

## Trabajo de Fin de Grado

---

# Agenda inteligente para la gestión eficiente de objetivos personales

TITULACIÓN: Grado en Ingeniería Informática  
AUTOR: Carlos Octavio Rodríguez del Toro

---

TUTORIZADO POR:  
José Fortes Gálvez

Fecha: enero de 2025

## SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D/D<sup>a</sup> Carlos Octavio Rodríguez del Toro, autor/a del Trabajo de Fin de Título Agenda inteligente para la gestión eficiente de objetivos personales, correspondiente a la titulación Grado en Ingeniería Informática, plan 41, en colaboración con la empresa/proyecto (indicar en su caso)

### S O L I C I T A

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida, haciendo constar que

[X] se autoriza / [ ] no se autoriza, la grabación en audio de la exposición y turno de preguntas.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

- [ ] Se ha iniciado o hay intención de iniciarla (defensa no pública).  
 [X] No está previsto.

Y para que así conste firma la presente. (fecha en firma electrónica)

El/la estudiante

RODRIGUEZ DEL  
TORO CARLOS  
OCTAVIO -  
Fdo.: \_\_\_\_\_  
45344594W

Firmado digitalmente por  
RODRIGUEZ DEL TORO  
CARLOS OCTAVIO -  
45344594W  
Fecha: 2025.01.17 14:22:48 Z

A llenar y firmar **obligatoriamente** por el/la/los/las tutor/a/es/as.  
En relación a la presente solicitud, se informa: (firmar donde corresponda)

Positivamente (en caso de detección de copia, esta firma quedará invalidada)

Negativamente (la justificación en caso de informe negativo deberá incluirse en el TFT05)

*Jose Fortes Galvez*

Fdo.: \_\_\_\_\_

Fdo.: \_\_\_\_\_

**DIRECTORA DE LA ESCUELA DE INGENIERÍA INFORMÁTICA**

## **Resumen**

En un mundo cada vez más saturado de actividades y responsabilidades, la gestión eficiente del tiempo se convierte en un desafío crucial. Muchas personas no logran alcanzar sus objetivos personales debido a una mala planificación, a una distribución inadecuada de sus tareas diarias o incluso por llegar a olvidarlos. La motivación detrás de este trabajo es proporcionar una solución innovadora y personalizada para este problema mediante el desarrollo de una aplicación web accesible y segura que funcione como una agenda inteligente que programe las tareas que conforman los objetivos del usuario. Para ello, se ha seguido una metodología Kanban, desarrollando una solución full stack que emplea Python para el backend y HTML, CSS y JavaScript para el frontend.

## **Abstract**

In a world increasingly saturated with activities and responsibilities, efficient time management has become a crucial challenge. Many people fail to achieve their personal goals due to poor planning, inadequate distribution of their daily tasks, or even due to forgetting them. The motivation behind this work is to provide an innovative and personalized solution to this problem by developing an accessible and secure web application that functions as an intelligent agenda that schedules the tasks that make up the user's objectives. To achieve this, a Kanban methodology was followed, developing a full stack solution that uses Python for the backend and HTML, CSS, and JavaScript for the frontend.

# Índice general

<b>1. INTRODUCCIÓN .....</b>	<b>11</b>
1.1. ESTADO ACTUAL.....	11
1.2. OBJETIVOS INICIALES.....	11
1.3. COMPETENCIAS ESPECÍFICAS CUBIERTAS .....	12
1.3.1. <i>C11</i> .....	12
1.3.2. <i>C12</i> .....	13
1.4. APORTACIONES.....	13
1.5. CONCEPTOS Y TERMINOLOGÍAS DEL PROYECTO .....	13
<b>2. MARCO TEÓRICO .....</b>	<b>14</b>
2.1. ESTUDIO DEL MERCADO .....	14
2.1.1. <i>Reclaim AI</i> .....	14
2.1.2. <i>Motion</i> .....	15
2.1.3. <i>Trevor AI</i> .....	16
2.1.4. <i>GoalsOnTrack</i> .....	16
2.1.5. <i>Habitica</i> .....	17
2.1.6. <i>Conclusiones</i> .....	17
2.2. ACCESIBILIDAD EN LA WEB .....	18
2.2.1. <i>WCAG 2.2</i> .....	18
2.2.2. <i>Conclusiones</i> .....	20
2.3. SEGURIDAD EN LA WEB .....	20
2.3.1. <i>OWASP</i> .....	20
2.3.2. <i>Conclusiones</i> .....	21
2.4. NORMATIVA Y LEGISLACIÓN .....	21
<b>3. ANÁLISIS.....</b>	<b>22</b>
3.1. FLUJO DE FUNCIONAMIENTO.....	22
3.2. METODOLOGÍA .....	23
3.3. TECNOLOGÍAS .....	24
3.4. HERRAMIENTAS .....	25
3.5. REQUISITOS .....	26
3.5.1. <i>Requisitos funcionales</i> .....	26
3.5.2. <i>Requisitos no funcionales</i> .....	27
3.6. HISTORIAS DE USUARIO .....	27
<b>4. DISEÑO.....</b>	<b>30</b>
4.1. OBJETIVO DEL DISEÑO .....	30
4.2. HERRAMIENTA PARA EL DISEÑO.....	31
4.3. DISEÑO ESTÉTICO .....	32
4.3.1. <i>Colores</i> .....	32
4.3.2. <i>Tipografía</i> .....	32
4.3.3. <i>Botones</i> .....	33
4.3.4. <i>Iconos</i> .....	33
4.3.5. <i>Formularios</i> .....	33
4.3.6. <i>Cuadros modales</i> .....	33
<b>5. DESARROLLO .....</b>	<b>34</b>
5.1. ORGANIZACIÓN .....	34
5.2. ESTRUCTURA DEL PROYECTO .....	35

5.3.	BASE DE DATOS .....	36
5.4.	VISTA DE USUARIO .....	42
5.4.1.	<i>Header</i> .....	42
5.4.2.	<i>Footer</i> .....	44
5.4.3.	<i>Portal de Usuario</i> .....	45
5.4.4.	<i>Registro</i> .....	47
5.4.5.	<i>Inicio de Sesión</i> .....	53
5.4.6.	<i>Objetivos</i> .....	56
5.4.7.	<i>Nuevo Objetivo</i> .....	59
5.4.8.	<i>Objetivo</i> .....	62
5.4.9.	<i>Nueva Tarea</i> .....	66
5.4.10.	<i>Tarea</i> .....	69
5.4.11.	<i>Agenda</i> .....	72
5.4.12.	<i>Progreso</i> .....	75
5.4.13.	<i>Notificaciones</i> .....	79
5.4.14.	<i>Perfil</i> .....	82
5.4.15.	<i>Ayuda</i> .....	87
5.4.16.	<i>Aviso Legal</i> .....	89
5.4.17.	<i>Política de Privacidad</i> .....	91
5.4.18.	<i>Error</i> .....	93
5.5.	PLANIFICACIÓN .....	95
5.6.	OTRAS IMPLEMENTACIONES .....	99
5.6.1.	<i>Almacenamiento de imágenes</i> .....	99
5.6.2.	<i>Inicio de sesión obligatorio para acceder a las funcionalidades</i> .....	99
5.6.3.	<i>Lectores de pantalla</i> .....	100
<b>6.</b>	<b>CONCLUSIONES .....</b>	<b>101</b>
6.1.	CONCLUSIONES.....	101
6.2.	TRABAJOS FUTUROS .....	103
	<b>BIBLIOGRAFÍA.....</b>	<b>105</b>

