2: Dificultad de instalación** 6](#_Toc479023819)

[**3.3.3 Criterio C.3: Dificultad de aprendizaje** 6](#_Toc479023820)

[**3.3.4 Criterio C.4: Criterio Seguridad** 6](#_Toc479023821)

[**3.3.5 Criterio C.5: Tolerancia a errores de sintaxis** 6](#_Toc479023822)

[**3.3.6 Criterio C.6: Variables** 6](#_Toc479023823)

[**3.3.7 Criterios C.7: Punteros** 6](#_Toc479023824)

[**3.3.8 Criterio C.8 Alcance** 6](#_Toc479023825)

[**3.3.9 Criterio C.9: Posee interfaz de desarrollo** 6](#_Toc479023826)

[**3.3.9 Dificultad de uso** 6](#_Toc479023827)

[**4. Evaluación de los criterios por tecnología** 7](#_Toc479023828)

[**4.1 Evaluación de los criterios para GO** 7](#_Toc479023829)

[**4.2 Evaluación de los criterios para RUST** 7](#_Toc479023830)

[**5. Comparación de las tecnologías** 8](#_Toc479023831)

[**6. Recomendaciones** 11](#_Toc479023832)

[**6.1 Autenticación de credenciales en Páginas Web** 11](#_Toc479023833)

[6.1.1 Descripción de la situación 11](#_Toc479023834)

[Mecanismo de autenticación: 11](#_Toc479023835)

[6.1.2 Recomendación de tecnología a utilizar 11](#_Toc479023836)

[**6.2 Situación 2** 12](#_Toc479023837)

[6.2.1 Descripción de la situación 12](#_Toc479023838)

[6.2.2 Recomendación de tecnología a utilizar 12](#_Toc479023839)

# **1. Autores del trabajo, planificación y entrega**

## **1.1 Autores**

Carlos A. Evangelista Curi

María F. Benavides Bolaños

Jonathan Moreno

Samuel Cesar

## **1.2 Planificación**

El reparto de tareas ha sido equitativo, diferenciando las tareas del coordinador, por un lado, y la del resto de integrantes del grupo. Cada miembro del grupo se ha especializado en un lenguaje de programación. (dos en cada uno)

En este enlace hay un ejemplo de una posible planificación del trabajo, que se puede usar como referencia, pero indicando los nombres reales de las tecnologías y de los participantes.

El peso de este trabajo en la calificación total de la asignatura es de un 10%, por lo tanto, requiere de una dedicación de 15 horas del total de 150 horas de la asignatura.

Al estar formado por 4 integrantes el tiempo total de desarrollo de dicho proyecto será 60 horas, repartidas en el siguiente diagrama de Gantt.

<https://app.ganttpro.com/shared/token/8bcdf9f4cef31142dd046336465332cd6389effe8fbd6e7a4aee761275827477>

## **1.3 Entrega**

- Incluimos el enlace (URL) a un repositorio en GitHub donde incorporamos nuestro trabajo y archivos:

<https://github.com/samuel36/TG2>

- Trabajo terminado: TG2\_final.docx

- Presentación del trabajo: TG2\_final.pptx

- La creación de una carpeta con el nombre de TG2, ha sido con visión al futuro, para la incorporación del futuro proyecto TG3 bajo el mismo repositorio.

