

Grado en Ingeniería Informática de Gestión y Sistemas de Información

Trabajo Fin de Grado

***RAMSÉS: Creación de nuevas funcionalidades,
reestructuración de la lógica aplicativa y puesta
en producción***

Alumno: Gaizka Valle Zabala

Director: Ismael Etxeberria Agiriano

Curso: 2021-2022

Fecha: 20 de mayo de 2022

AGRADECIMIENTOS

Gracias a mi familia, mascotas, amigos y compañeros por apoyarme a lo largo de los años, tanto en momentos buenos como malos a lo largo de todos estos años de carrera.

Gracias a mi compañera de proyecto, Mainer Pozo Yubero, y al director del mismo por la ayuda brindada para finalizar este proyecto.

RESUMEN

En este documento se trata sobre el proceso de puesta en producción y mejora de funcionalidades de la aplicación web RAMSÉS como Trabajo Fin de Grado en Ingeniería Informática y Gestión de la Información.

RAMSÉS es una herramienta de simulación de máquinas abstractas desarrollada para las prácticas y la evaluación de la asignatura de Lenguajes, Computación y Sistemas Inteligentes (LCSI) del segundo curso del Grado en Ingeniería Informática de Gestión y Sistemas de Información de la Escuela de Ingeniería de Vitoria-Gasteiz (UPV/EHU).

Este proyecto es la continuación directa del TFG de otra alumna, Estíbaliz Rodríguez de Yurre Vila, y se ha desarrollado en paralelo a otra compañera, Maider Pozo Yubero, mediante las instrucciones del director y beneficiario de este proyecto. Para el diseño de los elementos de la experiencia de usuario (UX) así como algunos retoques en la experiencia de usuario (UI) se ha colaborado con el diseñador gráfico Markel López de Arana Sanz. El nuevo diseño e implementación supone mejorar RAMSÉS cubriendo las funcionalidades existentes para este propósito, sentando las bases de futuras implementaciones, y al mismo tiempo aportando modernización en cuanto a tecnologías utilizadas.

El objetivo del proyecto es la implementación de los Autómatas con Pila (AP) y Máquinas de Turing (MT), así como nuevas funcionalidades para los Autómatas Finitos ya existentes. Por otro lado, se realizará a la puesta en producción de la aplicación mediante máquinas virtuales en un servidor, así como la reestructuración de la lógica aplicativa. Las principales tecnologías utilizadas para el desarrollo del proyecto han sido HTML, CSS, Node.js y JavaScript, con MySQL como SGBD. Para la puesta en producción se han utilizado Docker y Proxmox.

Se han completado los objetivos propuestos tanto de creación de nuevos autómatas como de reestructuración de la lógica aplicativa, así como la incorporación de nuevas funcionalidades. En estos momentos RAMSÉS está en producción en el servidor del departamento de LSI siendo accesible desde la Web.

ÍNDICE

Agradecimientos.....	2
Resumen.....	4
Índice	6
Índice de tablas	8
Índice de ilustraciones.....	10
1 Introducción	12
1.1 Título del proyecto	12
1.2 Inicio del proyecto	12
1.3 Objetivos	12
1.4 Alcance del proyecto.....	12
2 Contextualización	14
2.1 Teoría de autómatas	14
2.1.1 Tipos de autómatas	14
2.1.2 Operaciones sobre autómatas.....	15
3 Gestión del proyecto	18
3.1 Calendario del proyecto.....	18
3.2 Análisis de Riesgos	18
3.3 Análisis del proyecto.....	20
3.4 Diagrama de Gantt.....	22
3.5 Tareas del proyecto	22
3.6 Viabilidad financiera	23
3.7 Recursos humanos	24
3.8 Recursos materiales	24
3.9 Presupuesto	25
4 Desarrollo técnico.....	28
4.1 Tecnologías	28
4.1.1 HTML	28
4.1.2 CSS	28
4.1.3 JavaScript.....	29
4.1.4 Node.js.....	29
4.1.5 SVG	30
4.1.6 JSON.....	30
4.1.7 SQL.....	30
4.1.8 GIT.....	31

4.2	Herramientas	31
4.2.1	Visual Studio Code	31
4.2.2	MySQL Workbench	32
4.2.3	GitHub.....	32
4.2.4	Proxmox.....	33
4.2.5	Docker.....	33
4.2.6	PuTTY	33
4.2.7	Cisco Anyconnect.....	34
4.2.8	Diagrams.net	34
4.2.9	ProjectLibre	34
5	Análisis técnico	36
5.1	Integración continua	36
5.2	Base de datos	36
5.3	Puesta en producción	37
6	Trabajo realizado	40
7	Análisis del proyecto	42
7.1	Conclusiones	42
7.2	Líneas futuras.....	43
8	Bibliografía.....	44
	Anexos	48
	Anexo 1 - Manual de actualización del entorno de producción	48
	Anexo 2 - Manual de instalación	52
	2.1 Instalación de Node.js.....	52
	2.2 Instalación de MySQL.....	53
	2.3 Instalación de Git	54
	2.4 Configuración del repositorio de GitHub	55
	2.5 Instalación de Visual Studio Code	56

ÍNDICE DE TABLAS

Tabla 1 - Clasificación de riesgos	19
Tabla 2 - Posibles riesgos y soluciones	20
Tabla 3 - Programación del proyecto	21
Tabla 4 - Tareas y sus respectivos responsables	23
Tabla 5 - Costes de recursos humanos	24
Tabla 6 - Costes de amortización.....	25
Tabla 7 - Presupuesto del proyecto.....	26

ÍNDICE DE ILUSTRACIONES

Ilustración 1 - Tipos de autómatas en RAMSÉS	15
Ilustración 2 - Operaciones sobre autómatas en RAMSÉS	16
Ilustración 3 - Tareas del diagrama de Gantt	22
Ilustración 4 - Primera mitad del diagrama de Gantt	22
Ilustración 5 - Segunda mitad del diagrama de Gantt	22
Ilustración 6 - Logo HTML.....	28
Ilustración 7 - Logo CSS.....	28
Ilustración 8 - Logo JavaScript	29
Ilustración 9 - Logo Node.js	29
Ilustración 10 - Logo SVG	30
Ilustración 11 - Logo JSON	30
Ilustración 12 - Logo SQL	31
Ilustración 13 - Logo Git.....	31
Ilustración 14 - Logo Visual Studio Code	32
Ilustración 15 - Logo MYSQL Workbench	32
Ilustración 16 - Logo GitHub	32
Ilustración 17 - Logo Proxmox	33
Ilustración 18 - Logo Docker	33
Ilustración 19 - Logo PuTTY	33
Ilustración 20 - Logo Cisco AnyConnect.....	34
Ilustración 21 - Logo Diagrams.net.....	34
Ilustración 22 – Logo ProjectLibre	34
Ilustración 23 - Cómo se visualiza en GitHub los Merge Request	36
Ilustración 24 - Diagrama entidad relación de la base de datos	37
Ilustración 25 - Configuración de RAMSÉS en PuTTY	48
Ilustración 26 - Configuración de RAMSÉS en FileZilla	49
Ilustración 27 - Visualización de los archivos de RAMSÉS en FileZilla	49
Ilustración 28 - Instalador de Node.js.....	52
Ilustración 29 - Comprobación de versiones de Node.js y npm	53
Ilustración 30 - Pantalla de MySQL community installer.....	53
Ilustración 31 - Pantalla de instalación de Git	54
Ilustración 32 - Pantalla de <i>settings</i> en GitHub	55
Ilustración 33 - Pantalla de Visual Studio Code con RAMSÉS	56
Ilustración 34 - Pantalla de clonar con ssh en GitHub	57

1 INTRODUCCIÓN

1.1 TÍTULO DEL PROYECTO

RAMSÉS, creación de nuevas funcionalidades, reestructuración de la lógica aplicativa y puesta en producción.

RAMSÉS es acrónimo de *Recursive Abstract Machines Simulation EnvironmentS* [9].

1.2 INICIO DEL PROYECTO

El profesor de la universidad UPV/EHU Ismael Etxeberria Agiriano, planteó una serie de proyectos para realizar el TFG (Trabajo Fin de Grado). El proyecto consiste en la continuación de un proyecto de una antigua alumna, Estíbaliz Rodríguez de Yurre Vila, para crear una aplicación web para el uso educativo en el ámbito de la simulación de autómatas para el profesor que ofertaba los proyectos en cuestión.

Como había varias personas interesadas en el proyecto, se realizó una división general entre *frontend* y *backend* ya que éramos dos personas involucradas. Finalmente, después de unas cuantas reuniones se decidió que como se va a explicar a lo largo de este documento el autor desarrollaría la parte de *frontend* de la aplicación.

1.3 OBJETIVOS

Los objetivos de este proyecto son desarrollar y adaptar funcionalidades que ayuden al cliente de la aplicación en el uso del software para sus clases, así como la creación de un entorno de producción para que el cliente pueda seguir el proyecto en todo momento.

El escenario de trabajo es el de una pequeña empresa de software donde el cliente proporciona las especificaciones del software y va aportando los detalles a medida que se van implementando las diferentes partes.

1.4 ALCANCE DEL PROYECTO

Funcionalidades propuestas en el proyecto:

- ✓ Creación de entorno de producción.

- ✓ Puesta en producción del proyecto.
- ✓ Creación de manual de instalación.
- ✓ Añadir nuevos tipos de autómatas.
- ✓ Añadir nuevas funcionalidades sobre los autómatas.
- ✓ Adaptación de estructuras de datos y funcionalidades antiguas para modularizar o adaptarse a lo nuevo.
- ✓ Creación de nuevos modos en RAMSÉS: modo tutor y modo evaluación.
- ✓ Reestructuración de la lógica aplicativa para añadir nuevas funcionalidades.