# Índice de ilustraciones

Ilustración 1. Diseños de la interfaz de usuario en Figma .....	31
Ilustración 2. Tablero de Trello durante el desarrollo de la aplicación. ....	34
Ilustración 3. Imagen de 'profile.png'.....	35
Ilustración 4. Imagen de 'objective.png'.....	36
Ilustración 5. Implementación de la tabla 'users'.....	37
Ilustración 6. Implementación de la tabla 'objectives'.....	38
Ilustración 7. Implementación de la tabla 'tasks'.....	39
Ilustración 8. Implementación de la tabla 'task_schedules'.....	39
Ilustración 9. Implementación de la tabla 'activities'.....	40
Ilustración 10. Implementación de la tabla 'routines'.....	40
Ilustración 11. Implementación de la tabla 'routine_schedules'.....	41
Ilustración 12. Implementación de la tabla 'notifications'.....	41
Ilustración 13. Implementación de la tabla 'allocated_tasks'.....	42
Ilustración 14. Header.....	43
Ilustración 15. Implementación de 'header.html'.....	43
Ilustración 16. Inclusión del header en 'profile.html'.....	43
Ilustración 17. Footer.....	44
Ilustración 18. Implementación de 'footer.html'.....	45
Ilustración 19. Inclusión del footer en 'profile.html'.....	45
Ilustración 20. Página de Portal de Usuario .....	46
Ilustración 21. Implementación de 'index.html'.....	47
Ilustración 22. Implementación de 'index()'.....	47
Ilustración 23. Página de registro.....	48
Ilustración 24. Aviso de campo obligatorio.....	49
Ilustración 25. Validación del formulario de registro en 'register.js'.....	50
Ilustración 26. Alerta por error en el registro.....	50
Ilustración 27. Implementación de validatePhone(phone).....	51
Ilustración 28. Implementación de validatePassword(password, username) ..	51
Ilustración 29. Alerta de error en la validación de la contraseña.....	52
Ilustración 30. Aviso de la aceptación obligatoria de la Política de Privacidad.	52
Ilustración 31. Recogida de datos del formulario de registro en 'register()' ..	52
Ilustración 32. Comprobación de que el correo electrónico no está registrado en 'register()'.....	53
Ilustración 33. Mensaje de error si el nombre de usuario ya está registrado....	53
Ilustración 34. Creación de un nuevo usuario en la base de datos en 'register()'.....	53
Ilustración 35. Página de inicio de sesión. ....	54
Ilustración 36. Implementación de 'togglePassword(inputId, eyeld)'.....	55
Ilustración 37. Implementación de 'login()'.....	56
Ilustración 38. Página de objetivos. ....	57
Ilustración 39. Implementación de 'show_objectives()'.....	58
Ilustración 40. Implementación de 'my_objectives.html'.....	59
Ilustración 41. Página de nuevo objetivo. ....	60
Ilustración 42. Implementación de 'objective_form()'.....	61
Ilustración 43. Implementación de 'previewImage()'.....	62
Ilustración 44. Página de objetivo. ....	63
Ilustración 45. Implementación de la lista de tareas en 'objective.html'.....	64
Ilustración 46. Implementación de 'view_objective(objective_id)'.....	64

Ilustración 47. Implementación del cuadro modal para editar un objetivo en 'objective.html' .....	65
Ilustración 48. Cuadro modal para editar un objetivo.....	65
Ilustración 49. Implementación del cuadro modal para eliminar un objetivo en 'objective.html' .....	66
Ilustración 50. Cuadro modal para eliminar un objetivo.....	66
Ilustración 51. Página de nueva tarea. ....	67
Ilustración 52. Implementación de 'validateHours()'.....	68
Ilustración 53. Implementación de 'addSchedule()'.....	68
Ilustración 54. Formulario de tarea tras añadir días.....	68
Ilustración 55. Implementación de 'removeSchedule()'.....	69
Ilustración 56. Alerta de que debe haber al menos un día de la semana asociado a una tarea.....	69
Ilustración 57. Página de tarea. ....	70
Ilustración 58. Implementación de 'show_task(task_id)'.....	70
Ilustración 59. Cuadro modal para editar una tarea.....	71
Ilustración 60. Cuadro modal para eliminar una tarea. ....	71
Ilustración 61. Página de agenda. ....	73
Ilustración 62. Implementación de 'agenda(offset=0)'.....	74
Ilustración 63. Implementación de botones de navegación entre semanas. ....	74
Ilustración 64. Extracción de tareas asignadas.....	74
Ilustración 65. Asignación de tareas a 'week_data'. ....	75
Ilustración 66. Implementación de 'update_agenda_color()'.....	75
Ilustración 67. Página de progreso. ....	76
Ilustración 68. Implementación de 'progress()'.' .....	77
Ilustración 69. Implementación del resumen de objetivos.. ....	78
Ilustración 70. Estructuración del progreso de los objetivos.....	78
Ilustración 71. Página de notificaciones. ....	79
Ilustración 72. Implementación de 'inbox()'.....	80
Ilustración 73. Implementación de 'create_notification(user_id, title, description)'.....	80
Ilustración 74. Creación de una notificación al crear un objetivo.....	80
Ilustración 75. Implementación de 'delete_notification(notification_id)'.....	81
Ilustración 76. Cuadro modal para eliminar una notificación. ....	81
Ilustración 77. Implementación de 'delete_all_notifications()'.....	82
Ilustración 78. Cuadro modal para vaciar la bandeja de entrada. ....	82
Ilustración 79. Página de perfil.....	83
Ilustración 80. Implementación del mensaje de error o éxito al editar el perfil. 84	84
Ilustración 81. Mensaje de éxito al editar el perfil. ....	84
Ilustración 82. Cuadro modal para editar el perfil.....	85
Ilustración 83. Implementación de 'logout()'.....	85
Ilustración 84. Cuadro modal para cerrar sesión. ....	85
Ilustración 85. Implementación de 'delete_account()'.....	86
Ilustración 86. Cuadro modal para eliminar una cuenta.....	86
Ilustración 87. Página de ayuda.....	87
Ilustración 88. Implementación de help().....	88
Ilustración 89. Implementación de 'help.html' .....	88
Ilustración 90. Cuadro modal con información de ayuda.....	89
Ilustración 91. Página de Aviso Legal.....	90
Ilustración 92. Implementación de 'legal()' .....	90