# **2. Descripción de las tecnologías**

## **2.1 Descripción de la tecnología Go**

Go es un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [concurrente](https://es.wikipedia.org/wiki/Computaci%C3%B3n_concurrente) y [compilado](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_compilado) inspirado en la sintaxis de [C](https://es.wikipedia.org/wiki/C). Ha sido desarrollado por [Google](https://es.wikipedia.org/wiki/Google) y sus diseñadores iniciales son [Robert Griesemer](https://es.wikipedia.org/w/index.php?title=Robert_Griesemer&action=edit&redlink=1), [Rob Pike](https://es.wikipedia.org/wiki/Rob_Pike) y [Ken Thompson](https://es.wikipedia.org/wiki/Ken_Thompson). Actualmente está disponible en formato binario para los [sistemas operativos](https://es.wikipedia.org/wiki/Sistema_operativo) [Windows](https://es.wikipedia.org/wiki/Microsoft_Windows), [GNU/Linux](https://es.wikipedia.org/wiki/GNU/Linux), [FreeBSD](https://es.wikipedia.org/wiki/FreeBSD) y [Mac OS X](https://es.wikipedia.org/wiki/OS_X), pudiendo también ser instalado en estos y en otros sistemas con el código fuente.[2](https://es.wikipedia.org/wiki/Go_(lenguaje_de_programaci%C3%B3n)#cite_note-2) [3](https://es.wikipedia.org/wiki/Go_(lenguaje_de_programaci%C3%B3n)#cite_note-3) Go es un lenguaje de programación compilado, concurrente, imperativo, estructurado, orientado a objetos —de una manera bastante especial— y con recolector de basura que de momento está soportado en diferentes tipos de sistemas UNIX, incluidos [Linux](https://es.wikipedia.org/wiki/Linux), [FreeBSD](https://es.wikipedia.org/wiki/FreeBSD) y [Mac OS X](https://es.wikipedia.org/wiki/Mac_OS_X). También está disponible en [Plan 9](https://es.wikipedia.org/wiki/Plan_9_from_Bell_Labs) puesto que parte del compilador está basado en un trabajo previo sobre el sistema operativo [Inferno](https://es.wikipedia.org/wiki/Inferno). Las arquitecturas soportadas son i386, amd64 y ARM.



**Características**

* Go usa una sintaxis parecida a C por lo que los programadores que hayan usado dicho lenguaje se sienten muy cómodos con él.
* Go usa [tipado estático](https://es.wikipedia.org/wiki/Tipado_est%C3%A1tico) (statically typed) y es tan eficiente como C.
* Go tiene muchas de las características y facilidad de lenguajes dinámicos como [Python](https://es.wikipedia.org/wiki/Python)
* Aún siendo un lenguaje diseñado para la programación de sistemas, provee de un [recolector de basura](https://es.wikipedia.org/wiki/Recolector_de_basura), reflexión y otras capacidades de alto nivel que lo convierten en un lenguaje muy potente.
* Go admite el paradigma de programación orientada a objetos, pero a diferencia de los lenguajes de programación más populares no dispone de herencia de tipos y tampoco de palabras clave que denoten claramente que soporta este paradigma. Otro detalle que puede resultar confuso es que la definición de un tipo ("clase") se realiza por medio de declaraciones separadas (interfaces, structs, embedded values). Go permite el uso de delegación (a través de embedded values) y polimorfismo (por medio de interfaces).

## **2.2 Descripción de la tecnología Rust**

Rust es un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [compilado](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_compilado), de [propósito general](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_de_prop%C3%B3sito_general) y [multiparadigma](https://es.wikipedia.org/wiki/Programaci%C3%B3n_multiparadigma) que está siendo desarrollado por [Mozilla](https://es.wikipedia.org/wiki/Fundaci%C3%B3n_Mozilla). Ha sido diseñado para ser un lenguaje seguro, [concurrente](https://es.wikipedia.org/wiki/Programaci%C3%B3n_concurrente) y práctico. Es un [lenguaje de programación multiparadigma](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_multiparadigma), soporta programación [funcional](https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional) pura, [por procedimientos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_procedimientos), [imperativa](https://es.wikipedia.org/wiki/Programaci%C3%B3n_imperativa) y [orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos).

Según la política de Mozilla, Rust es desarrollado de forma totalmente abierta y busca la opinión y contribución de la comunidad. El diseño del lenguaje se ha ido perfeccionando a través de las experiencias en el desarrollo del motor de navegador [Servo](https://es.wikipedia.org/wiki/Servo_(motor_de_renderizado)), y el propio compilador de Rust. Aunque es desarrollado y patrocinado por Mozilla y Samsung, es un proyecto comunitario. Una gran parte de las contribuciones proceden de los miembros de la comunidad.