2 CONTEXTUALIZACIÓN

En este capítulo se explicarán los conceptos de la teoría de autómatas referentes a las funcionalidades añadidas en la aplicación.

2.1 TEORÍA DE AUTÓMATAS

Antes de iniciar con los detalles del proyecto, vamos a revisar los conceptos y términos necesarios para describir y entender la aplicación y sus funcionalidades.

Una máquina abstracta o **autómata**, también conocido como computador abstracto, es un modelo teórico de un sistema computador de hardware o software usado en la teoría de autómatas.

2.1.1 TIPOS DE AUTÓMATAS

- Autómatas finitos (AF): se definen por tener un alfabeto, un conjunto de estados finitos, una función de transición, un estado inicial y uno o más estados finales. Se distinguen en 2 tipos AFD y AFND:
 - ✓ Autómata Finito Determinista (AFD): lo que distingue los AFD de otro tipo de autómata finito es el determinismo. Ser un AF que aplica el determinismo significa que estando en un estado y recibiendo una entrada (cadena de caracteres o símbolos) este podrá moverse solo un estado del conjunto de estados posibles [9].
 - ✓ Autómata Finito No Determinista (AFND): este tipo de AF no aplica el determinismo, es decir, estando en un estado y recibiendo una cadena de entrada, el autómata podrá moverse de un estado a más de un estado en caso de ser posible. Por otro lado, admitirá épsilon (ϵ) como símbolo en sus transiciones [9].
- Autómata con Pila (AP): este tipo de autómata es reconocedor, lo que significa que acepta lenguajes que no pueden aceptar los autómatas finitos. Una diferencia respecto a los AF es que albergan una unidad de memoria llamada **pila** [9].
- Máquina de Turing (MT): las MT cuentan con una cinta de lectura/escritura, la cinta va desplazando el cabezal de lectura a izquierda o a derecha, borrando el

símbolo sobre el que se encuentra y sustituyéndolo por otro perteneciente al alfabeto de salida. En el caso de RAMSÉS distinguimos Máquina de Turing Reconocedora (MTR) y Máquina de Turing Calculadora (MTC) a la hora de ejecutarlas [9].

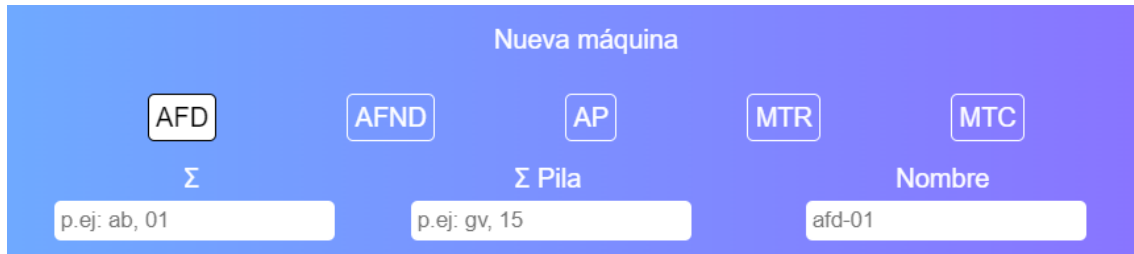


Ilustración 1 - Tipos de autómatas en RAMSÉS

2.1.1.2 OPERACIONES SOBRE AUTÓMATAS

Una vez introducidos los conceptos básicos sobre máquinas abstractas o autómatas, vamos a explicar las funcionalidades principales a añadir a la aplicación.

En los siguientes puntos vamos a explicar las operaciones sobre los autómatas finitos y lo que significa cada una de ellas.

- ✓ **Invertir un AFD:** invertir un AFD consiste en obtener un autómata que reconozca el lenguaje inverso. Para ello se hace que tenga un solo estado final y se cambia la dirección de sus transiciones, es decir si una transición va del estado 1 al estado 2 con la letra R, al invertir la transición iría del estado 2 al estado 1 con la letra R. Finalmente, el estado inicial pasará a ser un estado final y por su parte el estado final se convertirá en inicial [7].
- ✓ **Convertir un AFD en AFD total:** para que un autómata finito determinista sea considerado total, tiene que tener una función de transición total, es decir tiene que tener definidas todas las transiciones posibles. Todo AFD tiene un AFD total, basta con añadir un estado trampa al AFD [7].
- ✓ **Complementar AFD:** esta operación está pensada para la prueba libre a la hora de dibujar autómatas, y consiste en convertir los estados finales en no-finales y los estados no-finales en finales [7]. Si el autómata no es total se considerará un error.

- ✓ **Complementar AFD total:** para complementar un AFD total, lo primero es comprobar si dicha máquina es determinista y total. En caso de cumplir las condiciones previas, se convierten los estados no-finales en finales y a su vez los estados finales en no-finales [7].
- ✓ **Transformar un AFND en AFD:** cuando se habla del determinismo se refiere a la unicidad de la ejecución de cálculo. Para convertir un AFD en un AFN que reconozca el mismo lenguaje, se usa el algoritmo de construcción de subconjuntos. Este consiste en utilizar una tabla de transiciones de un AFN, donde cada entrada es un conjunto de estados; en la tabla de transiciones de un AFD, cada entrada es tan solo un estado. La idea general tras la construcción AFN a AFD es que cada estado de AFD corresponde a un conjunto de estados del AFN. El AFD utiliza un estado para localizar todos los posibles estados en los que puede estar el AFN después de leer cada símbolo de la entrada [7].



Ilustración 2 - Operaciones sobre autómatas en RAMSÉS

3 GESTIÓN DEL PROYECTO

3.1 CALENDARIO DEL PROYECTO

El comienzo de este proyecto es el 13 de octubre de 2021 y la fecha estimada para su finalización es el 12 de mayo de 2022, día en el que el autor termina las prácticas de empresa. Teniendo en cuenta días festivos, según el calendario de Álava y las vacaciones de navidad donde hubo vacaciones, la cantidad de días laborales disponibles son 136 días. Teniendo en cuenta que cada día laboral se estima una media de 3 horas de trabajo diarias, la cantidad de horas totales disponibles para el proyecto son un total de 408 horas.

Los días festivos durante el proyecto son:

- ✓ 1 de noviembre: Día de Todos los Santos
- ✓ 6 de diciembre: Día de la Constitución Española
- ✓ 8 de diciembre: Inmaculada Concepción
- ✓ 24 de diciembre - 7 de enero: Vacaciones de Navidad
- ✓ 14 de abril: Jueves Santo
- ✓ 15 de abril: Viernes Santo
- ✓ 18 de abril: Lunes de Pascua
- ✓ 28 de abril: San Prudencio

3.2 ANÁLISIS DE RIESGOS

En el paso previo a la planificación del proyecto se han analizado los posibles riesgos que pueda sufrir el desarrollo del mismo. De esta forma, se puede crear un plan de contingencia o medidas preventivas en caso de que finalmente se materialicen dichos riesgos. Para ello, se analiza y valora los riesgos teniendo en cuenta su probabilidad y gravedad.

Tanto la probabilidad como la gravedad se clasificaron con uno de los siguientes valores: baja, media o alta. La combinación de los valores de la probabilidad y la gravedad, dotarán al riesgo de una puntuación. La puntuación supondrá un valor numérico entre 1 y 5.

Valor	Equivalencia
1	Riesgo trivial: No hay necesidad de tomar acción.
2	Riesgo tolerable: No es grave, pero es preferible reducir la probabilidad de que ocurra.
3	Riesgo moderado: Se deben tomar medidas preventivas para reducir la probabilidad de que este riesgo ocurra.
4	Riesgo importante: Debe ser prioritario tomar medidas preventivas para evitar este riesgo.
5	Riesgo grave: No se puede continuar o empezar ningún tipo de acción sin antes haber solucionado el riesgo.

TABLA 1 - CLASIFICACIÓN DE RIESGOS

Aplicando los criterios explicados anteriormente, procedemos a listar y analizar los posibles riesgos y medidas preventivas a aplicar en ellos en este proyecto:

Riesgo	Gravedad	Probabilidad	Valoración del riesgo	Medida de prevención
Aumento de los requisitos	Media	Alta	4	A lo largo de todo el desarrollo se modularizará todo el código en pequeños módulos para que las consecuencias se mantengan al mínimo posible con cada cambio.
Baja por enfermedad en el programador	Baja	Baja	1	Con la Covid-19 como máxima preocupación actual se tomarán las medidas acordadas en todo momento por las autoridades sanitarias, así como aplicar

				el teletrabajo en la medida de lo posible.
Rotura del pc del desarrollador	Media	Baja	2	Al tener 2 ordenadores, cada 2 semanas se realizará una copia de todo el material necesario y actualizado en el pc de reserva.
Bloqueo en el desarrollo de una tarea	Media	Media	3	Se realizarán reuniones cada 2 semanas con el director del TFG para intentar subsanar todas las dudas que puedan surgir en la implementación.
Bloqueo por baja del cliente	Baja	Baja	2	Al principio del proyecto y después de cada reunión se le pedirá al cliente un documento con las especificaciones requeridas para la ocasión y con vistas a futuro para así tener material en caso de tener algún problema para asistir a reuniones el cliente.
Tareas no programadas	Baja	Alta	3	Se asignará un día a la semana para salirse de la especificación inicial e intentar subsanar dichas tareas.
Pérdida de datos	Alta	Baja	3	Creación frecuente y programada de copias de seguridad en varios dispositivos diferentes.