Ilustración 93. Implementación de 'legal.html'.....	91
Ilustración 94. Página de Política de Privacidad. ....	92
Ilustración 95. Implementación de 'privacy_policy()'.....	92
Ilustración 96. Implementación de 'privacy_policy.html' .....	93
Ilustración 97. Página de error.....	93
Ilustración 98. Implementación de 'handle_exception(e)'.....	94
Ilustración 99. Implementación de 'error()'.....	94
Ilustración 100. Implementación de 'error.html'.....	94
Ilustración 101. Implementación de 'calculate_next_sundat(date)'.....	95
Ilustración 102. Implementación de 'extract_activities_routines_tasks()'.....	96
Ilustración 103. Implementación de 'order_busy_time(activities, routines, tasks, start_date, end_date)'.....	96
Ilustración 104. Implementación de 'calculate_free_time(busy_time, start_date, end_date)'.....	97
Ilustración 105. Implementación de 'allocate_tasks(objectives, free_time)'.....	98
Ilustración 106. Creación del nombre del archivo de foto de perfil.....	99
Ilustración 107. Comprobación de que hay una sesión iniciada.....	100
Ilustración 108. Textos alternativos en las imágenes de objetivos.....	100
Ilustración 109. Etiquetas ARIA en los botones para añadir una actividad o rutina .....	100
Ilustración 110. Lectura de pantalla con VoiceOver.....	101

## **Índice de tablas**

Tabla 1. Tecnologías utilizadas .....	24
Tabla 2. Herramientas utilizadas. ....	25
Tabla 3. Historias de usuario. ....	27
Tabla 4. Colores utilizados.....	32

# 1. Introducción

## 1.1. Estado actual

Vivimos en una era de constante digitalización y automatización. Desde hace varias décadas, la humanidad ha trabajado arduamente para construir máquinas y sistemas que faciliten nuestras labores diarias. Con el reciente auge de las inteligencias artificiales en nuestra vida cotidiana, una vida más cómoda y menos estresante parece cada vez más alcanzable.

A lo largo de su vida, una persona se fija diversos objetivos tanto a nivel personal como profesional. Sin embargo, muchos de estos objetivos no se alcanzan por diversos motivos, como la falta de realismo, cambios en el contexto personal, o una mala gestión del tiempo. No cumplir con estos objetivos puede generar una gran frustración, afectando a múltiples aspectos de la vida de una persona, incluyendo su salud física y mental.

Según el Dr. John Norcross, Profesor Distinguido en la Universidad de Scranton y Profesor Clínico en Psiquiatría en la Universidad Estatal de Nueva York Upstate Medical University, aproximadamente el 60% de los estadounidenses se fija objetivos personales en Año Nuevo cuando se les pregunta a inicios de diciembre, pero solo alrededor del 40% la hace para finales de mes o el 1 de enero según una encuesta de Harris Interactive. Una importante proporción de quienes hacen resoluciones de Año Nuevo logra cumplir sus objetivos. Entre el 40% y el 44% tiene éxito al cabo de seis meses (Tan, 2020).

Esto demuestra que una gran cantidad de personas, si extrapolamos estos datos de Estados Unidos de América, se plantean objetivos anualmente, lo que equivale a millones de personas en todo el mundo. Sin embargo, menos de la mitad logra cumplirlos, destacando la importancia de la planificación y la persistencia para alcanzar objetivos personales.

## 1.2. Objetivos iniciales

El objetivo principal de este proyecto consistirá en la elaboración del prototipo funcional de una aplicación web que funcione como agenda inteligente para asignarle a un usuario las tareas que conforman sus objetivos en un calendario.

A raíz de este objetivo principal, se proponen tres objetivos complementarios:

- 1. Desarrollar un algoritmo eficiente que planifique la agenda de un usuario en base a sus objetivos.**
  - El algoritmo debe ser rápido a la hora de asignar las tareas.
  - El algoritmo debe ser capaz de adaptarse de manera dinámica a los cambios que surjan para poder volver a planificar.

## **2. Desarrollar una aplicación web accesible.**

- Se diseñará una interfaz de usuario siguiendo las directrices de accesibilidad WCAG (Web Content Accessibility Guidelines) para garantizar que la aplicación pueda ser utilizada por usuarios con discapacidades.
- Se implementará un diseño responsive que ofrezca una experiencia gratificante en dispositivos de diferentes tamaños, asegurando una navegación fluida en ordenadores, tabletas y móviles.

## **3. Desarrollar una aplicación web segura.**

- Se aplicarán técnicas recomendadas por OWASP (Open Worldwide Application Security Project) para mitigar las consecuencias de potenciales ataques y garantizar la ciberseguridad.

En resumen, el objetivo del trabajo consiste desarrollar una aplicación web eficaz, accesible y segura. La eficiencia del algoritmo de planificación es crucial, garantizando que las tareas se asignen de manera rápida y adaptativa. Además, la aplicación debe ser intuitiva y fácil de usar para todos los usuarios, proporcionando una experiencia de usuario agradable y satisfactoria. La seguridad es igualmente fundamental, asegurando que ni el sistema ni los usuarios estén en riesgo de ataques o pérdidas de información.

## **1.3. Competencias específicas cubiertas**

Se han desarrollado las siguientes competencias específicas relacionadas con el Grado en Ingeniería Informática (Escuela de Ingeniería Informática. Universidad de Las Palmas de Gran Canaria., 2019):

### **1.3.1. CI1**

**“Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.”**

Este proyecto se enfoca en el desarrollo y la evaluación de una aplicación web que, además de destacar por su facilidad de uso, garantiza la seguridad y accesibilidad mediante la implementación de las mejores prácticas. La calidad de la aplicación radica en la sinergia de estos elementos, ofreciendo una experiencia de usuario óptima.

### 1.3.2. CI12

**“Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.”**

Los datos capturados por la aplicación se almacenan en una base de datos relacional con una serie de tablas que contienen información sobre usuarios, sus objetivos y sus eventos diarios. La correcta estructuración y utilización de estos datos son fundamentales para proporcionar una experiencia personalizada a cada usuario.

## 1.4. Aportaciones

Esta aplicación web se presenta como una solución ideal para impulsar y motivar a las personas a alcanzar sus objetivos, y así, mejorar su bienestar general. La aplicación utilizará un algoritmo óptimo para planificar la agenda del usuario, basándose en los parámetros específicos de cada uno de sus objetivos. De esta manera, permitirá a los usuarios tener sus metas permanentemente integradas en su agenda, monitorizar su progreso y gestionar su tiempo de manera eficiente para lograr la satisfacción de cumplir esos objetivos.

## 1.5. Conceptos y terminologías del proyecto

Esta aplicación web pretende que un usuario pueda marcar sus objetivos personales e indique las tareas que debe realizar para cumplir cada objetivo.

**Objetivo:** Meta que un usuario desea cumplir en un periodo concreto de tiempo. Por ejemplo, aprender inglés entre el 1 de enero y el 31 de diciembre de 2024.

**Tarea:** Acción a realizar para cumplir el objetivo. Por ejemplo, hacer examen de prueba escrito.

Esas tareas se asignarán a la agenda del usuario a través de un algoritmo. Este algoritmo empleará la priorización y las restricciones de cada objetivo y tarea. También, se basará en la disponibilidad de horarios para plasmar las tareas en la agenda. En su agenda, un usuario podrá visualizar los eventos que tiene para cada día. Los diferentes tipos de evento son:

**Actividad:** Evento puntual que un usuario añade a la agenda para un día concreto. Por ejemplo, ir a un concierto el 22 de diciembre de 20:00 a 22:00.

**Rutina:** Evento periódico que un usuario añade y se repetirá ciertos días de la semana. Por ejemplo, ir a trabajar de lunes a viernes de 08:00 a 15:00.

**Tarea asignada:** Acción asociada a un objetivo que se asigna a la agenda a través del algoritmo. Por ejemplo, hacer examen de prueba escrito el 17 de enero de 11:00-13:00.

