lENGUAJES go/rust

Grado Sistemas de Información

SAMUEL CESAR SANTANA (Coordinador)

CARLOS ALBERTO EVANGELISTA CURI

JONATAN MORENO LAZARO

MARIA BENAVIDES BOLAÑOS

Desarrollo de Tecnologías Emergentes 2017

Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc481962757)

[1.1 Autores 3](#_Toc481962758)

[1.2 Planificación 3](#_Toc481962759)

[1.3 Entrega 3](#_Toc481962760)

[2. Requisitos del prototipo a implementar 4](#_Toc481962761)

[2.1 Requisitos funcionales 4](#_Toc481962762)

[2.2 Otros requisitos 5](#_Toc481962763)

[3. Criterios de comparación en la implementación 6](#_Toc481962764)

[3.1 Criterio 1: Horas empleadas en el desarrollo del sistema 6](#_Toc481962765)

[3.2 Criterio 2: Velocidad de ejecución de Código 6](#_Toc481962766)

[3.3 Criterio 3: Documentación 6](#_Toc481962767)

[3.4 Criterio 4: Coste del entorno de desarrollo 6](#_Toc481962768)

[3.5 Criterio 5: Líneas de código 7](#_Toc481962769)

[3.6 Criterio 6: Tiempo de instalación del programa 7](#_Toc481962770)

[3.7 Criterio 7: Tiempo de aprendizaje 7](#_Toc481962771)

[3.8 Criterio 8: Calidad del sistema 7](#_Toc481962772)

[3.9 Criterio 9: Tiempo de arranque de tecnologías 7](#_Toc481962773)

[4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología A 9](#_Toc481962774)

[4.1 Documentación de diseño 9](#_Toc481962775)

[4.2 Documentación de construcción 9](#_Toc481962776)

[4.3 Documentación de pruebas 9](#_Toc481962777)

[4.4 Documentación de instalación 9](#_Toc481962778)

[4.5 Manual de usuario 9](#_Toc481962779)

[5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología B 10](#_Toc481962780)

[5.1 Documentación de diseño 10](#_Toc481962781)

[5.2 Documentación de construcción 10](#_Toc481962782)

[5.3 Documentación de pruebas 10](#_Toc481962783)

[5.4 Documentación de instalación 10](#_Toc481962784)

[5.5 Manual de usuario 10](#_Toc481962785)

[6. Comparación de las dos implementaciones 11](#_Toc481962786)

[6.1 Evaluación de los criterios en la implementación usando la tecnología A 11](#_Toc481962787)

[6.2 Evaluación de los criterios en la implementación usando la tecnología B 11](#_Toc481962788)

[7. Comparación de la implementación de las tecnologías 12](#_Toc481962789)

[8. Conclusiones 14](#_Toc481962790)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

SAMUEL CESAR SANTANA (Coordinador)

CARLOS ALBERTO EVANGELISTA CURI

JONATAN MORENO LAZARO

MARIA BENAVIDES BOLAÑOS

## 1.2 Planificación

Hay que tener en cuenta que cada participante del grupo debe tener asignadas tareas que sumen al menos 45 horas. El peso de este trabajo en la calificación total de la asignatura es de un 30%, por lo tanto, requiere de una dedicación de 45 horas del total de 150 horas de la asignatura.

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 esta la 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 30%, por lo tanto, requiere de una dedicación de 45 horas del total de 150 horas de la asignatura.

Al estar formado por 4 integrantes el tiempo total de desarrollo de dicho proyecto será 180 horas, repartidas en el siguiente diagrama de Gantt.

<https://app.ganttpro.com/shared/token/8bcdf9f4cef31142dd046336465332cd6389effe8fbd6e7a4aee761275827477/170949>

## 1.3 Entrega

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

<https://github.com/samuel36/TG3>

* Informe del trabajo: con el nombre TG3\_final.docx
* Presentación del trabajo: TG3\_final.pptx
* Prototipos obtenidos implementando cada una de las tecnologías (deben incluir el código fuente y todos los archivos necesarios para la instalación y uso de cada prototipo):
  + PrototipoTecnologiaA\_final.zip (o .rar)
  + PrototipoTecnologiaB\_final.zip (o .rar)

# 2. Requisitos del prototipo a implementar

Implementar 3 tipos de códigos en ambas tecnologías con sus respectivas comparaciones, diseños, pruebas y ejecución:

***A. Código simple para diferenciar la sintaxis***

\* hola mundo

**B. Menú funcional de calculadora**

\*suma

\*resta

\*multiplicación

\*división

## 2.1 Requisitos funcionales

Los requisitos funcionales deben ser los mismos para las dos implementaciones.

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

| REQ. | DESCRIPCIÓN |
| --- | --- |
| RF01 | Se genera un código que sacara por pantalla Un texto “Hola Mundo” |
| RF02 | Se creará una calculadora con las funcionalidades de suma, resta, multiplicación y división usando 2 argumentos |
| RF03 | Se mostrará con rasgos visibles cuando se ejecute código en Go o Rust a través de la consola del sistema |
| RF04 | La aplicación utilizara los métodos correspondientes que serán mostrados por pantalla (Menú) para ser elegidos, hasta que compile exitosamente. |
| RF05 | Los datos Generados serán en tiempo Real |
| RF06 | Códigos fáciles de comparar, usar y practicar |
| RF07 | Podrá operar con números de más de 9 dígitos |
|  |  |