TABLA 2 - POSIBLES RIESGOS Y SOLUCIONES

3.3 ANÁLISIS DEL PROYECTO

Hay un periodo inicial en donde se deciden las funcionalidades o especificaciones deseadas para el proyecto, así como reuniones con el cliente con una periodicidad de 2 semanas en donde se añaden o modifican las especificaciones según

las necesidades vistas con el paso del tiempo. Por otro lado, cada tarea tiene un responsable asignado para su implementación.

	Tarea	Dependencia
0	Proyecto RAMSÉS	
1	Análisis del proyecto	
1.1	Estudio de requerimientos	
1.2	Reparto de funcionalidades	1.1
1.3	Declaración de alcance y objetivos	1.2
2	Diseño	1.3
2.1	Diseño de requisitos	1.3
2.2	Análisis del proyecto anterior	2.1
2.3	Análisis de software y requerimientos	2.1
3	Implementación	2.3
3.1	Puesta en producción	2.3
3.2	Adaptación de la lógica	3.1
3.3	Implementación de funcionalidades	3.2
4	Etapas de pruebas con cliente	3.3
4.1	Reunión con cliente	3.3
4.2	Implementación de cambios sugeridos	4.1
5	Memoria	4.2

TABLA 3 - PROGRAMACIÓN DEL PROYECTO

3.4 DIAGRAMA DE GANTT

Siguiendo el análisis realizado en el apartado anterior hemos generado el Diagrama de Gantt:

ID	Nombre	Duración	Inicio	Terminado	Predecesores
1	Análisis del proyecto	9 days?	13/10/21 8:00	25/10/21 17:00	
2	Estudio de requerimientos	7 days?	13/10/21 8:00	21/10/21 17:00	
3	Reparto de funcionalidades	1 day?	22/10/21 8:00	22/10/21 17:00	2
4	Declaración de alcance y objetivo	1 day?	25/10/21 8:00	25/10/21 17:00	3
5	Diseño	22 days?	26/10/21 8:00	24/11/21 17:00	
6	Diseño de requisitos	10 days?	26/10/21 8:00	8/11/21 17:00	4
7	Análisis de proyecto anterior	7 days?	9/11/21 8:00	17/11/21 17:00	6
8	Análisis de software y requerimientos	5 days?	18/11/21 8:00	24/11/21 17:00	7
9	Implementación	96 days?	25/11/21 8:00	7/04/22 17:00	
10	Puesta en producción	21 days?	25/11/21 8:00	23/12/21 17:00	8
11	Adaptación de la lógica	20 days?	24/12/21 8:00	20/01/22 17:00	10
12	Implementación de funcionalidades	55 days?	21/01/22 8:00	7/04/22 17:00	11
13	Etapas de pruebas con cliente	11 days?	8/04/22 8:00	22/04/22 17:00	
14	Reunión con cliente	7 days?	8/04/22 8:00	18/04/22 17:00	12
15	Implementación de cambios sugeridos	4 days?	19/04/22 8:00	22/04/22 17:00	14
16	Memoria	15 days?	25/04/22 8:00	13/05/22 17:00	15

Ilustración 3 - Tareas del diagrama de Gantt

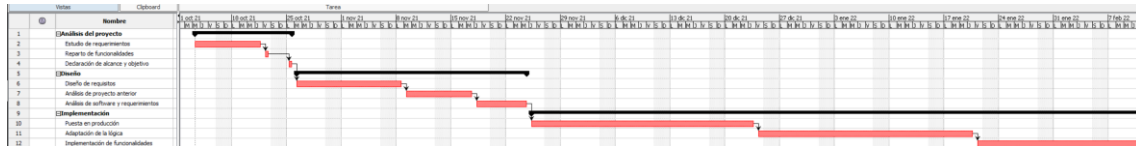


Ilustración 4 - Primera mitad del diagrama de Gantt

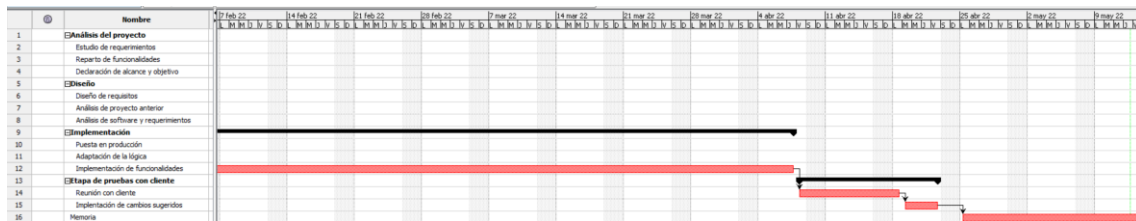


Ilustración 5 - Segunda mitad del diagrama de Gantt

3.5 TAREAS DEL PROYECTO

En el siguiente apartado se detallarán las tareas a realizar, así como la persona a cargo de cada una de estas. Las tareas han sido repartidas entre las 4 personas que han participado en este proyecto.

- ✓ Gaizka Valle: Programador junior
- ✓ Mainer Pozo: Programador junior
- ✓ Markel López de Arana: Diseñador gráfico

✓ Ismael Etxeberria: Cliente del proyecto

Tarea	Responsable
Definición de objetivos y requisitos	Ismael Etxeberria
Pruebas de funcionalidades añadidas	Ismael Etxeberria
Diseño de nuevos elementos gráficos	Markel López de Arana
Puesta en producción (entorno)	Gaizka Valle
Puesta en producción (base de datos)	Maidier Pozo
Mantenimiento de la base de datos	Maidier Pozo
Añadir nuevos tipos de autómatas	Gaizka Valle
Crear nuevos tipos de simulaciones para los nuevos tipos de autómatas	Gaizka Valle
Descargar autómatas en XML	Maidier Pozo
Alterar lógica de las transiciones para poder añadir más de una entre estados	Gaizka Valle
Facilitar creación de estados iniciales	Gaizka Valle
Invertir autómatas finitos	Gaizka Valle
Convertir un autómata finito en total	Gaizka Valle
Complementar un autómata finito	Gaizka Valle
Complementar un autómata finito total	Gaizka Valle
Tratamiento de errores	Gaizka Valle

TABLA 4 - TAREAS Y SUS RESPECTIVOS RESPONSABLES

3.6 VIABILIDAD FINANCIERA

En este apartado analizaremos los gastos que deberá tener en cuenta la empresa y en base a estos se generará un presupuesto.

3.7 RECURSOS HUMANOS

Para determinar los costes de recursos humanos en el proyecto, nos basamos en los convenios laborales publicados en el BOE. Con esto como referencia, se listan los costes según el rol desarrollado en el proyecto así como los cálculos totales del mismo.

- ✓ Programador junior: 15 €/hora
- ✓ Diseñador gráfico: 12 €/hora

Rol	Cantidad	Salario/hora	Importe total
Programador junior	2	15,00 €/hora	32.640,00 €
Diseñador gráfico	1	12,00 €/hora	120,00 €
Total			32.760,00 €

TABLA 5 - COSTES DE RECURSOS HUMANOS

3.8 RECURSOS MATERIALES

En el siguiente apartado, se definirán las amortizaciones de los costes materiales, siendo estos costes tanto de software (licencias) como de hardware. Los costes de amortización se tendrán en cuenta a 2 años de horas trabajadas.

Siendo la cantidad de horas laborales 1750 horas, según el Boletín Oficial del Estado, las horas de amortización a dos años serán 3500 horas y a un año 1750 horas. Para el cálculo del coste unitario de amortización y la amortización total se utilizarán unas fórmulas matemáticas.

Coste Unitario de Amortización = Coste unitario / Horas de Amortización

Amortización total = horas de uso (408) * Amortización unitaria * Cantidad

Todo el software utilizado para el desarrollo técnico del proyecto es de uso gratuito por lo cual no se nombrará ningún tipo de software en estos costes.

Elemento	Coste unitario	Cantidad	Años amortización	Amortización unitaria	Amortización total
Ordenador Acer	1.015,00 €	1	2	0,29 €	118,32 €
Monitor LG	325,00 €	1	1	0,18 €	73,44 €
Total					191,76 €

TABLA 6 - COSTES DE AMORTIZACIÓN

Según investigaciones realizadas por el autor, en el día en el que se realiza este documento, la tarifa promedio de internet ronda 35 € al mes, teniendo en cuenta un año y realizando los cálculos explicados anteriormente, estimamos un coste de internet de 97,92 €.

Por otro lado, se ha calculado un gasto en electricidad para un año de la misma manera, siendo este de 179,52 €.

Añadiendo estos costes a los calculados anteriormente, el coste total de los recursos materiales es de 469,20 €.

3.9 PRESUPUESTO

Teniendo en cuenta los costes de recursos humanos y materiales calculados anteriormente, se calculará el presupuesto del proyecto.

Para ello se tiene en cuenta un 21 % de IVA y unos beneficios de 15 %.

Concepto	Importe
Recursos humanos	32.760,00 €
Recursos materiales	469,20 €
Suma	33.229,20 €
Beneficios	5.113,24 €
Subtotal	38.342,44 €
IVA	8.051,91 €
Total	46.394,35 €

TABLA 7 - PRESUPUESTO DEL PROYECTO

4 DESARROLLO TÉCNICO

4.1 TECNOLOGÍAS

4.1.1 HTML

HTML (*Hypertext Markup Language*) es un lenguaje de marcado que se usa más comúnmente para crear sitios web, con etiquetas o marcas que definen su estructura y mostrarlo en un navegador web [14].

Este tipo de archivos tienen extensión .html en la mayoría de los casos, aunque también se pueden encontrar con la extensión .htm y se leen e interpretan para mostrar al usuario mediante un navegador web (Chrome, Firefox...).