## 2. Marco teórico

### 2.1. Estudio del mercado

El blog de la empresa Zapier, especializada en integraciones para aplicaciones web para su uso en flujos de trabajo automatizados, realizó en julio de 2024 una clasificación de los ocho mejores asistentes de planificación con inteligencia artificial (Rebelo, 2024). Como se resalta en el artículo, existen diferencias entre los calendarios inteligentes y los calendarios de inteligencia artificial ya que los calendarios inteligentes adoptan algoritmos deterministas que ofrecerán resultados iguales cuando los datos de entrada sean iguales. Mientras, los calendarios de inteligencia artificial son capaces de aprender de la experiencia y ofrecer resultados más personalizados. Pese a que en este trabajo se esté desarrollando una agenda inteligente, se analizarán las características y funcionalidades de tres de estas ocho mejores agendas de inteligencia artificial: Reclaim AI, Motion y Trevor AI. Cada una con una finalidad diferente. Además, se analizarán dos aplicaciones con la meta de ayudar a un usuario a cumplir sus objetivos más específicamente: GoalsOnTrack y Habitica.

#### 2.1.1. Reclaim AI

##### **Descripción:**

Reclaim AI (<https://reclaim.ai/>) hace uso de la inteligencia artificial para optimizar la agenda de un usuario al priorizar sus tareas, hábitos y reuniones. Esta herramienta se puede aplicar en individuos o en equipos pequeños y grandes.

##### **Características:**

- Bloquea tiempo del calendario para hábitos o rutinas recurrentes.
- Permite a un usuario compartir sus horarios de disponibilidad para agendar reuniones.
- Integra tareas automáticamente basándose en la disponibilidad del calendario del usuario.
- Ofrece analíticas sobre el uso del tiempo tanto a nivel personal como de los miembros de un equipo. También, desarrolla informes semanales para evaluar la productividad.
- Se integra con diversas herramientas de trabajo: Slack, Asana, Clickup, Google Tasks, Jira Software, Todoist, Zoom, Linear, Raycast y Hubspot.
- Ofrece soporte por chat en vivo y correo electrónico.

##### **Disponibilidad:**

- Web.

**Precios:**

- 'Lite': gratis.
- 'Starter': \$8 por usuario al mes.
- 'Business': \$12 por usuario al mes.
- 'Enterprise': \$18 por usuario al mes.

## 2.1.2. Motion

**Descripción:**

Motion (<https://www.usemotion.com/>) trata de evitar la planificación manual de la agenda de un usuario. Para ello, emplea sus técnicas de inteligencia artificial que planifica para cada día las tareas de un usuario además de organizar planes óptimos estructurando las tareas de un equipo en la gestión de proyectos. Esta aplicación está orientada a personas o equipos con flujos de trabajo complejos que necesiten una optimización en sus agendas para ajustar los tiempos y momentos que dedicar a las tareas y reuniones.

**Características:**

- Optimiza dinámicamente los proyectos y tareas en la agenda basados en la prioridad varias veces al día de manera automática.
- Planifica las tareas de un proyecto para elaborar un plan óptimo para la gestión de proyectos.
- Aplica un algoritmo único autodenominado 'The Happiness Algorithm' con el fin de incrementar la productividad.
- Notifica al usuario con alertas avisando de la fecha límite de las tareas.
- Permite combinar los calendarios de Google Calendar, Outlook Calendar y Apple Calendar.
- Alerta al usuario en caso de existir una sobrecarga de trabajo o si se han establecido fechas límites que no se van a poder cumplir.
- Ofrece los horarios de disponibilidad de un usuario para poder agendar reuniones.

**Disponibilidad:**

- Web.
- Escritorio (Windows y MacOS).
- Móvil (Android e iOS).

**Precios:**

- 'Individual': \$19 al mes.
- 'Business Standard': \$12 por usuario al mes.
- 'Business Pro': No se especifica.

### 2.1.3. Trevor AI

**Descripción:**

Trevor AI (<https://www.trevorai.com/>) busca que sus usuarios tengan control, claridad y enfoque en las tareas diarias. Para ello, ofrece sugerencias personalizadas para la asignación de tareas en un día y facilita el enfoque en la tarea actual al activar un modo de enfoque. El público objetivo de este producto son aquellas personas que necesiten una plataforma sencilla que les permita mejorar la productividad y centrarse en las tareas.

**Características:**

- El sistema de inteligencia artificial puede asignar la duración de cada tarea basado en la planificación personal.
- Se integra con las siguientes herramientas de trabajo: Todoist, Google Calendar, Outlook Calendar, Office 365 Calendar, Microsoft To Do. Además, tienen en desarrollo la integración con Google Tasks.
- Las tareas se pueden añadir y gestionar en listas intuitivas además de asignarlas a una planificación diaria. El sistema realizará sugerencias para la planificación un día, tres días o una semana.
- Incluye un modo de enfoque que utiliza un temporizador y un espacio para notas con el objetivo de mantener la concentración en la tarea en curso.
- Ofrece un análisis de horarios recientes y envía sugerencias personalizadas por correo electrónico para la optimización del tiempo.

**Disponibilidad:**

- Web.

**Precios:**

- 'Free Plan': gratis.
- 'Pro Plan': \$6 al mes o \$60 al año.

### 2.1.4. GoalsOnTrack

**Descripción:**

GoalsOnTrack (<https://www.goalsontrack.com/>) trata de ayudar a sus usuarios a cumplir sus objetivos. Para ello, tratan de promover objetivos SMART (específicos, medibles, alcanzables, realistas y de duración limitada) que apoyándose en su sistema de inteligencia artificial para recomendaciones, un usuario pueda tener sus objetivos presentes y monitorizados.

**Características:**

- Permite crear objetivos a raíz de plantillas ya determinadas.
- Posee una herramienta de inteligencia artificial que recomienda planes de acción para cada objetivo con la meta de fomentar el progreso.
- Permite desglosar cada objetivo en diversos subobjetivos y a la vez desglosar cada subobjetivo en más subobjetivos.

- Se integra con los calendarios de Google Calendar, iCal y Outlook.
- Ofrece escribir un diario para las reflexiones del usuario con sus objetivos.
- Ofrece informes y gráficos mostrando los objetivos cumplidos, tareas y hábitos completados y tiempo de uso.
- Tiene una versión para equipos en la que los miembros del equipo podrán compartir sus objetivos y conversaciones.

**Disponibilidad:**

- Web.

**Precios:**

- \$68 al año.

## 2.1.5. Habitica

**Descripción:**

Habitica (<https://habitica.com/static/home>) busca la construcción de hábitos positivos y productividad mediante un sistema de recompensas y castigos que trata de impulsar a sus usuarios a lograr sus metas personales. Según sus proponentes, tratan de convertir la vida en un juego.

**Características:**

- Un usuario podrá registrar sus hábitos, tareas diarias y tareas pendientes para llevar un seguimiento de ellas.
- Diariamente, un usuario podrá tachar una tarea completada para subir de nivel y desbloquear características del juego como armaduras de batalla, mascotas, habilidades o misiones.

**Disponibilidad:**

- Web.
- Móvil (Android e iOS).