## 2.2 Otros requisitos

En la siguiente tabla se indicará el catálogo de requisitos no funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
|  | REQUISITOS HW |
| RNF01 | Será necesaria Conectividad Ethernet. |
| RNF02 | Conexión constante a la corriente. |
| RNF03 | 256MB RAM mínima requerida para el dispositivo sobre el que se instala la app |
|  | REQUISITOS SW |
| RNF04 | El lenguaje de programación debe de ser Go y Rust |
| RNF05 | Funcionará al menos sobre los siguientes sistemas operativos:   * + Windows 8   + OsX   + Android   iOS |
| RNF06 | La información mostrada no requiere de ningún coste ni de aplicaciones externas ya que son propias del equipo donde se ejecute |
|  | REQUISITOS DE RENDIMIENTO |
| RNF07 | La información se debe mostrar en menos de 30 segundos |
| RNF08 | Los usuarios deben ser capaces de aprender a utilizar el programa en 3 minutos |

# 3. Criterios de comparación en la implementación

A través del siguiente apartado se trata de establecer un conjunto de criterios que permitan evaluar la implementación de forma íntegra en cada una de las tecnologías en cuestión.

Todos los criterios de las diferentes categorías se valorarán en euros cuando este en el ámbito económico, en horas y minutos en los casos que se trate del tiempo en realizar alguna tarea. El tiempo, el precio y la calidad tendrán puntuaciones en intervalos de 1 a 10 en las tablas, siendo 1 la peor puntuación y 10 la mejor.

## 3.1 Criterio 1: Horas empleadas en el desarrollo del sistema

**Nombre del criterio:** Horas empleadas en el desarrollo del sistema.

**Descripción:** Horas invertidas en la creación de distintos códigos utilizados para la comparación

**Tipo de valor:** Numérico (horas).

## 3.2 Criterio 2: Velocidad de ejecución de Código

**Nombre del criterio:** Velocidad de ejecución de Código

**Descripción:** Relación tiempo de ejecución/tiempo de desarrollo

**Tipo de valor:** Numérico (horas).

## 3.3 Criterio 3: Documentación

**Nombre del criterio:** Documentación

**Descripción:** Se utiliza Atom como editor de código Open Source y manuales de aprendizaje e introducción a golang <https://golang.org/doc/> y <https://goyox86.github.io/elpr/README.html> para rust.

**Tipo de valor:** Económico (euros).

## 3.4 Criterio 4: Coste del entorno de desarrollo

**Nombre del criterio:** Coste del entorno de desarrollo

**Descripción:** AL ser un software distribuido y desarrollado libremente NO siendo necesario desembolsar en ningún momento un capital para adquirir el entorno de desarrollo (licencia) con lo cual el acceso ha sido gratuito desde el primer momento (libre). (Ambos lenguajes)

**Tipo de valor:** Económico (euros).

## 3.5 Criterio 5: Líneas de código

**Nombre del criterio:** Líneas de código

**Descripción:** Número de líneas que componen los distintos códigos generados en la tecnología

**Tipo de valor:** Numérico (Líneas).

## 3.6 Criterio 6: Tiempo de instalación del programa

**Nombre del criterio:** Tiempo de instalación del programa

**Descripción:** Tiempo estimado que se tarda en descargar el programa de desarrollo y la puesta en marcha del mismo

**Tipo de valor:** Numérico (minutos/horas).

## 3.7 Criterio 7: Tiempo de aprendizaje

**Nombre del criterio:** Tiempo de aprendizaje

**Descripción:** aprendizaje en horas para la investigación y pruebas en el entorno de desarrollo

**Tipo de valor:** Numérico (minutos/horas).

## 3.8 Criterio 8: Calidad del sistema

**Descripción:** Se evalúa la calidad del sistema creado por los desarrolladores, es decir, realizar una comparación de todos los aspectos de cada aplicación para ver en cuál es mejor.

Tipo de valor: intervalo [1, 10]

## 3.9 Criterio 9: Tiempo de arranque de tecnologías

**Descripción:** Es el Tiempo que pasa desde que se conecta a la aplicación, se ejecuta el código y finalmente da resultados

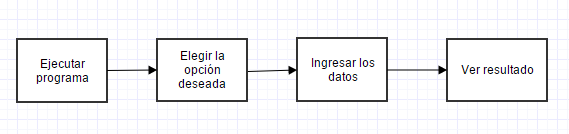
**Tipo de valor:** Numérico (segundos)

# 4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología A (Go)

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología A, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

## 4.1 Documentación de diseño

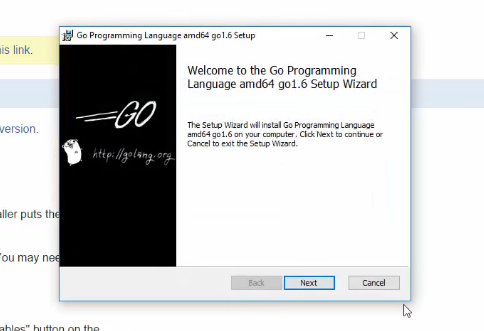
Nuestro prototipo “principal” es una calculadora que puede resolver las 4 operaciones básicas (suma, resta, multiplicación y división), que tiene las siguientes iteraciones.



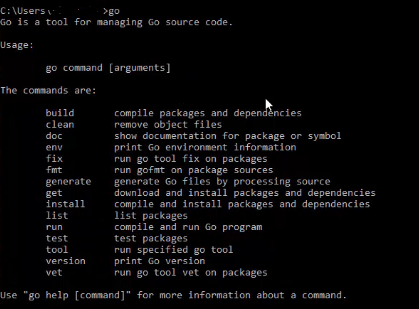
Que se ejecuta mediante el CMD.