Ilustración 6 - Logo HTML

4.1.2 CSS

CSS (*Cascading Style Sheets*) es un lenguaje de estilos utilizado para describir la apariencia o estilo de documentos HTML o XML.

Hay un estándar definido por parte de W3C (*World Wide Web Consortium*) [13] y en el caso de este proyecto se ha utilizado mediante el *framework Bootstrap*.



Ilustración 7 - Logo CSS

4.1.3 JAVASCRIPT

JavaScript es un lenguaje de programación orientado a objetos, interpretado y multiplataforma que se puede usar para crear código para las partes de servidor y cliente [12].

Es responsable de las interacciones con elementos HTML, como eventos activados por el usuario como por ejemplo pulsar un botón o validar un formulario o input.



Ilustración 8 - Logo JavaScript

4.1.4 NODE.JS

Node.js es un entorno de ejecución multiplataforma para JavaScript formado con V8, el motor de JavaScript de Google. Es una de las tecnologías más utilizadas y permite ejecutar JavaScript en el servidor empleando un solo hilo para la ejecución mediante un bucle de eventos asíncronos [6].

Por otro lado, gracias al *Node Package Manager* (NPM) [5], tienes acceso a las librerías de acceso gratuito creadas por la vasta comunidad de usuarios que usa esta tecnología.



Ilustración 9 - Logo Node.js

4.1.5 SVG

SVG (*Scalable Vector Graphics*) es un estándar web basado en XML para definir gráficos basados en vectores en páginas web [17]. Su característica más importante es que proporciona marcado para describir rutas, formas y texto dentro de una ventana gráfica, almacenando toda esta información en un fichero de texto plano modificable.



Ilustración 10 - Logo SVG

4.1.6 JSON

JSON (*JavaScript Object Notation*) se refiere a un archivo de extensión .json con un funcionamiento muy similar a XML para la transferencia de datos estructurados entre un servidor de Web y una aplicación Web [9].

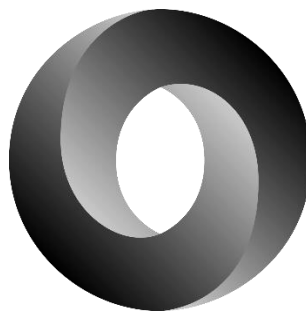


Ilustración 11 - Logo JSON

4.1.7 SQL

SQL (*Structured Query Language*) es el lenguaje estándar utilizado para la manipulación (operaciones CRUD) y descarga de datos en bases de datos relacionales [20][21].



Ilustración 12 - Logo SQL

4.1.8 GIT

Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para gestionar todo tipo de proyectos mediante repositorios.

Permite ramificar el desarrollo y mantener el código en una línea de progreso separada de la línea de progreso principal para aislar posibles errores o cambios de prueba.



Ilustración 13 - Logo Git

4.2 HERRAMIENTAS

4.2.1 VISUAL STUDIO CODE

Visual Studio Code es un editor de código multiplataforma y gratuito redefinido y optimizado para construir y depurar aplicaciones web.

Es uno de los editores más utilizados en el mundo ya que permite instalar plugin de terceros [24] que ayudan mucho a la hora de trabajar cómo integración directa con software de control de versiones cómo podría ser en este caso Git.

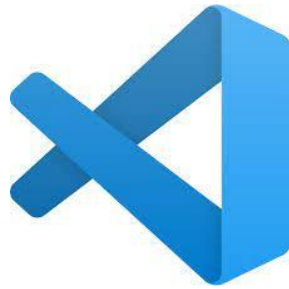


Ilustración 14 - Logo Visual Studio Code

4.2.2 MYSQL WORKBENCH

MySQL *Workbench* es una herramienta visual unificada que sustituye la línea de comandos para trabajar con bases de datos relacionales.



Ilustración 15 - Logo MYSQL Workbench

4.2.3 GITHUB

Se trata de un servicio de control de versiones y desarrollo de software colaborativo basado en Git. En este caso se ha utilizado el servicio de gestión y host de repositorio privado e integración continua, pero tiene muchos otros servicios cómo despliegue automático o automatización de procesos básicos [25].



Ilustración 16 - Logo GitHub

4.2.4 PROXMOX

Proxmox (*Proxmox Virtual Environment*) es un entorno de virtualización de servidores de código abierto. Usa una distribución de GNU/Linux basada en Debian, con una versión modificada del Kernel Ubuntu LTS y permite el despliegue y la gestión de máquinas virtuales y contenedores [26].



Ilustración 17 - Logo Proxmox

4.2.5 DOCKER

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Con esta finalidad empaqueta software en contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución.

La gran ventaja de usar este software es que se puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno sin el problema de saber si el código se ejecutará [3][8].



Ilustración 18 - Logo Docker

4.2.6 PUTTY

PuTTY es un emulador de terminal gratuito que admite varios protocolos de red tal como SSH.



Ilustración 19 - Logo PuTTY

4.2.7 CISCO ANYCONNECT

Cisco AnyConnect ofrece múltiples servicios de seguridad para ayudar a habilitar y proteger cómo podría ser el servicio utilizado en este proyecto para conectarse a la UPV, la VPN [28].



Ilustración 20 - Logo Cisco AnyConnect

4.2.8 DIAGRAMS.NET

Diagrams.net es una aplicación gratuita integrada con Google para crear todo tipos de diagramas siendo una buena alternativa a *Visual Paradigm* [29].



Ilustración 21 - Logo Diagrams.net

4.2.9 PROJECTLIBRE

ProjectLibre es un software de administración y gestión de proyectos de código abierto para controlar y planificar todos los niveles de un proyecto de forma fácil y visual [30].

ProjectLibreTM

Ilustración 22 – Logo ProjectLibre

5 ANÁLISIS TÉCNICO

5.1 INTEGRACIÓN CONTINUA

La integración continua o *Continuous Integration* [31] es la metodología de automatizar la integración o actualización de código en un proyecto de software con varios contribuidores. Para esta finalidad se ha utilizado la fusión (*Merge Request*) que proporciona GitHub para fusionar el código de una rama a la rama de desarrollo o principal.

El proceso para este método es el siguiente: en el repositorio de código correspondiente se dispone de una rama principal a la cual se restringe subir código si no es a través de fusión. Por lo cual, cada integrante del grupo crea ramas a partir de la principal para desarrollar la funcionalidad específica y al subir la rama y una vez pasen los test unitarios solicita la fusión a otro integrante del proyecto. Entonces el integrante al que se le solicita la fusión revisa los cambios propuestos y acepta la fusión o comenta posibles modificaciones.

Finalmente, en caso de aceptar la fusión el código se fusiona a la rama principal y se borra la rama creada para el desarrollo y así se consigue que todo el mundo conozca todas las funcionalidades y la detección de posibles errores antes de pasar a producción el código.

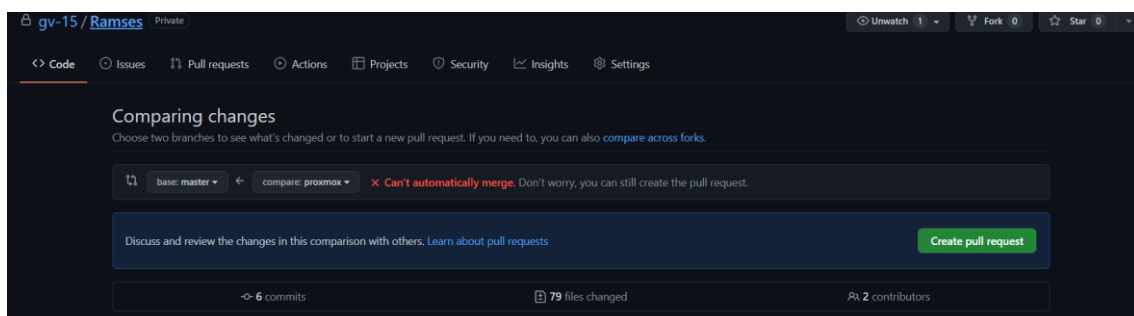


Ilustración 23 - Cómo se visualiza en GitHub los Merge Request

5.2 BASE DE DATOS

La parte de la base de datos, al ser correspondiente a la parte de *frontend* del proyecto ha sido desarrollada casi en su totalidad por mi compañera de proyecto, Mainer Pozo Yubero, aunque al necesitar cuadrar las estructuras de datos hemos

colaborado para ello. De todas maneras, será explicado mayormente en la parte del proyecto correspondiente a su trabajo.