**Precios:**

- Plan individual: gratis.
- Plan grupal: \$9 al mes + \$3 por miembro adicional.

## 2.1.6. Conclusiones

En conclusión, se puede apreciar que ya existen diferentes aplicaciones para la asignación de manera inteligente o automática de tareas en una agenda además de aplicaciones que ayuden a un usuario a cumplir sus objetivos y metas. Sin embargo, existen diferencias en los enfoques y funcionalidades de estas herramientas con este trabajo.

Herramientas como Reclaim AI, Motion y Trevor AI, que destacan por sus sistemas de inteligencia artificial y algoritmos para ubicar las tareas de un usuario en su agenda automáticamente, están principalmente orientadas a la gestión de proyectos y a la maximización de la productividad laboral que al cumplimiento de objetivos personales.

Por su parte, GoalsOnTrack y Habitica son herramientas que sí están claramente enfocadas en incentivar a un usuario a lograr sus objetivos. Pese a ello, GoalsOnTrack tan solo usa su sistema de IA para elaborar un plan de acción y en ningún momento asignar las tareas de una manera eficiente directamente a la agenda del usuario. Habitica por su parte, tiene un sistema que requiere la asignación manual de tareas sin una planificación temporal detallada.

Además, la mayoría de estas aplicaciones son de pago y algunas de ellas con precios considerablemente elevados la cual podría suponer una barrera o impedimento a usuarios que busquen o necesiten soluciones más accesibles.

Es por ello por lo que pienso que este trabajo puede cubrir una brecha significativa en el mercado ya que no he sido capaz de localizar ninguna aplicación que combine un sistema de planificación inteligente y un enfoque claro en el cumplimiento de objetivos personales.

## 2.2. Accesibilidad en la web

### 2.2.1. WCAG 2.2

W3C (World Wide Web Consortium) es una organización fundada en 1994 cuya misión es impulsar una arquitectura consistente y unos estándares robustos en lo que al desarrollo web refiere. Para ello implementa las WCAG. Estas pautas publicadas proporcionan recomendaciones sobre cómo mejorar la accesibilidad de los sitios web para usuarios con discapacidades. El 12 de diciembre de 2024 se publicó la guía WCAG 2.2 (Campbell, y otros, 2024) con las extensiones de las recomendaciones de la versión 2.1. Los cuatro principios fundamentales según la WCAG para desarrollar una aplicación web accesible son:

#### 1. Perceptible:

El contenido de la página web debe poder ser percibido por cualquier usuario independientemente de sus capacidades sensoriales. Para ello, se adoptan los siguientes métodos:

- Textos alternativos:

Las imágenes, vídeos, gráficos, etc. deben estar descritas mediante un texto alternativo que permita a los usuarios con deficiencias visuales puedan acceder a esa información mediante lectores de pantalla.

- Medios basados en el tiempo:

Los audios o vídeos deben ir acompañados de subtítulos, descripciones auditivas o transcripciones que permitan a las personas con discapacidad auditiva o visual comprender la información.

- Adaptable:

El contenido de la web se debe ajustar a diferentes tamaños de dispositivos o sistemas para que la presentación sea entendible para el usuario.

- Distingible:

El contenido de una página web debe ser fácil de entender y diferenciar para un usuario. Por ello, debe haber un contraste óptimo entre el fondo y el texto para que éste sea legible además de que se entienda claramente cuando hay un botón o un enlace.

## **2. Operable:**

Los usuarios deben poder interactuar con el contenido sin impedimentos.

- Accesible por teclado:

El contenido debe poder ser operable utilizando tan solo el teclado. No debe ser necesario tener un dispositivo apuntador ya que el contenido debe ser accesible por personas con discapacidades motoras.

- Tiempo suficiente:

Los usuarios deben tener tiempo suficiente para leer o interactuar con todo el contenido. No debe haber imágenes, textos, etc., que desaparezcan de la pantalla demasiado rápido.

- Convulsiones y reacciones físicas:

Se debe evitar que la página web contenga elementos que parpadeen rápidamente o tengan efectos visuales que pudiesen provocar en el usuario malestar.

- Navegable:

La estructura de la página web debe ser clara para que los usuarios encuentren lo que buscan con facilidad de forma intuitiva.

- Modalidades de entrada:

Los usuarios deben poder interactuar con el contenido de la web con tecnologías de asistencia como controladores de voz o dispositivos de seguimiento ocular y no solo con el teclado o el ratón.

## **3. Comprensible:**

La información proyectada en la web debe ser fácil y comprensible para que el usuario la pueda entender y usar sin dificultades.

- Legible:

Se debe aplicar un lenguaje sencillo y comprensible para que el usuario pueda entender la información de manera clara.

- Predecible:

Los usuarios deben poder prever lo que ocurrirá una vez interactúen con un elemento interactivo como un enlace o un botón.

- Asistencia de entrada:

Cuando un usuario deba llenar un formulario o campo de entrada, se deben ofrecer instrucciones comprensibles y mensajes de error claros para que se puedan llenar con éxito.

#### **4. Robusto:**

El contenido de la página web debe poder ser accesible y funcional con una amplia gama de tecnologías y dispositivos. Esto se debe mantener en el tiempo con la evolución de las tecnologías actuales e irrupción de nuevas tecnologías y herramientas.

- Compatible:

Se debe seguir los estándares web para que el sitio web sea correctamente interpretado por los navegadores y tecnologías de asistencia.

### **2.2.2. Conclusiones**

Seguir las pautas marcadas por WCAG 2.2 para conseguir desarrollar una aplicación web accesible es vital para que cualquier usuario, sin importar sus condiciones, pueda interactuar y disfrutar de todas las funcionalidades de la aplicación sin complicaciones. Como desarrollador de software, considero una responsabilidad el desarrollar un sistema que pueda ser utilizado por cualquier persona para así romper las barreras que puedan impedir a personas que ya tienen que lidiar con múltiples obstáculos en su día a día.

## **2.3. Seguridad en la web**

### **2.3.1. OWASP**

OWASP (Open Worldwide Application Security Project) es una fundación sin ánimo de lucro que se inauguró en diciembre de 2001 en Estados Unidos de América. Como misión (OWASP, s.f.a) tienen promover buenas prácticas que desarrolladores de software y profesionales de la seguridad puedan aplicar con la intención de hacer el mundo del software un mundo más seguro. Una de las múltiples prácticas que adopta esta organización para promover la seguridad digital es publicar el 'OWASP Top Ten'. Ésta es una lista que enumera los diez riesgos más críticos referentes a la seguridad en aplicaciones web. OWASP incentiva a las empresas de alrededor de todo el mundo a emplear este documento para cambiar la cultura de desarrollo de software en la empresa a una que produzca código más seguro. Con el objetivo de evitar o mitigar estas diez potenciales vulnerabilidades, OWASP facilita unas 'cheat sheets' (OWASP, s.f.b) como guías con las prácticas y recomendaciones que aplicar para cada uno de los riesgos. Esta es la lista de vulnerabilidades desde 2021 ordenadas de mayor a menor riesgo:

1. Control de acceso roto.
2. Fallas criptográficas.

3. Inyección.
4. Diseño inseguro.
5. Mala configuración de seguridad.
6. Componentes vulnerables y obsoletos.
7. Fallas de identificación y autenticación.
8. Fallas en la integridad de software y datos.
9. Fallas en el registro y monitoreo de seguridad.
10. Solicitudes de lado del servidor.