**Características**

* La sintaxis de Rust es similar a la de [C](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)) y C++, con bloques
* El sistema está diseñado para tener un acceso seguro a la memoria, y no permite [punteros nulos](https://es.wikipedia.org/wiki/Puntero_(inform%C3%A1tica)) o [punteros colgantes](https://es.wikipedia.org/wiki/Referencia_colgante).
* El sistema de tipos soporta un mecanismo similar a las clases de tipos, llamado "traits", inspirados directamente por el lenguaje [Haskell](https://es.wikipedia.org/wiki/Haskell). Esta es una facilidad para el polimorfismo que soporta distintos tipos de argumentos (*polimorfismo ad-hoc*), lograda mediante la adición de restricciones para escribir declaraciones de variables.
* Rust cuenta con [inferencia de tipos](https://es.wikipedia.org/wiki/Inferencia_de_tipos), para las variables declaradas con la palabra clave let. Tales variables no requieren ser inicializadas con un valor asignado con el fin de determinar su tipo. A tiempo de compilación produce un error si cualquier rama de código falla al asignar un valor a la variable.
* La concurrencia es soportada a través de tareas simples, similares a los encontrados en [Erlang](https://es.wikipedia.org/wiki/Erlang) y otros lenguajes basado en actores. En dichos sistemas, las tareas se comunican a través del paso de mensajes, en lugar de compartir los datos directamente.
* El sistema de objetos de Rust se basa en implementaciones, *traits* y tipos estructurados. Las implementaciones cumplen una función similar a la de las [clases](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)) en otros lenguajes.

# **3. Criterios de comparación**

## **3.1 Categoría A: Criterios Generales**

### **3.1.1 Criterio A.1: Orientado a objetos GO**

**Descripción**: NO

**Tipo de valor**: Booleano (Sí/No)

**3.1.2 Criterio A.2: Software Abierto**

**Descripción**: Existe una documentación abierta y libre de los diseños y componentes software

**Tipo de valor**: Booleano (Sí/No)

:

**3.1.3 Criterio A.3: Madurez y versiones del producto**

**Descripción**: Cuántas versiones y revisiones de los productos existen en el mercado. Se prefieren segundas versiones/revisiones o más porque se entiende que se han arreglado fallos de diseño de las primeras versiones. Escala 1-5, donde 1 es primera versión y 5 existe un amplio abanico revisiones.

**Tipo de valor**: Escala 1-5.

**3.1.4 Criterio A.4: Multiplataforma**

**Descripción**: Se puede programar con estos lenguajes desde distintos SSOO.

**Tipo de valor**: Booleano (Sí/No)

**3.1.5 Criterio A.5: Desarrollador**

**Descripción**: Empresa desarrolladora del lenguaje de programación.

**Tipo de valor**: String.

**3.1.6 Criterio A.6: Año de Aparición**

**Descripción**: Año en el que fue lanzado el lenguaje de programación.

**Tipo de valor**: Numérico.

**3.1.7 Criterio A.7: Influencia**

**Descripción**: Lenguajes de programación por los que han sido influenciados.

**Tipo de valor**: String.

## **3.2 Categoría B: Funcionales**

**3.2.1 Criterio B.1: Garbage Collector**

**Descripción**: El programa posee una solución automática implícito de gestión de memoria

**Tipo de valor**: Booleano (Sí/No)

**3.2.2 Criterio B.2: Portabilidad**

**Descripción**: El programa puede ser ejecutado en diferentes entornos

**Tipo de valor**: Booleano (Sí/No)

**3.2.3 Criterio B.3: Representaciones (Interfaces)**

**Descripción**: Permite la obtención de programas con interfaces

**Tipo de valor**: Booleano (Sí/No)

## **3.3 Categoría C: Criterios Software**

**3.3.1 Criterio C.1: Rapidez de compilación: texto libre**

**Descripción:** Se espera que sea más lento que otros lenguajes debido a su seguridad, sin embargo, si se aplica la misma seguridad a otros lenguajes puede ser incluso más rápido.