El diagrama de entidad relación resultante de la colaboración entre ambos se muestra a continuación:

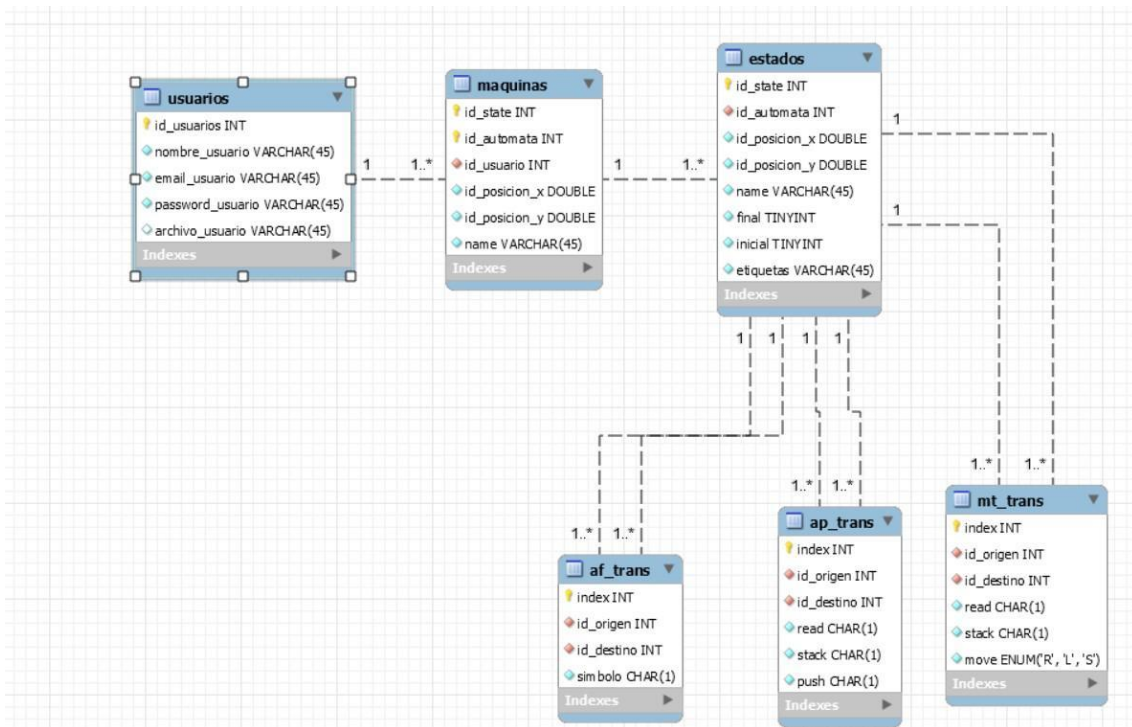


Ilustración 24 - Diagrama entidad relación de la base de datos

5.3 PUESTA EN PRODUCCIÓN

La puesta en producción de esta aplicación se ha realizado siguiendo un proceso de varios pasos.

El primer paso ha sido preparar el entorno de producción en Proxmox, un software para almacenar máquinas virtuales. Para ello se ha instalado Node.js, Docker y una base de datos MySQL en la máquina virtual (Ubuntu) [4].

El segundo paso seguido ha sido subir los archivos de la aplicación a un directorio en dicha máquina virtual. Se añaden en un directorio aparte para poder actualizar la aplicación automáticamente con cada cambio realizado.

El tercer y último paso es configurar los archivos de Docker con todas las dependencias necesarias dentro del proyecto así cómo enlazando con la base de datos

creada en la máquina virtual. En estos archivos de Docker es el lugar donde indicaremos que se vuelva a subir la aplicación cada vez que se realice un cambio el directorio del proyecto [8].

Cómo punto extra cabe decir que siempre hay que tener 2 versiones del proyecto, ya que las redirecciones cambian dentro de la máquina virtual respecto al entorno de desarrollo, teniendo que añadir siempre “**ramses**” delante de cada dirección dentro del código.

Se puede ver la aplicación corriendo en el entorno de producción en la siguiente dirección: <https://lsi.vc.ehu.eus/ramses/>.

6 TRABAJO REALIZADO

En este apartado se detallará el trabajo realizado en la aplicación mediante el uso de las herramientas y metodologías descritas en los apartados anteriores.

La primera funcionalidad añadida al proyecto, fue implementada junto a la colaboración de mi compañera de proyecto, Maider Pozo, la puesta en producción (Proxmox y Docker) para el acceso mediante la web de la aplicación.

Una vez en marcha el entorno de producción y ya de manera individual, se comenzó con la primera fase de implementación se añadieron nuevos tipos de autómatas, Autómatas con Pila (AP) y Maquinas de Turing (MT). En el caso de las Maquinas de Turing, se implementaron con una distinción de tipos, calculadoras y reconocedoras.

En este punto del proyecto se necesitaban crear nuevas interfaces para completar la implementación de los nuevos tipos (diálogos, transiciones...), así que aprovechando el momento y con la colaboración del diseñador gráfico, Markel López de Arana, el diseñador gráfico con el que se ha colaborado en el proyecto, se crearon nuevas interfaces y se modernizo el aspecto gráfico de varios sitios en la aplicación, como podría ser la pantalla de *login*.

Con los nuevos autómatas en funcionamiento, se vio que eran necesarios cambios en la lógica aplicativa de la aplicación para poder implementar las nuevas funcionalidades propuestas y poder realizar las mejoras propuestas para las ya existentes. Los cambios realizados en la lógica, mejoran la modularización de la aplicación y arreglan ciertos problemas a la hora de dibujar transiciones (ahora permite más de una entre 2 estados).

Por otro lado, se han implementado métodos para que en un futuro sea más fácil la ampliación hacia nuevas funcionalidades en RAMSÉS, así como un botón para poder borrar y crear un nuevo autómata desde 0 ya que hasta el momento había que recargar la pagina y no era nada intuitivo para el usuario.

Finalmente, una vez arreglados los problemas con la lógica se implementaron nuevas funcionalidades para los autómatas finitos deterministas (AFD). Las funcionalidades añadidas a los AFD son las siguientes:

- ✓ Invertir el AFD
- ✓ Convertir el AFD en un AFDT
- ✓ Complementar un AFDT
- ✓ Complementar un AFD

7 ANÁLISIS DEL PROYECTO

7.1 CONCLUSIONES

Este proyecto es una continuación de un proyecto anterior para crear una aplicación web que pueda sustituir a JFLAP creado por otra alumna, Estibaliz Rodríguez Yurre, como software de apoyo para los docentes de la asignatura Lenguajes, Computación y Sistemas Inteligentes (LCSI) en el Grado de Ingeniería Informática de Gestión y Sistemas de Información.

Habiendo realizado este proyecto en conjunto con otra compañera que se encargaba del *backend* de la aplicación, Maider Pozo Yubero, y con los consejos del profesor beneficiario de este software y a la vez director de este proyecto, se han modificado y agregado funcionalidades para la puesta en producción y fácil proceso de actualización de la aplicación de cara a añadir funcionalidades en el futuro. Por otro lado, se ha colaborado con un diseñador gráfico lo que ha supuesto un enfoque más global de todo el trabajo necesario para crear una aplicación.

Se ha realizado un análisis del conjunto global de especificaciones y se han implementado las funcionalidades que se han considerado más útiles o importantes con un resultado satisfactorio.

RAMSÉS ha supuesto un aprendizaje extra de algunas herramientas y tecnologías. El autor ya había trabajado con anterioridad algunas de las herramientas para el desarrollo web, pero con esto ha podido mejorar en su uso así como descubrir nuevos usos o integraciones con otras nuevas. Por otro lado, la interacción para detallar las especificaciones con cliente ha supuesto un gran aprendizaje de cara al mundo laboral aprendiendo a cómo adaptarse en todo momento a cualquier cambio o mejora propuesta.

Tras la finalización del proyecto, teniendo en cuenta que ha sido un desarrollo autónomo a empresa y con el tiempo dedicado al mismo, se considera el proyecto de manera satisfactoria.

7.2 LÍNEAS FUTURAS

La modificación de cómo se actualiza el entorno de producción mediante los archivos de Docker puede ser muy interesante. En un entorno laboral normalmente se suele configurar para que cuando se realiza un *merge* a la rama principal del repositorio de código automáticamente haga un *deploy* al entorno de producción, así sería mucho más rápido y cómodo actualizar los cambios.

Debido a varias limitaciones técnicas y sobre todo recursos a la hora de entre otras muchas cosas realizar todo como si fuera *single page application*, migrar la aplicación a un *framework* debería de ser uno de los principales problemas a tratar. Hoy en día prácticamente ninguna aplicación web se realiza sin un *framework*. Teniendo conocimiento sobre los 3 principales que son *Vue.js*, *React* y *Angular*, se propone migrar la aplicación a Angular ya que mejoraría todo de manera drástica tanto futuro como con lo realizado hasta el momento.

Añadir nuevas funcionalidades a autómatas finitos como podría ser el paso de AFND a AFD o las interacciones entre 2 AF como sería buscar la intersección debería ser el camino a seguir en el futuro más próximo, así como mejorar la ejecución de los AP o MT ya que actualmente tienen algunas fallas en casos específicos.

Al ser una aplicación pensada para el ámbito estudiantil, sería muy interesante añadir un modo tutorial donde te vaya explicando paso a paso las operaciones a realizar. Por otro lado, podríamos añadir el modo examen o evaluación donde se irían añadiendo u ocultando funcionalidades según el nivel del alumno o lo deseado por el profesor.

Para terminar, aunque no esté terminada la aplicación para su uso total, estaría bien realizar una sesión con alumnos para ver su *feedback* real, así como hacer una pequeña encuesta para descubrir dónde se puede mejorar gracias al *feedback* proporcionado por lo que al final serán los usuarios finales/reales de esta aplicación.