### 2.3.2. Conclusiones

Existen muchísimas potenciales amenazas a una aplicación web. Por ello, considero fundamental adoptar todas las medidas de seguridad posibles para evitar o mitigar los efectos de estos ataques. Gracias a proyectos como OWASP, los desarrolladores podemos estar informados de las nuevas formas de ataque, las más populares y cómo defendernos ante ellas. Considero que es un hito imposible tener la certeza de que una aplicación web es completamente segura en todos los aspectos. Sin embargo, es una responsabilidad de los desarrolladores tratar de llegar lo más lejos posible al adoptar todas las medidas de seguridad aplicables.

## 2.4. Normativa y Legislación

Para cumplir con la legislación vigente, se han desarrollado las páginas de Aviso Legal y Política de Privacidad.

### Aviso Legal:

Para cumplir con la Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico (Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico., 2002), se ha implementado una página de Aviso Legal en la cual tanto usuarios registrados como no registrados podrán visitar. En ella, basado en el ‘Artículo 10. Información general.’ Se detalla toda la información correspondiente acerca del propietario y responsable de la aplicación web (incluyendo nombre, N.I.F. y dirección de correo electrónico con el que se podrá establecer una comunicación directa entre el responsable y el cliente). Además, se informa de la motivación detrás de la implementación del sitio web y su finalidad. Por último, el titular especifica las responsabilidades que elude para que el usuario sea plenamente consciente de los riesgos a los que se enfrenta al acceder a este sitio web. En este caso, se especifican tres responsabilidades de las que el titular se excluye:

- Si el sitio web no está disponible durante un periodo de tiempo.
- Si el sitio web suspende su actividad en cualquier momento.
- Si el usuario sufre ciberataques por el contenido del sitio web.

### Política de Privacidad:

Para cumplir la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (Ley Orgánica 3/2018, de 5 de

diciembre, de Protección de Datos Personales y garantía de los derechos digitales., 2018), primeramente, se detallan los datos personales y de contacto del titular del sitio web. A partir de ahí, se informa al usuario de los datos personales recopilados por la aplicación y el porqué de cada uno. En este caso, los datos personales recopilados son:

- Correo electrónico.
- Número de teléfono.
- Imágenes.

Ninguno de estos datos es mandatorio para poder usar la aplicación.

Además, se informa al usuario que los datos personales proporcionados serán almacenados permanentemente en la base de datos de la aplicación, pero completamente eliminados, sin dejar rastro, al eliminar la cuenta.

Basado en el ‘CAPÍTULO II Ejercicio de los derechos’ de esta Ley, se informa al usuario de todos los derechos a los que se puede acoger: acceso, rectificación, supresión, limitación del tratamiento, portabilidad y oposición. Se especifica un correo electrónico de contacto para que el usuario informe al responsable de su deseo de ejercer cualquiera de los derechos.

Para que un usuario pueda registrarse, deberá aceptar de manera obligatoria la Política de Privacidad, que se informa que podrá ser modificada en cualquier momento. En caso de modificación, se informará al usuario.

### 3. Análisis

#### 3.1. Flujo de funcionamiento

El flujo de funcionamiento de la aplicación comienza con el registro o inicio de sesión del usuario en la plataforma. Una vez autenticado, el usuario podrá comenzar a definir sus objetivos personales según sus necesidades. Para cada objetivo, se pueden añadir tareas específicas, que representan las acciones concretas necesarias para alcanzarlo. Posteriormente, el usuario podrá acceder a la sección de su agenda, donde tiene la posibilidad de añadir rutinas diarias y actividades puntuales programadas. Una vez configurada la agenda, el usuario puede hacer clic en el botón de planificación. Este botón activa un algoritmo que asigna automáticamente las tareas asociadas a los objetivos en los horarios disponibles de lo que resta de la semana actual y de la siguiente. A medida que el tiempo avanza, el usuario puede ajustar las prioridades de sus objetivos y tareas, lo que influirá directamente en cómo estas se distribuyen en su planificación futura, permitiendo una personalización continua según los constantes cambios en las prioridades del usuario.

## 3.2. Metodología

Para la elaboración de este proyecto, se ha decidido emplear la metodología Kanban (Vacanti & Coleman, 2020). Esta metodología, utilizada para implementar desarrollo de software ágil, tiene como objetivo optimizar el flujo de trabajo a la hora de organizar las tareas de un proyecto de una manera muy visual. Un sistema Kanban se basa en tres prácticas básicas que se complementan:

1. Definir y visualizar el flujo de trabajo.
2. Gestionar activamente los elementos en un flujo de trabajo.
3. Mejorar el flujo de trabajo.

Los elementos del trabajo se representan visualmente en un tablero Kanban, permitiendo ver el estado de cada tarea en cualquier momento. Las tareas serán representadas como tarjetas compuestas por una descripción. Con el objetivo de evitar la sobrecarga y mejorar el enfoque, se establecerá un máximo de 5 tareas que puedan estar simultáneamente en progreso. Las etapas del flujo de trabajo son:

1. Por hacer
2. En curso
3. Pruebas
4. Hecho

Los beneficios de adoptar esta estrategia son (Radigan, s.f.):

**1. Flexibilidad en la planificación:**

El desarrollo se centrará en las tareas en curso, permitiendo ajustes dinámicos dependiendo del contexto y las necesidades del proyecto.

**2. Reducir los cuellos de botella:**

Al limitar el número de tareas que puede haber en curso, se minimizan los bloqueos y mejora el flujo de trabajo.

**3. Optimizar el flujo de trabajo:**

Las tareas se seleccionan y se completan en función de la capacidad y disponibilidad actual, evitando bloqueos innecesarios y asegurando una entrega más rápida y eficiente.

Implementar la metodología Kanban en mi proyecto no solo ha mejorado mi eficiencia y productividad, sino que también me ha permitido generar un entorno de trabajo mejor estructurado y más claro. Al visualizar el progreso y limitar el trabajo en curso, he tenido mayor facilidad para identificar potenciales problemas y resolverlos, lo cual considero que como consecuencia ha producido un flujo de trabajo más efectivo.

### 3.3. Tecnologías

Tabla 1. Tecnologías utilizadas.

Tecnología	Versión	Descripción
<b>Python</b>	3.8.5	Lenguaje de programación de alto nivel.
<b>Flask</b>	3.0.2	Framework de Python para el desarrollo de aplicaciones web.
<b>SQLAlchemy</b>	2.0.28	ORM para modelar los datos para la base de datos.
<b>Flask-SQLAlchemy</b>	3.1.1	Extensión de Flask para interactuar con bases de datos utilizando SQLAlchemy.
<b>SQLite</b>	3.32.3	Motor de base de datos relacional.
<b>Werkzeug</b>	3.0.1	Gestiona contraseñas cifradas.
<b>python-dateutil</b>	2.9.0.post0	Utilizado para el manejo avanzado de fechas.
<b>Unidecode</b>	1.3.8	Procesamiento de cadenas de texto.
<b>Jinja2</b>	3.1.3	Motor de plantillas utilizado por Flask para renderizar HTML dinámico.
<b>UUID</b>	1.30	Generador de Identificadores Únicos Universales
<b>python-dateutil</b>	2.9.0.post0	Biblioteca para ampliar el módulo datetime.