### **3.3.2 Criterio C.2: Dificultad de instalación**

**Tipo de valor:** Escala 1-5.

### **3.3.3 Criterio C.3: Dificultad de aprendizaje**

**Tipo de valor:** Escala 1-5.

### **3.3.4 Criterio C.4: Criterio Seguridad**

**Tipo de valor:** Escala 1-5.

### **3.3.5 Criterio C.5: Tolerancia a errores de sintaxis**

**Descripción**: dado que se basa en la seguridad cualquier posible error en la sintaxis llegara con un aviso, aunque el programa compile

### **3.3.6 Criterio C.6: Variables**

**Descripción:** Obliga a inicializar una variable.

### **3.3.7 Criterios C.7: Punteros**

**Descripción**: Si, y no permite punteros nulos o colgantes

**Tipo de valor:** Booleano

### **3.3.8 Criterio C.8 Alcance**

**Descripción**: Funcional pura, por procedimientos, imperativa y Orientada a Objetos

### **3.3.9 Criterio C.9: Posee interfaz de desarrollo**

**Tipo de valor:** Booleano

### **3.3.9 Dificultad de uso**

**Descripción:** Rust se usa desde la consola, dado que windows la usa menos, es más difícil moverse por ella

# **4. Evaluación de los criterios por tecnología**

## **4.1 Evaluación de los criterios para GO**

|  |  |
| --- | --- |
| CRITERIOS | EVALUACIÓN |
| **GENERALES** |  |
| Criterio A.1: Orientado a Objetos | NO |
| Criterio A.2: Software abierto | SI |
| Criterio A.3: Versión | 1.8[1](https://es.wikipedia.org/wiki/Go_(lenguaje_de_programaci%C3%B3n)#cite_note-1) |
| Criterio A.4 Multiplataforma | Si podemos programar codigo GO en  [GNU/Linux](https://es.wikipedia.org/wiki/GNU/Linux), [Mac OS X](https://es.wikipedia.org/wiki/OS_X), [Windows](https://es.wikipedia.org/wiki/Microsoft_Windows), otros |
| Criterio A.5: Desarrollador | GOOGLE |
| Criterio A.6: Año de Aparición | 2009 |
| Criterio A.7: Influencia | C, C++, Python o Java |
| **FUNCIONALES** |  |
| Criterio B.1: Garbage Collector | Si  <http://golang.org/doc/go_faq.html#garbage_collection> |
| Criterio B.2: Portabilidad | SI |
| Criterios B.3: Representaciones (interfaces) | SI |

## **4.2 Evaluación de los criterios para RUST**

|  |  |
| --- | --- |
| CRITERIOS | EVALUACIÓN |
| **GENERALES** |  |
| Criterio A.1: Orientado a Objetos | Si |
| Criterio A.2: Software abierto | SI |
| Criterio A.3: Versión | 1.16.0 |
| Criterio A.4 Multiplataforma | Si |
| Criterio A.5: Desarrollador | Mozilla |
| Criterio A.6: Año de Aparición | 2015 |
| **SOFTWARE** |  |
| Criterio C.7: Rapidez | Algo más baja  Esto es debido a que se centra en la seguridad, si a otros lenguajes se le aplica la misma seguridad en el código, rust podría incluso superarles |
| Criterio C.8 Dificultad de instalación | 2 |
| Criterio C.9: Dificultad de aprendizaje | 4 |
| Criterio C.10: Seguridad | 5 |
| Criterio C.11: Tolerancia en sintaxis | 1  El programa te indicara en todo momento si tu código puede llegar a dar error o incumplir alguna norma |
| Criterio C.12: Variables | Se han de inicializar siempre |
| Criterio C.13: Punteros | Si  Tiene como ventaja que no permite los punteros nulos o colgantes, lo que hace aprovechar la memoria |
| Criterio C.14: Alcance | Si |
| Criterio C.15: Interfaz de desarrollo | No |
| Criterio C.16: Dificultad de uso | 3  Dado que el lenguaje es nuevo y no tiene interfaz de desarrollo, se han de utilizar comandos, lo que dificulta, sobre todo al principio, su uso |