8 BIBLIOGRAFÍA

[1] MDN Web Docs. Web Components. Concepts and usage, reference and tutorials.
https://developer.mozilla.org/es/docs/Web/Web_Components

[2] Gabriel, T. (s. f.). *The Definitive Guide to Docker Swarm*. Gabrieltanner.
<https://gabrieltanner.org/blog/docker-swarm>

[3] B. (2021, 2 septiembre). *Dockerize Node.js Express and MySQL example - Docker Compose*. BezKoder.
<https://www.bezkoder.com/docker-compose-nodejs-mysql/>

[4] Nym, F. (2021, 9 febrero). *How to Create Dockerized NodeJS with MySQL Database*. DEV Community.
<https://dev.to/frasnym/how-to-create-dockerized-nodejs-with-mysql-database-1o44>

[5] *npm: package*. (2012, 20 abril). Npm. <https://www.npmjs.com/package/package>

[6] *Index / Node.js v16.14.2 Documentation*. (s. f.). Nodejs.Org.
<https://nodejs.org/dist/latest-v16.x/docs/api/>

[7] Asignatura *Lenguajes, Computación y Sistemas Inteligentes* (LCSI). Plan de estudio Grado en Ingeniería Informática de Gestión y Sistemas de Información (UPV/EHU).
https://www.ehu.es/es/web/guest/grado-ingenieria-informatica-de-gestion-y-sistemasdeinformacionalava/creditasyasignaturasporcurso?p_redirect=consultaAsignatura&p_cod_proceso=egr&p_anyo_acad=20200&p_ciclo=X&p_curso=2&p_cod_asignatura=26021

[8] Asignatura *Administración de Sistemas*. Plan de estudio del Grado en Ingeniería Informática de Gestión y Sistemas de Información (UPV/EHU).
<https://www.ehu.es/es/web/guest/grado-ingenieria-informatica-de-gestion-y-sistemas-de-informacion-alava/creditos-y->

- [21] Asignatura Diseño de Bases de Datos. Plan de estudio del Grado en Ingeniería Informática de Gestión y Sistemas de Información (UPV/EHU).
https://www.ehu.es/es/grado-ingenieria-informatica-de-gestion-y-sistemas-de-informacionalava/creditasyasignaturasporcurso?p_redirect=consultaAsignatura&p_cod_proceso=egr&p_anyo_acad=20210&p_ciclo=X&p_curso=3&p_cod_asignatura=26026
- [22] *Tutorial de SQL.* (s. f.). DesarrolloWeb.Com.
<https://desarrolloweb.com/manuales/tutorial-sql.html>
- [23] *Visual Studio Code - Code Editing. Redefined.* (2021, 3 noviembre). Visual Studio Code.
<https://code.visualstudio.com/>
- [24] Chacón, J. L. (2022, 18 abril). *Los mejores plugins de Visual Studio Code para desarrolladores web.* Profile Software Services.
<https://profile.es/blog/visual-studio-code-plugins/>
- [25] Microsoft. (s. f.). GitHub Documentation.
<https://docs.github.com/es>
- [26] *Proxmox VE Documentation Index.* (s. f.). Proxmox VE Documentation Index.
<https://pve.proxmox.com/pve-docs/>
- [27] *Conectarse a la instancia de Linux desde Windows mediante PuTTY - Amazon Elastic Compute Cloud.* (s. f.). Conectarse PuTTY.
https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/putty.html
- [28] *Red privada virtual VPN (acceso seguro desde internet) - Vicegerencia de Tecnologías de la Información y las Comunicaciones - UPV/EHU.* (s. f.). Vicegerencia de Tecnologías de la Información y las Comunicaciones.
<https://www.ehu.es/es/web/ikt-tic/vpn>
- [29] *Diagram Software and Flowchart Maker.* (s. f.). Diagrams.net.
<https://www.diagrams.net/>
- [30] *#1 Alternative to Microsoft Project Open Source | Projectlibre.* (s. f.). ProjectLibre.
<https://www.projectlibre.com/>
- [31] Atlassian. (s. f.-a). *¿En qué consiste la integración continua?*
<https://www.atlassian.com/es/continuous-delivery/continuous-integration>

Anexos

ANEXO 1 - MANUAL DE ACTUALIZACIÓN DEL ENTORNO DE PRODUCCIÓN

En primer lugar, tenemos que poder conectarnos a RAMSÉS mediante dos métodos: PuTTY y FileZilla.

En el caso de PuTTY en la configuración añadimos lo siguiente:

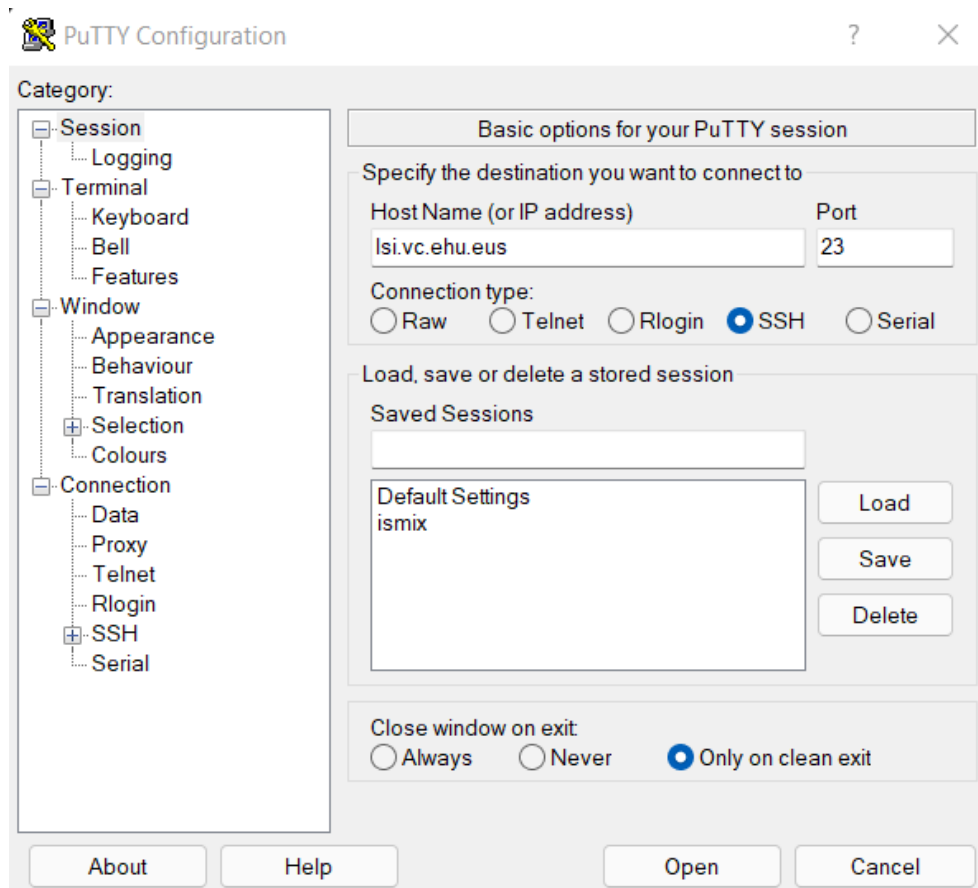


Ilustración 25 - Configuración de RAMSÉS en PuTTY

Respecto a FileZilla entramos en la pestaña “*Gestor de sitios*”, al igual que hemos realizado anteriormente en el PuTTY introducimos el host (lsi.vc.ehu.eus), el usuario (ramses) y la contraseña. En este caso el protocolo a utilizar sería SFTP.

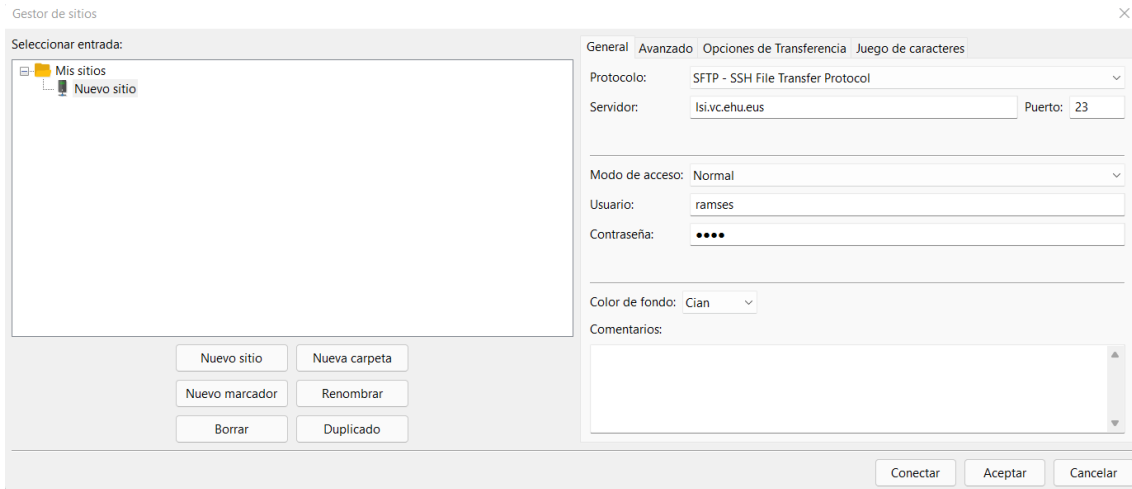


Ilustración 26 - Configuración de RAMSÉS en FileZilla

En la siguiente ilustración podemos observar el contenido de la máquina virtual en FileZilla.