<b>HTML</b>	HTML5	Lenguaje para definir la estructura y contenido de una página web.
<b>CSS</b>	CSS3	Lenguaje de estilo para diseñar el aspecto visual de una página web.
<b>JavaScript</b>	ES6	Lenguaje de programación para agregar interactividad a una página web.
<b>Fontawesome</b>	5.10.1.post1	Permite aplicar iconos a la interfaz de usuario.

Fuente: Elaboración propia.

### 3.4. Herramientas

Tabla 2. Herramientas utilizadas.

Herramienta	Descripción
<b>Pycharm</b>	Entorno de desarrollo integrado para Python.
<b>GitHub</b>	Plataforma para el control de versiones.
<b>Trello</b>	Aplicación web para la gestión del proyecto.
<b>Figma</b>	Aplicación web para el diseño de mock-ups.
<b>DB Browser for SQLite</b>	Aplicación para la gestión de la base de datos SQLite.
<b>VoiceOver</b>	Lector de pantalla integrado de macOS.

Fuente: Elaboración propia.

## 3.5. Requisitos

### 3.5.1. Requisitos funcionales

#### 1. Planificación de tareas:

El sistema deberá planificar la agenda de un usuario, asignando las tareas de sus objetivos en los días que restan de la semana actual y la semana siguiente.

#### 2. Notificaciones:

El sistema deberá emitir notificaciones a la bandeja de entrada del usuario.

#### 3. Acceso a las funcionalidades:

El sistema solo deberá permitir acceder a las funcionalidades a aquellos usuarios que hayan iniciado sesión previamente.

#### 4. Página de error:

El sistema mostrará una página de error con información relevante.

#### 5. Inicio de sesión:

El sistema deberá permitir al usuario iniciar sesión mediante el nombre de usuario o el correo electrónico junto con la contraseña.

El sistema deberá cargar la página de la agenda al iniciar sesión.

#### 6. Registro:

El sistema no deberá registrar a un usuario con un nombre de usuario mayor a diez caracteres.

El sistema deberá obligar al usuario a registrarse con una contraseña que cumpla las siguientes condiciones:

1. La contraseña debe contener mínimo 8 caracteres.
2. La contraseña debe contener al menos una letra mayúscula.
3. La contraseña debe contener al menos una letra minúscula.
4. La contraseña debe contener al menos un número.
5. La contraseña debe contener al menos un símbolo:  
!@#\$%^&\*(),.:{}|<
6. La contraseña no puede contener el nombre de usuario.

El sistema requerirá confirmar la contraseña para verificar que coinciden.

El sistema deberá aplicar un algoritmo de hash a la contraseña del usuario antes de almacenarla en la base de datos.

El sistema deberá cargar la página de objetivos al registrar un nuevo usuario.

### 3.5.2. Requisitos no funcionales

#### 1. Compatibilidad multiplataforma:

El sistema deberá ser compatible con múltiples sistemas operativos y dispositivos.

#### 2. Diseño responsive:

El sistema deberá aplicar un diseño responsive para garantizar una correcta visualización y funcionalidad en dispositivos con diferentes tamaños de pantalla como ordenadores, tabletas o móviles.

#### 3. Tiempo de respuesta:

El sistema deberá tener un tiempo de respuesta inferior a cinco segundos para cualquier operación.

#### 4. Accesibilidad:

El sistema deberá cumplir con las pautas de accesibilidad web WCAG.

#### 5. Seguridad:

El sistema deberá implementar medidas de seguridad basadas en las recomendaciones de OWASP.

## 3.6. Historias de usuario

Para priorizar la implementación de las diferentes Historias de Usuario, se ha adoptado el método MoSCoW. Con este método, se ha catalogado la prioridad de cada Historia de Usuario con una de las cuatro categorías: 'MUST', 'SHOULD', 'COULD' y 'WON'T (de mayor a menor prioridad). Primeramente, se han implementado todas las tareas con prioridad 'MUST'. Seguidamente, se desarrollaron las catalogadas como 'SHOULD'. Por último, se llevaron a cabo las definidas como 'COULD' con más impacto. Las menos relevantes no fueron implementadas finalmente. Las 'WON'T' no se consideró implementarlas en esta etapa del desarrollo del prototipo de la aplicación web, pero son tareas que se pretenden aplicar en el futuro para incrementar la calidad del sitio web.

Tabla 3. Historias de usuario.

Código	Descripción	Prioridad
HU-01	Un usuario podrá registrarse.	MUST
HU-02	Un usuario podrá iniciar sesión.	MUST

<b>HU-03</b>	Un usuario podrá crear un objetivo.	MUST
<b>HU-04</b>	Un usuario podrá editar los parámetros de un objetivo.	COULD
<b>HU-05</b>	Un usuario podrá marcar un objetivo como completado	COULD
<b>HU-06</b>	Un usuario podrá eliminar un objetivo.	SHOULD
<b>HU-07</b>	Un usuario podrá crear una tarea.	MUST
<b>HU-08</b>	Un usuario podrá editar los parámetros de una tarea.	COULD
<b>HU-09</b>	Un usuario podrá eliminar una tarea.	SHOULD
<b>HU-10</b>	Un usuario podrá añadir una actividad.	SHOULD
<b>HU-11</b>	Un usuario podrá editar una actividad.	WON'T
<b>HU-12</b>	Un usuario podrá eliminar una actividad.	SHOULD
<b>HU-13</b>	Un usuario podrá añadir una rutina.	SHOULD
<b>HU-14</b>	Un usuario podrá editar una rutina.	WON'T
<b>HU-15</b>	Un usuario podrá eliminar una rutina.	SHOULD
<b>HU-16</b>	Un usuario podrá visualizar la planificación de su agenda en un calendario.	COULD
<b>HU-17</b>	Un usuario podrá visualizar la planificación de su agenda en una vista semanal.	MUST
<b>HU-18</b>	Un usuario podrá replanificar manualmente su agenda.	MUST

<b>HU-19</b>	Un usuario podrá marcar como completada una tarea asignada en su agenda.	<b>SHOULD</b>
<b>HU-20</b>	Un usuario podrá cambiar el color de su agenda.	<b>COULD</b>
<b>HU-21</b>	Un usuario podrá marcar como completada una tarea asignada en su agenda.	<b>SHOULD</b>
<b>HU-22</b>	Un usuario podrá exportar su planificación semanal por email, Whatsapp o PDF.	<b>COULD</b>
<b>HU-23</b>	Un usuario podrá visualizar el progreso de sus objetivos.	<b>SHOULD</b>
<b>HU-24</b>	Un usuario podrá marcar su nivel de motivación con un objetivo.	<b>WON'T</b>
<b>HU-25</b>	Un usuario podrá eliminar una notificación de su bandeja de entrada.	<b>COULD</b>
<b>HU-26</b>	Un usuario podrá vaciar su bandeja de entrada.	<b>COULD</b>
<b>HU-27</b>	Un usuario podrá editar los datos de su perfil.	<b>SHOULD</b>
<b>HU-28</b>	Un usuario podrá cerrar su sesión.	<b>MUST</b>
<b>HU-29</b>	Un usuario podrá eliminar su cuenta.	<b>MUST</b>
<b>HU-30</b>	Un usuario podrá acceder a la página de ayuda.	<b>SHOULD</b>
<b>HU-31</b>	Un usuario podrá acceder a un cuadro con información sobre la página en la que se encuentre.	<b>COULD</b>
<b>HU-32</b>	Un usuario podrá cambiar el idioma de la aplicación.	<b>WON'T</b>

HU-33	Un usuario podrá acceder a las cuentas de redes sociales de Facebook, TikTok, Instagram y X.	COULD
HU-34	Un usuario podrá acceder a la política de privacidad.	SHOULD
HU-35	Un usuario podrá acceder a la política de cookies.	WON'T
HU-36	Un usuario podrá acceder a la página de aviso legal	SHOULD

Fuente: Elaboración propia.