## 4.2 Documentación de construcción

Lo primero que hacemos es instalar Go



Una vez instalado podemos ver los comandos que disponemos en Go



Antes de empezar tenemos que descargar también un entorno de desarrollo con el cual podamos programas en Go, hay múltiples entornos de desarrollo que podemos usar, en nuestro caso hemos optado por Atom (más adelante veremos su instalación).

Para poder crear cualquier programa en Go es necesario seguir con el siguiente formato

**package main** (todo programa en Go debe tener un paquete main , que se ejecuta en primera instancia)

**import "fmt"** (abreviación de format, se inicia en package main)

**func main()** { (declaración de funciones)

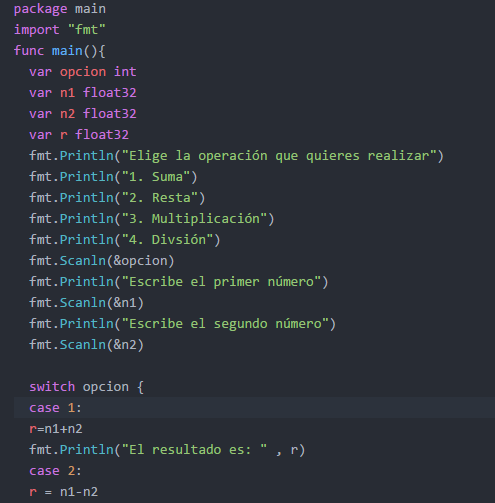
**fmt.Println("Hola mundo")** (parámetro de tipo String Hola Mundo)

}

Tenemos que guardar el programa con la extensión .**go.** Por ejemplo holamundo.go

Una vez creado el programa, para poder ejecutarlo es necesario acceder al CMD:

1. **Especificar la ruta D:\Go\src\HolaMundo** (Tenemos que situarnos en la carpeta donde guardamos el programa)
2. **go run holamundo.go** (Para ejecutar los programas, una vez estemos dentro de la carpeta es necesario poner la siguiente línea de texto: **go run** programa.**go**
3. **go build holamundo.go** (para crear el ejecutable (binario) tenemos que poner la línea de texto: **go build** programa**.go** que a su vez generará un archivo con el nombre de programa**.exe**

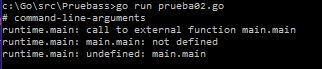


Aquí podemos ver parte de código creado en el entorno de desarrollo Atom,

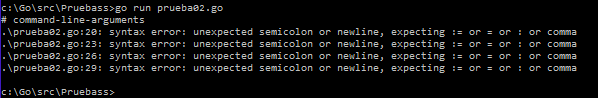
## 4.3 Documentación de pruebas

Al principio, cuando estábamos probando la sintaxis de Go, tuvimos multiples errores debido al poco conocimiento acerca de este lenguaje de programación.

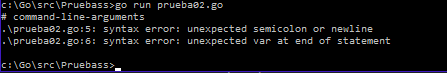
Probamos…



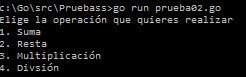
Y probamos…



Y seguimos probando…

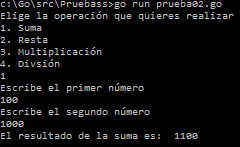


Hasta que…

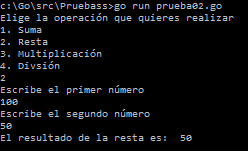


Una vez hecho el programa, pasamos a probar si puede realizar los 4 operaciones básicas y ver qué pasa si, no elegimos ninguna opción que nos muestra nuestro menú de calculadora.

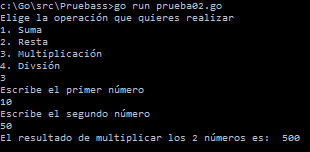
**Suma**

****

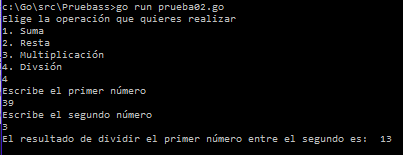
**Resta**

****

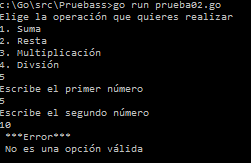
**Multiplicación**

****

**División**

****

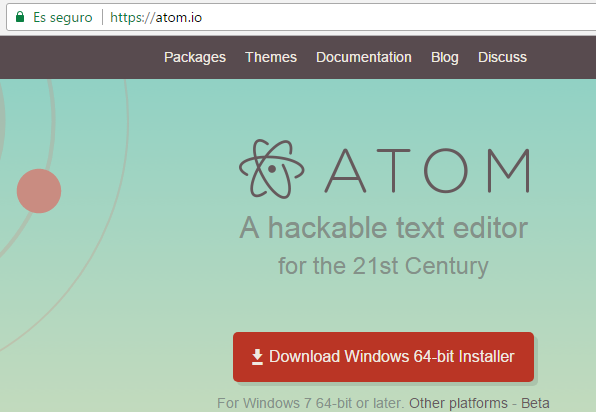
**Y si ahora elegimos una opción que no muestra el menú…**

****

## 4.4 Documentación de instalación

Una vez instalado Go en el ordenador necesitamos instalar un entorno de desarrollo, con el cual podamos usar este lenguaje, Hemos elegido Atom, ya que gran parte de usuarios de Go en foros, dan un gran valor a este entorno de desarrollo.