# **5. Comparación de las tecnologías**

Debe incluir al menos una tabla resumen, en sección de página horizontal, cruzando los criterios y los valores de cada tecnología. Con una columna de comentarios sobre la comparación

Esta tabla anterior es obligatoria y deben completarla los autores del trabajo, aunque se pueden incluir otros gráficos o tablas complementarias copiadas y pegadas desde diversas fuentes de información, siempre que debajo de cada uno se indique la fuente (al menos la URL).

|  |  |  |  |
| --- | --- | --- | --- |
| CRITERIOS | GO | RUST | COMENTARIOS |
| CriterioA.1: Orientado a Objetos | En GO, NO admite Herencia | Si |  |
| CriterioA.2: Software abierto | OPEN SOURCE | Si |  |
| CriterioA.3: Versión | 1.8 | 1,16,0 | En Go([16 de febrero](https://es.wikipedia.org/wiki/16_de_febrero) de [2017](https://es.wikipedia.org/wiki/2017) (1 mes y 15 días)) |
| Criterio A.4 Multiplataforma | En GO para Linux y Windows | Si, Linux, Windows y Mac | En GO Ya que introducen unas sentencias nuevas basadas en lenguajes que no son tan conocidos como C o java |
| Criterio A.5: Desarrollador | google | Mozilla |  |
| Criterio A.6: Año de Aparición | 2009 | 2015. | Es menos reciente GO. Rust apareció en el 2015 pero se desarrolló desde 2016 |
| Criterio A.7: Influencia | C, C++, Python o Java, eso lo hace muy flexible para los que ya han programado en dichos lenguajes y claro es ahí donde está el secreto de su poderosa capacidad para la creación de programas. | C#, c++, Haskell, Limbo, Ruby, Ocalm, Racket, Swift, Lisp | Aunque lo más interesante es la influencia que tiene GO ya que posee lo mejor de cada lenguaje de programación. Rust posee la esencia de lenguajes de bajo nivel. |
| Criterio B.1: Garbage Collector | En GO totalmente precisa,  Barrido simultáneo | No, esta en desarrollo | En GO (resultados en tiempos de pausa más pequeños |
| Criterio B.2: Portabilidad | Se considera parcial | Si, al ser un lenguaje de bajo nivel como C | Constante mejoría en la portabilidad |
| Criterios B.3: Representaciones (interfaces) | Go no está orientado a objetos porque no existe jerarquía de tipos, pero implementa interfaces | No, esta en desarrollo | Se maneja todo mediante comandos de consola y un editor de texto |
| Criterio C.7: Rapidez | las características de este lenguaje hacen pensar a Google que Go puede llegar a ser adoptado rápidamente por los programadores en la creación de las primeras aplicaciones. | Dado que se centra en la seguridad, puede ser más lento que otros lenguajes de bajo nivel | En GO Pese a tratarse de un lenguaje experimental cuyo desarrollo se espera siga evolucionando. |
| Criterio C.8 Dificultad de instalación | sus creadores se inspiraron en la versatilidad | Nula, es fácil de instalar y trabajar con el | A la hora de la instalación GO destaca por eficiencia, rapidez de ejecución y facilidad de lenguaje |
| Criterio C.9: Dificultad de aprendizaje | Resulta intuitivo | Medio, dado que se trabaja desde consola hay que conocer comandos aparte del lenguaje |  |
| Criterio C.10: Seguridad | Go son capaces de reconocer y bloquear el uso de certificados revocados debido a que los propios lenguajes de programación carecen de las medidas de seguridad necesarias para poder detectarlos | Su pilar es la seguridad y su acceso a memoria |  |
| Criterio C.11: Tolerancia en sintaxis | Estática | El programa avisa de posible error o falta de seguridad, aunque el programa compile | A pesar de que GO utiliza la compilación estática, es importante tener en cuenta que las declaraciones de importación son mucho más cercanas a las de Java o Python que las directivas de inclusiones en C |
| Criterio C.12: Variables | Similitud a Java | No pueden ser nulas, siempre se han de inicializar |  |
| Criterio C.13: Punteros | Go no tiene aritmética de punteros, según los creadores, la razón es la seguridad. | Si, posee punteros. No admite punteros nulos o colgantes | Sin aritmética de punteros es posible crear un lenguaje en el que no se puede obtener una dirección ilegal que sea usada de forma incorrecta. Rust admite punteros y se centra en la seguridad de los mismos |
|  |  |  |  |