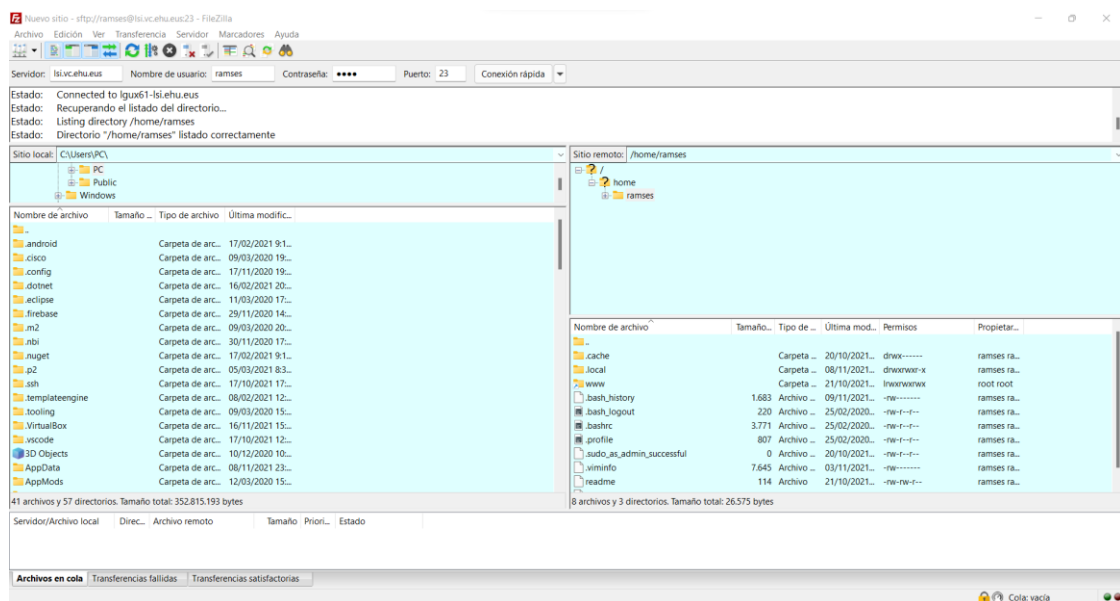


Ilustración 27 - Visualización de los archivos de RAMSÉS en FileZilla

Una vez conseguido el acceso, en FileZilla se buscan los archivos a sustituir y se arrastran las nuevas versiones, sobrescribiendo las anteriores.

En todos los sitios que se necesita redirección (como podría ser el *path* a una imagen o un cambio de página) hay que poner /ramses/ al inicio en la versión que se sube a la máquina virtual. Esto pasa mayormente en los CSS, button-array.js y el botón de volver al *login* en el *home* por lo cual hay que prestar mucha atención a que las

redirecciones se hacen correctamente antes de subirlo ya que si no no se verá correctamente.

El otro archivo que tiene cambios y redirecciones en comparación a la versión de desarrollador para funcionar correctamente en el entorno preparado para producción es el *index.js* donde hay que tener especial cuidado si se sobrescribe.

Al subir mediante FileZilla los archivos modificados se actualizan solos, pero en el caso de cambiar parte de la configuración entrar en el PuTTY y ejecutar los siguientes comandos:

```
$ docker-compose down  
$ docker-compose up -d
```


ANEXO 2 - MANUAL DE INSTALACIÓN

El presente anexo es una versión ampliada y más actualizada del manual de instalación proporcionada en el Anexo 1 de la memoria del TFG de Estibaliz Rodríguez de Yurre [9]. En este anexo se detallan los pasos a seguir para poder correr RAMSÉS desde 0.

Para ejecutar la aplicación final es necesario tener instalado tanto *Node.js* como MySQL en el ordenador. Los módulos de terceros de *Node.js* se incluyen en el entregable de la aplicación, por lo que no es necesario volver a descargarlos, pero en caso de necesitarlos por algún error se instalan con uno de los siguientes comandos:

```
$ npm i
```

```
$ npm install
```

2.1 INSTALACIÓN DE NODE.JS

Descargar el instalador (*.msi*) desde la página oficial de descargas de Node.js, <https://nodejs.org/es/download/>.

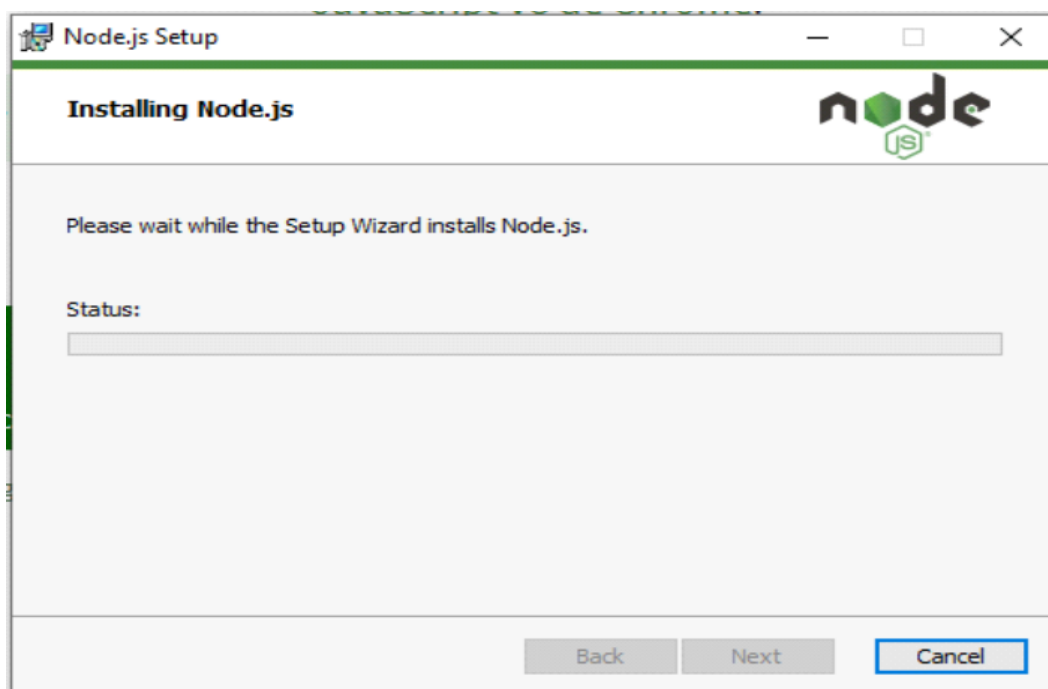


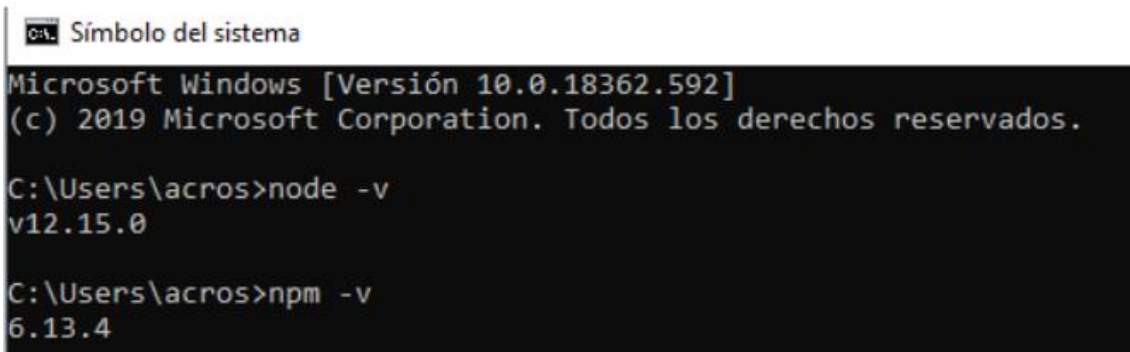
Ilustración 28 - Instalador de Node.js

Una vez terminada la instalación, se puede comprobar la versión instalada utilizando el siguiente comando:

```
$ node -v
```

Para comprobar la versión del *node package manager* instalada utilizaremos el comando:

```
$ npm -v
```



```

C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.18362.592]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\acros>node -v
v12.15.0

C:\Users\acros>npm -v
6.13.4
  
```

Ilustración 29 - Comprobación de versiones de Node.js y npm

2.2 INSTALACIÓN DE MYSQL

Se ha instalado la última versión estable de MySQL Community, que incluye MySQL Workbench como interfaz gráfica para la gestión CRUD de la BD.



Ilustración 30 - Pantalla de MySQL community installer

En este punto también se crea una base de datos, y un usuario (nombre de usuario y contraseña) para acceder como administrador. Estas credenciales son las que se utilizarán para realizar la conexión a la BD desde el código de la aplicación.

En este mismo documento hay un diagrama entidad relación para crear la base de datos, pero en los entregables del proyecto hay una copia llamada ramses.sql para su importación completa.

2.3 INSTALACIÓN DE GIT

La importancia de la instalación de Git para Windows radica en la utilización de un repositorio de GitHub (cuya inicialización se explica en el siguiente apartado), ya que es necesaria la creación de una clave SSH.

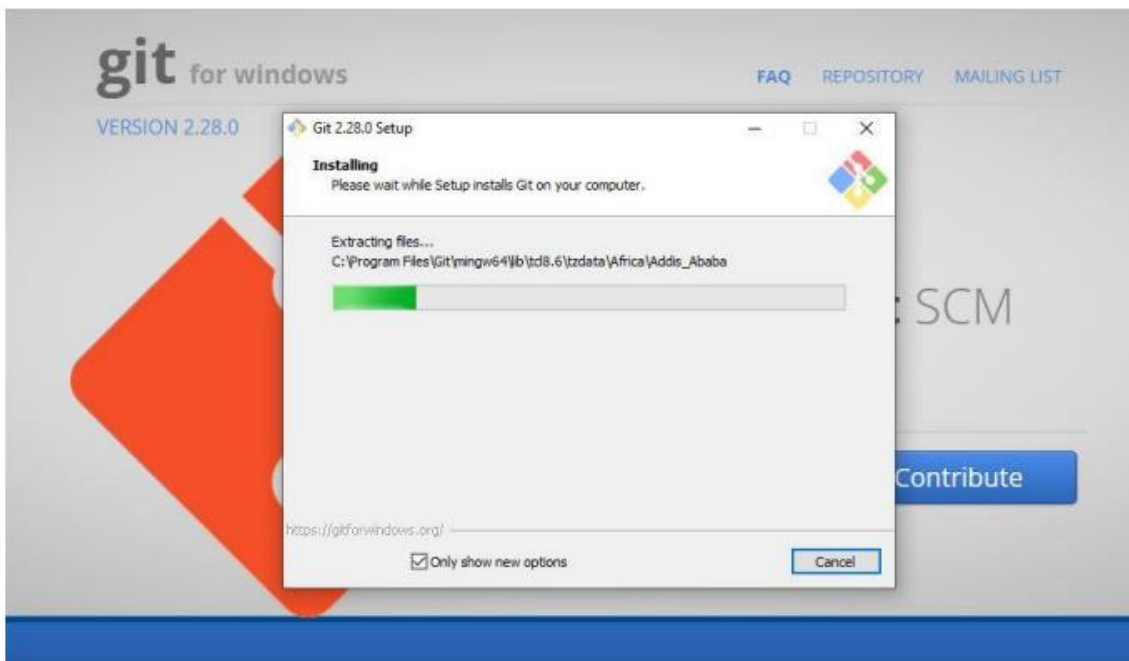


Ilustración 31 - Pantalla de instalación de Git

Con Git se instalan varias aplicaciones distintas, pero solo es necesario utilizar una de ellas, *Git Bash*.