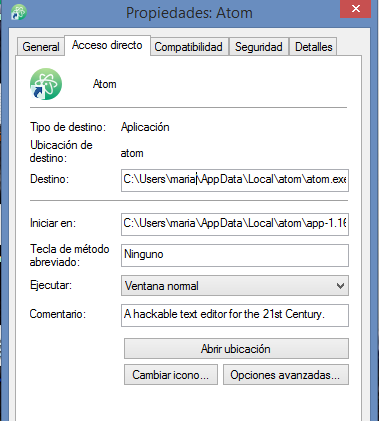
La instalación de Atom es bastante simple. Generalmente, podemos ir a https://atom.io y en la parte superior de la página debe ver un botón de descarga como se muestra aquí:



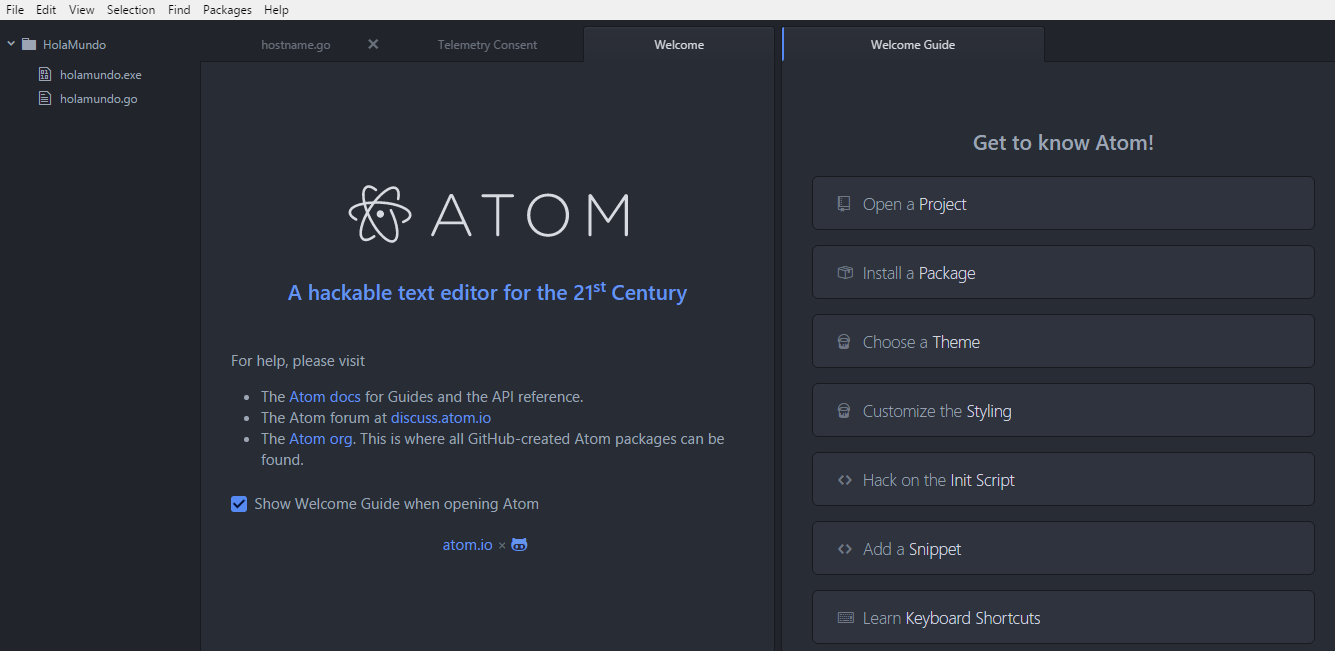
Después de descargar Atom para nuestro sistema operativo, pasamos a la instalación

 Elegimos las opciones por defecto e instalamos Atom.

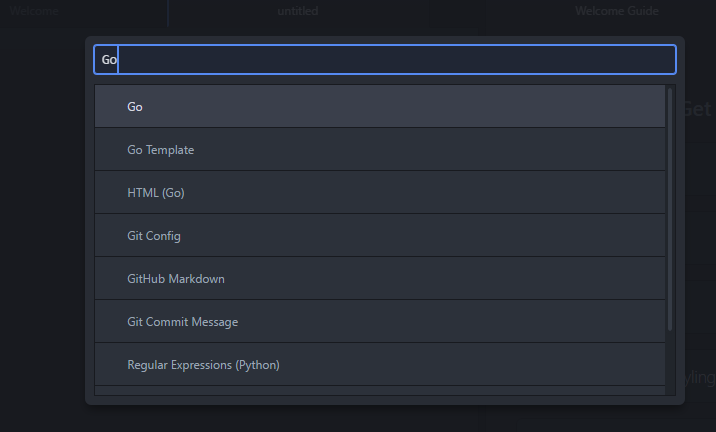




Una vez instalado Atom, abrimos el programa

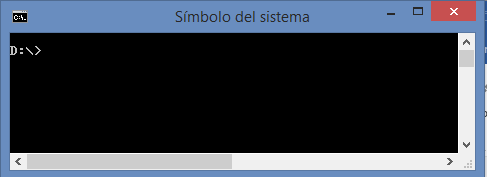


Es preciso que elijamos Go, ya que en Atom podemos programar en diferentes lenguajes de programación