# **6. Recomendaciones**

Deben platearse posibles situaciones de uso, y recomendar justificadamente una u otra tecnología en función de la situación. Al menos 2 situaciones diferentes.

## **6.1 Autenticación de credenciales en Páginas Web**

### 6.1.1 Descripción de la situación

El proceso de confirmar que algo (o alguien) es quien dice ser.

Los métodos de autenticación están en función de lo que utilizan para la verificación y nos centraremos en:

* Sistemas basados en algo conocido. Ejemplo, un *password*.

## Mecanismo de autenticación:

1. El usuario solicita acceso a un sistema.
2. El verificador solicita al usuario que se autentique.
3. El usuario aporta las credenciales que le identifican(Id y Password) y permiten verificar la autenticidad de la identificación.
4. El verificador valida según sus reglas si las credenciales aportadas son suficientes para dar acceso al usuario o no.

### 6.1.2 Recomendación de tecnología a utilizar

El lenguaje propuesto para desarrollar esta situación es el lenguaje de programación Go

|  |  |  |
| --- | --- | --- |
| **Criterios relevantes para la decisión** | **Ventajas de Go** | **Ventajas de Rust** |
| **Orientado a objetos** | No | Si |
| **Multiplataforma** | Si | Si |
| **Seguridad** | Si | Si |
| **Concurrencia** | Si. el punto fuerte de Go. Las **goroutines** (las que permiten utilizar esta propiedad) son muy sencillas de aprender y con una implementación con poco que criticar. | Si |
| **Garbage Collector** | SI | - |
|  | *Facilidad de desarrollo de páginas y aplicaciones Web* |  |

Go además es, Excelente una herramienta para servidores: proporciona una buena opción para sustituir a C y ensamblador en servidores ya que conserva muchas de las facilidades que estos últimos contienen, como la gestión de memoria, pero con un sintaxis más simple y fácil de aprender.

**6.2 Situación 2**

6.2.1 Descripción de la situación

Somos un grupo de programadores contratados por una gran empresa para crear un servicio de trafico de archivos y datos en local. Esta empresa es muy recelosa en cuanto a la seguridad y las vulnerabilidades. nos ha dado via libre para elegir entre dos lenguajes de programacion emergentes: Go y Rust

|  |  |  |
| --- | --- | --- |
| **Criterios relevantes para la decisión** | **Ventajas de Go** | **Ventajas de Rust** |
| **Orientado a objetos** | No | Si |
| **Multiplataforma** | Si | Si |
| **Seguridad** | Si | Si, sobre todo en memoria |
| **Punteros** | No | Si, con su facilidad de acceso a memoria |

6.2.2 Recomendación de tecnología a utilizar

Tras una coparativa, hemos decidido usar Rust, ya que aunque sea mas lento potencia la seguridad en su lenguaje.

Ademas al trabajar a bajo nivel permite hacer aplicaciones mas robustas.