Git Bash permite utilizar herramientas Linux *Bash* con Git en la línea de comandos.

2.4 CONFIGURACIÓN DEL REPOSITORIO DE GITHUB

El primer paso es la creación de una cuenta en GitHub, tras lo que procederemos a generar una clave SSH con los siguientes comandos en la terminal:

```
$ ssh-keygen -t rsa -b 2048
```

Después ejecutaremos el siguiente comando para activar el agente de ssh para no tener que añadirla cada vez que encendemos el ordenador:

```
$ ssh-add
```

Podemos comprobar que se han generado correctamente con:

```
$ ssh-add -l
```

El último paso con respecto a la clave SSH es añadirla a nuestra cuenta de GitHub en *Settings* > *SSH and GPG Keys*.

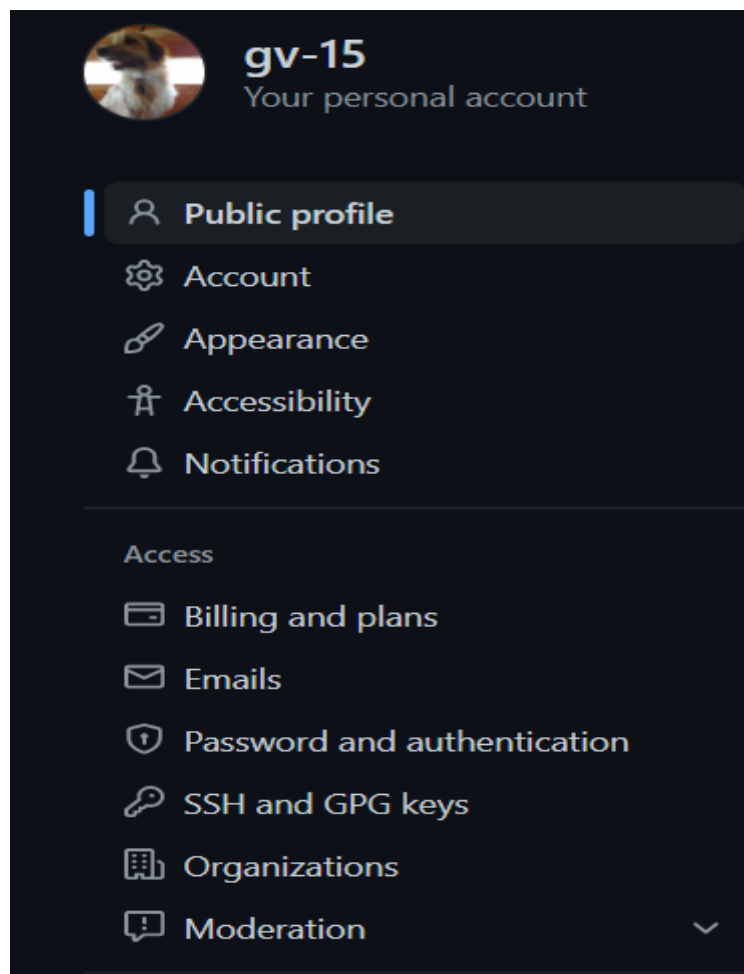


Ilustración 32 - Pantalla de *settings* en GitHub

En caso de que el repositorio ya esté creado se necesitaría acceso a él por ser privado, y en caso de crearlo por primera vez para seguir avanzando podría crearse de la manera habitual, pero es más recomendable y seguro de esta forma a la hora de clonarlo ya que de esta forma no habrá fallas de seguridad.

2.5 INSTALACIÓN DE VISUAL STUDIO CODE

Visual Studio Code (generalmente abreviado cómo VS Code) es el editor de código escogido para realizar el proyecto ya que es gratuito y con grandes añadidos de una comunidad muy activa. Se descarga desde la página oficial, <https://code.visualstudio.com/>.

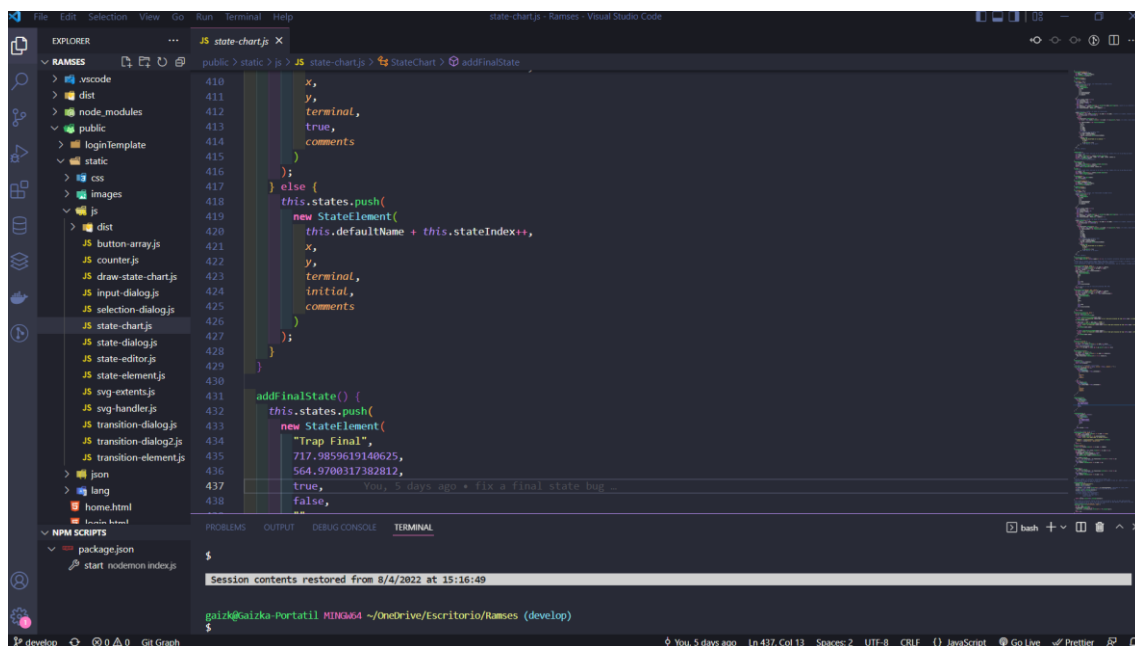


Ilustración 33 - Pantalla de Visual Studio Code con RAMSÉS

Por otro lado, de tener configurado el proyecto con *ssh* y añadidas de forma correcta, desde el mismo GitHub se puede clonar el repositorio directamente.

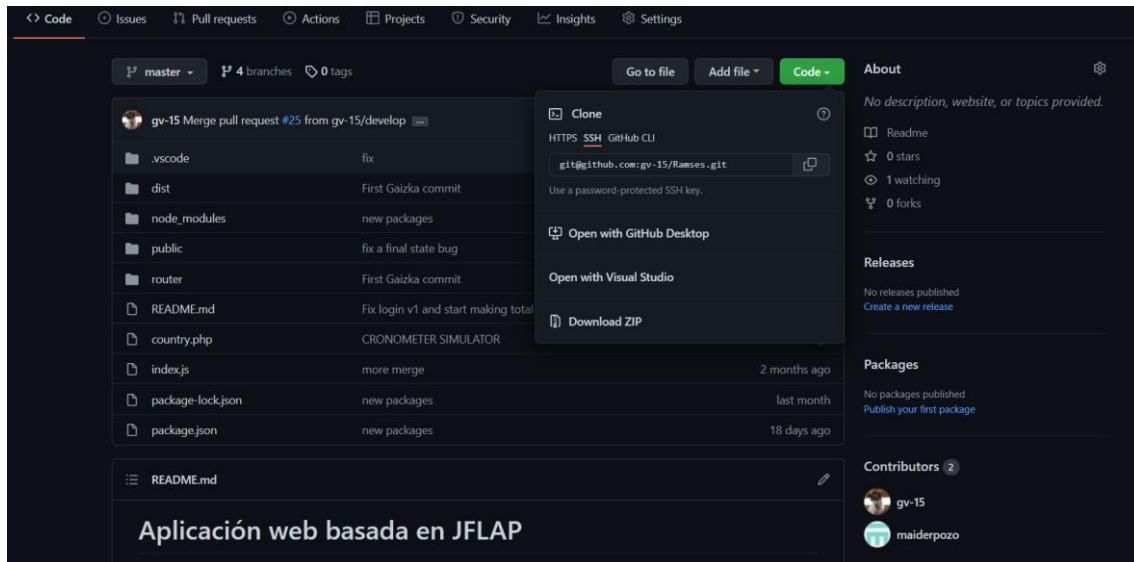


Ilustración 34 - Pantalla de clonar con ssh en GitHub