## 4.5 Manual de usuario

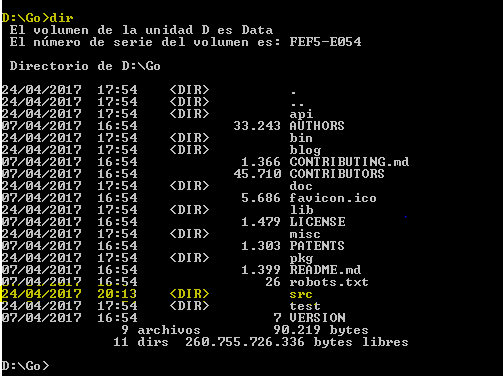
1. Abrir el símbolo de sistema CMD

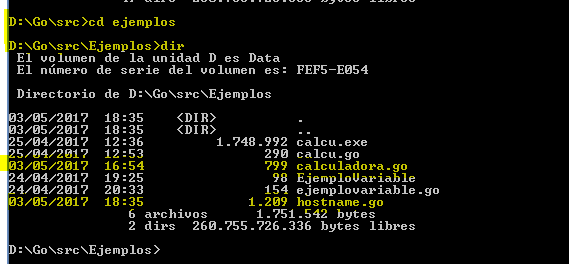


1. Navegar por todas las carpetas hasta finalmente estar en la carpeta que guarda los códigos GO/src/ejemplos.

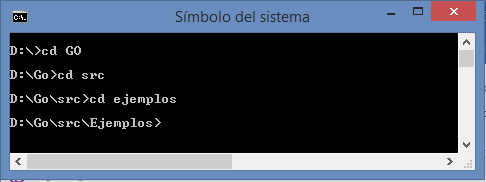
Tiempo de navegación(requerimientos de software **RFN04)**

Todo esto depende de la fluidez con que la persona sepa utilizar los comando CD O Dir (para listar el contenido de cada carpeta) menos de 3 minutos



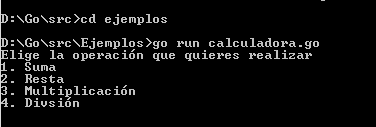


Navegación directa hasta la carpeta donde se guardan los programas



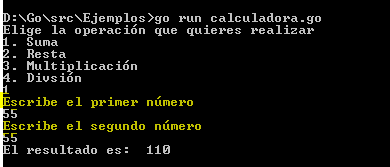
1. Ejecutamos el programa de la Calculadora

Aparece un Menú y podrá elegir la opción deseada, en este caso la opción 1 🡪suma



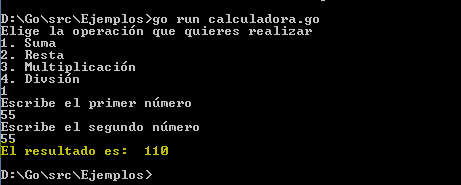
1. Introducir los dos argumentos

(Aquí se comprueba uno de los requerimientos funcionales **RF02)**



1. Respuesta del programa rápida y sin retraso **RFN07**

Fácil aprendizaje **RFN08**



**ANEXO**

Adicionalmente a este programa, hemos estado intentando hacer otro más complejo que vamos a mostrar a continuación:

Menos de sesenta líneas y donde mostramos el *hostname*, fecha y hora y probar más cosas:

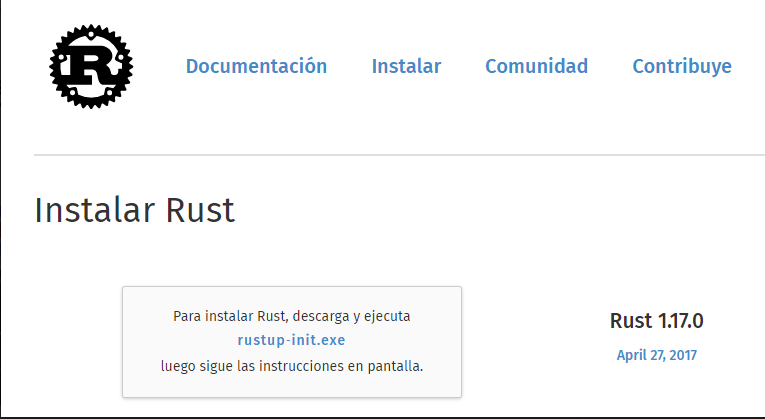


Aquí vemos cómo funciona el programa



# 5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología B (Rust)

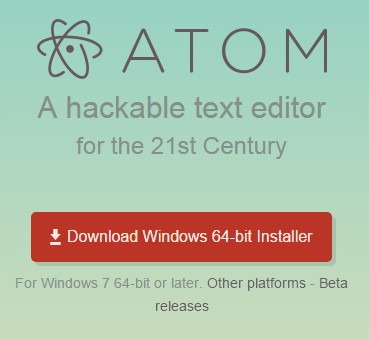
Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología B, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.



Nos metemos en la pagina oficial de rust para instalar el ejecutable de la ultima version disponible

<https://www.rust-lang.org/es-ES/install.html>

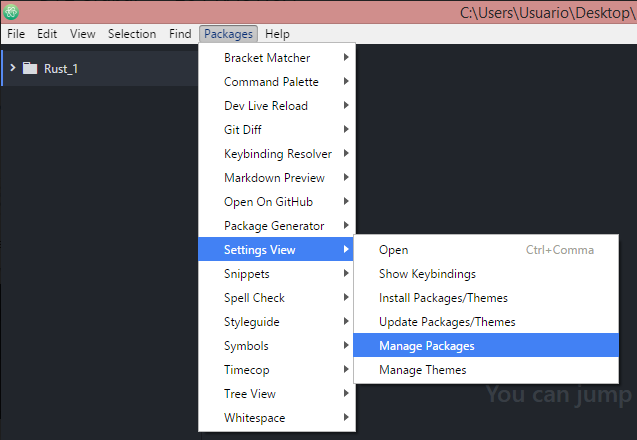
Podemos comprobar que se ha instalado Rust si ponemos en el cmd el comando "rustc"

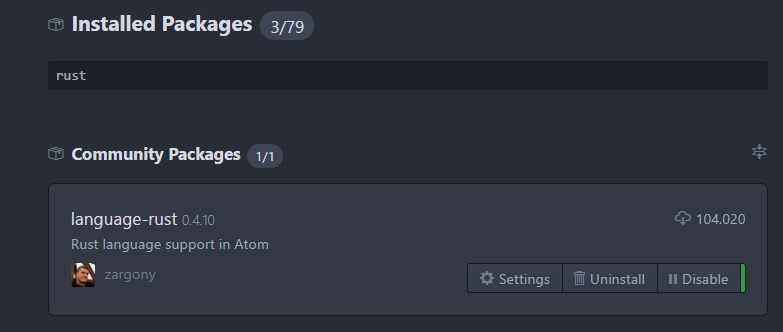


Para poder escribir el codigo necesitamos cualquier tipo de editor de texto, nosotros hemos usado Atom por su facilidad de uso

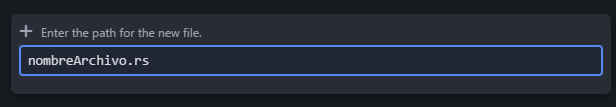
<https://atom.io/>

Descargamos la última versión disponible y nos ponemos con el programa para adaptarlo al nuevo lenguaje de programación de Rust



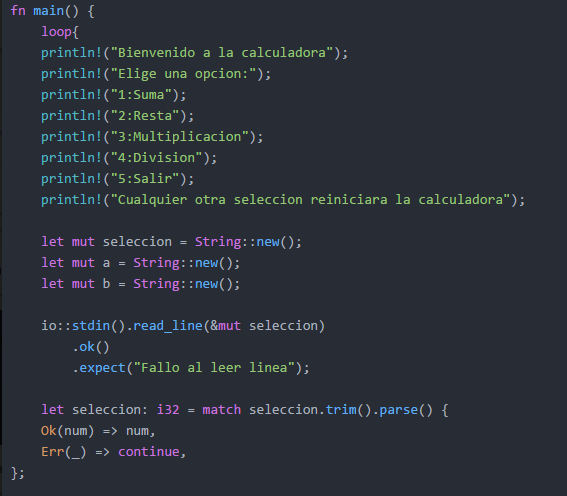


Lo primero es descargarse lo necesario para que Atom reconozca el lenguaje de programación de Rust, nosotros hemos descargado lo mas recomendado tanto por la comunidad por el número de descargas como en un video tutorial que seguimos a lo largo del proceso de instalación de Rust y Atom



Ahora hemos de preparar el lugar donde vamos a guardar todos los archivos generados por y para Rust (El creado por Atom, el ejecutable creado por Rust, etc)

Lo mas importante cuando creemos el nuevo fichero es acabarlo con .rs que es la extension del lenguaje Rust



Una vez listo todo lo anterior, comenzamos a programar:

fn main(){...} es el indicador de la funcion principal de nuestro programa

loop{...} es un bucle infinito, perfecto para usarlo cuando no sabemos cuantas veces queremos repetir una parte del codigo

println!("..."); es una sentencia que devuelve una cadena de caracteres

let mut nombreVariable permite crear variables

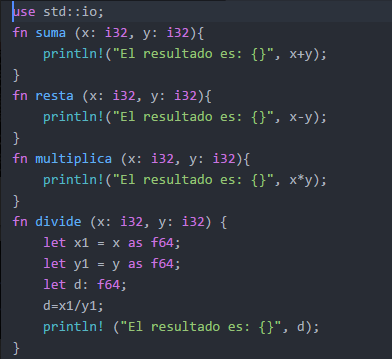
=  String::new(); crea las variables como una nueva cadena de caracteres

io::stdin().read\_line(&mut nombreVariable).ok().expect("Fallo al leer linea"); permite asignar a la variable en cuestion lo que el usuario meta por teclado

let nombreVariable: i32 = match nombreVariable.trim().parse(){...} permite cambiar el tipo de la variable (en este caso a i32)

Se muestra el bucle loop que contiene todo el programa, un menu por println! 3 variables (seleccion, a y b) de tipo new String

El valor de seleccion viene dado por teclado y lo convierte a entero, de no poder, el programa pasara a la siguiente iteracion (dara error y loop dara otra vuelta)



Esto es anterior a fn main() ya que contiene las funciones que llamaremos luego en el main

use std::io; permite que funcione la asignacion del teclado a una variable

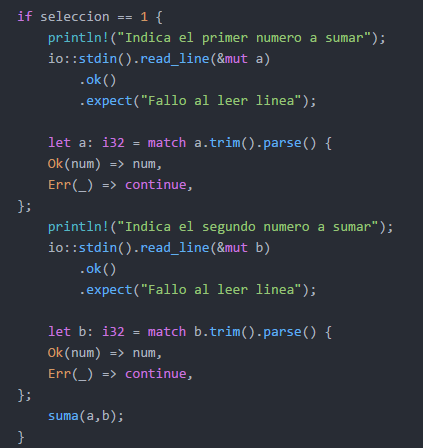
fn nombreFuncion(...){...}

indica los argumentos con los tipos de las variables y el codigo que se genera en la llamada de cada una de ellas

en nuestro caso hay 4 -> Suma, Resta, Multiplicacion y Division

Las tres primeras son identicas, pero la ultima es diferente

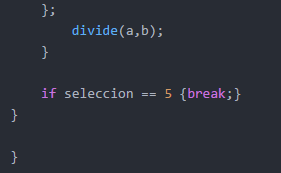
Para poder mostrar un resultado fiable (no un entero, que no permite decimales) leemos los enteros como float (as f64;) y hacemos lo mismo de antes



Volviendo al fn main() comenzamos a comparar, creando un if para cada una de las opciones, y de coincidir se ejecutara el codigo contenido en el if

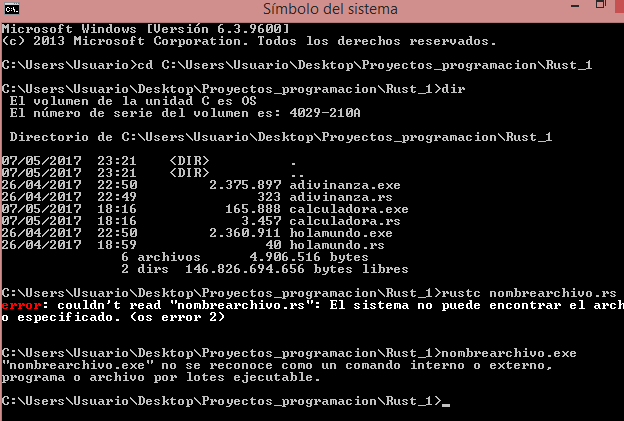
En todos los casos del 1 al 4 se mete por teclado el numero y el programa lo convierte a entero, y si hay error reinicia el programa

Si llega al final del codigo hace una llamada a la funcion, pasando como argumentos los valores introducidos por el teclado



La ultima opcion (If seleccion == 5 {break;}) permite salir del bucle loop y con eso dar por finalizado el programa

Como se ha comentado antes, si en algun momento se mete un tipo de dato erroneo, el programa se reiniciara de forma automatica, solo terminando cuando se introduzca la opcion correcta ("5")



Una vez preparado el codigo en Atom, vamos al cmd y a darle uso a rust

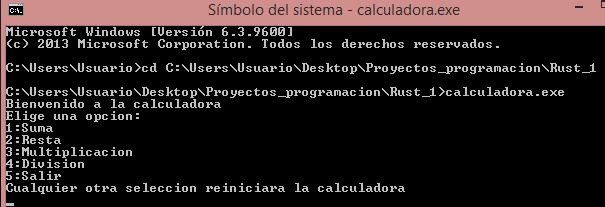
Comenzamos, con el comando cd, llevandonos al directorio donde se encuentre nuestro archivo .rs, Para asegurarnos de estar bien situados podemos usar el comando dir que muestra los archivos en ese directorio

Si queremos ejecutar el archivo que hemos creado en Atom utilizamos el comando: rustc y añadimos el nombre del archivo con extension .rs

Si todo va bien y el codigo no genera ningun fallo no pasara nada en apariencia

Si repetimos el comando dir podremos ver que e ha ejecutado un archivo con extension .exe

Ahora metemos el nombre del archivo y su extension .exe y comenzara la ejecucion de nuestro programa



# 6. Comparación de las dos implementaciones

## 6.1 Evaluación de los criterios en la implementación usando el lenguaje Go

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Criterio 1: Horas empleadas en el desarrollo del sistema | 8 |
| Criterio 2: Velocidad de ejecución de Código | 7 |
| Criterio 3: Documentación | 8 |
| Criterio 4: Coste del entorno de desarrollo | 9 |
| Criterio 5: Líneas de código | 8 |
| Criterio 6: Tiempo de instalación del programa | 8 |
| Criterio 7: Tiempo de aprendizaje | 7 |
| Criterio 8: Calidad del sistema | 8 |
| Criterio 9: Tiempo de arranque de tecnologías | 9 |

Criterios valorados de 1 al 10, donde 10 es la máxima puntuación según la facilidad para solucionar las cuestiones que se planteaban.

1: 30 horas. la instalación no es complicada. Hay que tener en cuenta que el lenguaje es nuevo, por lo tanto, lleva un tiempo el acostumbrarse a cada una de las situaciones que nos vamos encontrando mientras vamos avanzando.

2: rápida adaptación teniendo en cuenta que es un lenguaje nuevo.

3: la documentación es gratuita, libros en inglés. Posee buenas guías para ir adentrándonos en el lenguaje. mucha variedad de contenidos.

4: Un lenguaje gratuito, se pueden importar docs de otros usuarios a través de GitHub.

5: en total 35 líneas de código.

6: instalación en unos 30 min sencilla y rápida. hay que tener en cuenta la instalación de atom.

7: muchas horas de aprendizaje, 10 aproximadamente. leyendo, analizando, traduciendo, probando código, etc.

8: muy intuitivo. Tiene lo mejor de otros lenguajes por ejemplo de programas como c.

9: su ejecución es muy rápida.

## 6.2 Evaluación de los criterios en la implementación usando el lenguaje Rust

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Criterio 1: Horas empleadas en el desarrollo del sistema | 7 |
| Criterio 2: Velocidad de ejecución de Código | 6 |
| Criterio 3: Documentación | 8 |
| Criterio 4: Coste del entorno de desarrollo | 9 |
| Criterio 5: Líneas de código | 6 |
| Criterio 6: Tiempo de instalación del programa | 7 |
| Criterio 7: Tiempo de aprendizaje | 7 |
| Criterio 8: Calidad del sistema | 7 |
| Criterio 9: Tiempo de arranque de tecnologías | 9 |

Criterios valorados de 1 al 10, donde 10 es la máxima puntuación según la facilidad para solucionar las cuestiones que se planteaban.

1: 35 horas. lo primero decir que la instalación no es tan complicada, pero puede dar algunos errores de compatibilidad si no se está acostumbrado a instalar todo lo dicho anteriormente. El lenguaje es nuevo y hay que acostumbrarse a cada uno de sus opciones. aunque una vez hecho no es tan complejo, pero la primera vez que pruebas es largo y difícil de entender.

2: la dificultad ha residido sobretodo en la puesta en marcha. en la adaptación de nuestro conocimiento de al entorno de un lenguaje nuevo podemos decir que fue rápida.

3: la documentación es gratuita, libros/videos. Posee buenas guías para ir adentrándonos en el lenguaje. La única pega es que no existe mucha variedad de contenidos al ser un lenguaje nuevo.

4: coste 0. Un lenguaje gratuito y de código abierto.

5: en total 131 líneas de código. La base parecida a otros lenguajes, con la diferencia de sus códigos particulares.

6: instalación sencilla y rápida del leguaje, pero hay que tener en cuenta la instalación de atom y de visual studio para su correcto funcionamiento. Además de la variante cargo (anexo 1)

7: muchas horas de aprendizaje, leyendo, analizando, traduciendo, probando código, etc. Hemos probado y ejecutado otros programas para el mejor conocimiento de rust, para así poder ejecutar mejor nuestro proyecto. 12 horas aproximadamente. (Incluiremos 2 programas anexos)

8: los programas van bien en rúst. Es intuitivo, aunque menos en comparación con go.

9: una vez terminado el programa podemos decir que su ejecución es muy rápida.

# 7. Comparación de la implementación 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

| **CRITERIOS** | **LENGUAJE GO** | **LENGUAJE RUST** | **COMENTARIOS** |
| --- | --- | --- | --- |
| Criterio1: Horas empleadas en el desarrollo del sistema | 8 | 7 | Tiempo empleado parecido, aunque mayor en rust, debido al código y a un pequeño problema de compatibilidad que tuvimos en rust.(con Windows 10) |
| Criterio2:Velocidad de ejecución de Código | 7 | 6 | La diferencia reside en el código, aunque una vez realizado el programa su velocidad de ejecución es muy rápida en ambos lenguajes |
| Criterio3: Documentación | 8 | 8 | Documentación gratuita en ambos lenguajes.  Poseen la misma problemática con los manuales, ya que están en ingles |
| Criterio4: Coste del entorno de desarrollo | 9 | 9 | Ambos gratuitos. Algo de agradecer para el usuario. |
| Criterio5: Líneas de código | 8 | 6 | Son intuitivos pero es más complicada en rust. |
| Criterio6: Tiempo de instalación del programa | 8 | 7 | Instalación parecida, rust tarda un poco más por las cosas anexas que hay que instalarle para que todo vaya bien. |
| Criterio7: Tiempo de aprendizaje | 7 | 7 | El tiempo de aprendizaje es parecido, dependerá sobretodo de lo difícil que se nos del adaptarnos al nuevo lenguaje. |
| Criterio8: Calidad del sistema | 8 | 7 | Ambos intuitivos, pero go lo es un poco más. |
| Criterio 9: Tiempo de arranque de tecnologías | 9 | 9 | Programas rápidos a la hora de ejecutar. |

# 

# 8. Conclusiones

A partir de la información incluida en el apartado 7 y de la experiencia al realizar el trabajo, el grupo debe estar en condiciones de manifestar su opinión sobre la implementación del sistema utilizando ambas tecnologías, y debe plasmarla en este apartado, indicando las ventajas e inconvenientes más relevantes de utilizar una u otra tecnología para implementar el sistema.

**La puntuación final de Go:** 72 puntos

**La puntuación final de Rust:** 66 puntos

Podemos ver como go es ligeramente superior según nuestra experiencia. Esto se debe sobre todo a mayor dificultad a la hora de programar en rust.

Ambos poseen similitudes con c. pero go resulta más intuitivo. Además, hay que tener en cuenta el mayor número de información existente de go.

Documentación gratuita en ambos lenguajes, hay multitud de manuales, la gran mayoría de ellos en inglés. Aunque se puede buscar información en castellano. También hay videos tutoriales, aunque la gran mayoría están en inglés.

Aunque hay extensos manuales generales de ambas tecnologías, a la hora de trabajar con ellos y buscar dudas específicas costaba más solucionarlas.

Creemos que para alguien que vaya a emprender el aprendizaje de un lenguaje de programación, go será más intuitivo y rápido de aprender, aunque para un usuario más avanzado rust también tendrá su atractivo.

La diferencia principal reside en el código, aunque una vez realizado el programa su ejecución es muy rápida en ambos lenguajes.

Además del trabajo principal, vemos adecuado incluir un anexo de la herramienta cargo, además de dos pequeños programas hechos en dicha herramienta.