## 4. Diseño

### 4.1. Objetivo del diseño

El objetivo del diseño de esta aplicación web se basa principalmente en cumplir los criterios de accesibilidad pautados por WCAG 2.2. Se ha tratado de cumplir con los siguientes requisitos mediante el diseño:

#### 1. Adaptable:

Aplicando un diseño responsive que adapte el contenido de cada página de la mejor manera posible a las dimensiones del dispositivo utilizado por el usuario.

#### 2. Distinguible:

Se han aplicado colores que se complementan muy bien entre sí para ofrecer siempre un buen contraste que no dificulte la diferenciación entre el fondo y el texto. Para ello, también se permitirá al usuario personalizar su experiencia al poder seleccionar los colores de su agenda, objetivos, actividades y rutinas. Uno de los objetivos de esta configuración es tratar de garantizar que personas que pueden tener dificultades para diferenciar ciertos colores, puedan adaptar los colores de la interfaz a su gusto. Además, los botones están bien marcados y se les aplica un hover para que, al pasar el ratón por encima, el usuario note que es posible interactuar. Al pasar el ratón por encima, el puntero cambiará de una flecha a una mano para así hacer todavía más evidente que se puede interactuar con ese componente.

#### 3. Navegable:

La estructura de la página web es muy simple e intuitiva con la menor carga de texto y botones posible para el usuario en todo momento tenga

claras las posibilidades que tiene. Además, fomenta que la navegación por teclado sea rápida.

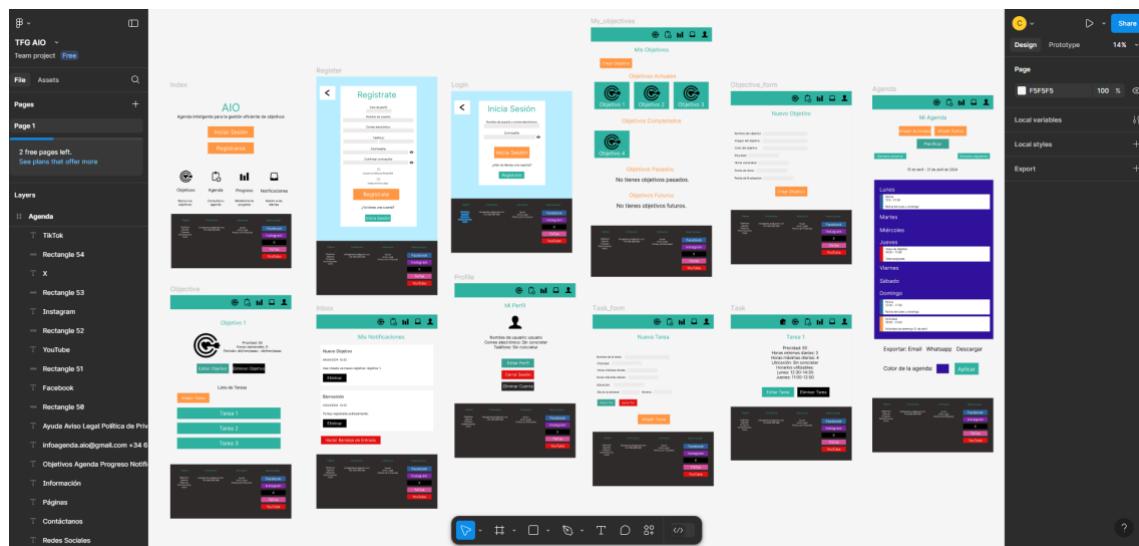
#### 4. Predecible:

Los botones contienen texto y/o iconos que indican claramente al usuario la acción que va a desencadenar hacer clic sobre ese botón.

## 4.2. Herramienta para el diseño

Para la realización de los mock-ups con el diseño de la interfaz de usuario se ha empleado Figma. Esta herramienta online colaborativa permite el modelaje de las vistas de las páginas web para ordenadores, tabletas y móviles. En este caso, se han desarrollado los mock-ups para las dimensiones de un ordenador.

Ilustración 1. Diseños de la interfaz de usuario en Figma.



Fuente: Elaboración propia.

## 4.3. Diseño estético

### 4.3.1. Colores

Tabla 4. Colores utilizados.

Color	Código hexadecimal	Descripción
	#2EB19F	Verde
	#38A092	Verde oscuro
	#B7EDFF	Azul celeste
	#003366	Azul marino
	#FF9642	Naranja
	#F5F5F5	Gris
	#262626	Gris oscuro
	#E21010	Rojo
	#FFFFFF	Blanco
	#000000	Negro

Fuente: Elaboración propia.

### 4.3.2. Tipografía

La tipografía seleccionada para todos los textos de la aplicación web ha sido Montserrat de la familia de fuentes sans-serif. Esta decisión está motivada por múltiples motivos. Principalmente, por su alta legibilidad. Es muy limpia y atractiva visualmente. Además, se amolda muy bien tanto en títulos como en

textos largos y es una tipografía web muy popular por lo que tiene un amplio soporte.

### 4.3.3. Botones

Todos los botones tienen alguna interacción al posicionar el puntero sobre ellos. Estas interacciones se traducen a cambios de color, aumentos de tamaño y/o el cambio del puntero de una flecha a una mano. Algunos cambios de color hacen el efecto de inversión de colores entre el fondo y el texto. Por ejemplo, el botón para crear objetivo o el de añadir actividad. Otros cambios de color tan solo modifican el color del texto. Por ejemplo, el botón de ayuda en el footer. Por su parte, se pueden apreciar aumentos de tamaño cuando se coloca el ratón sobre un objetivo en la página de objetivos o sobre una tarea en la lista de tareas de un objetivo.

### 4.3.4. Iconos

Se han aplicado iconos mediante la biblioteca de iconos web Font Awesome con el objetivo de hacer que la página esté menos cargada de texto para mejorar la estética y hacer que las funcionalidades sean más fáciles o rápidas de interpretar.

### 4.3.5. Formularios

Los formularios tendrán los títulos de los campos obligatorios en negrita. Además, aquellos formularios que permitan añadir alguna imagen mostrarán una vista previa de la imagen seleccionada.

### 4.3.6. Cuadros modales

Se han aplicado cuadros modales para las siguientes situaciones en la aplicación web:

1. Formularios de añadir actividad y rutina en la página de la agenda.
2. Cuadros de ayuda en múltiples páginas.
3. Edición de objetivos, tareas y perfiles.
4. Confirmación de eliminación, vaciado de bandeja de entrada y cerrado de sesión.

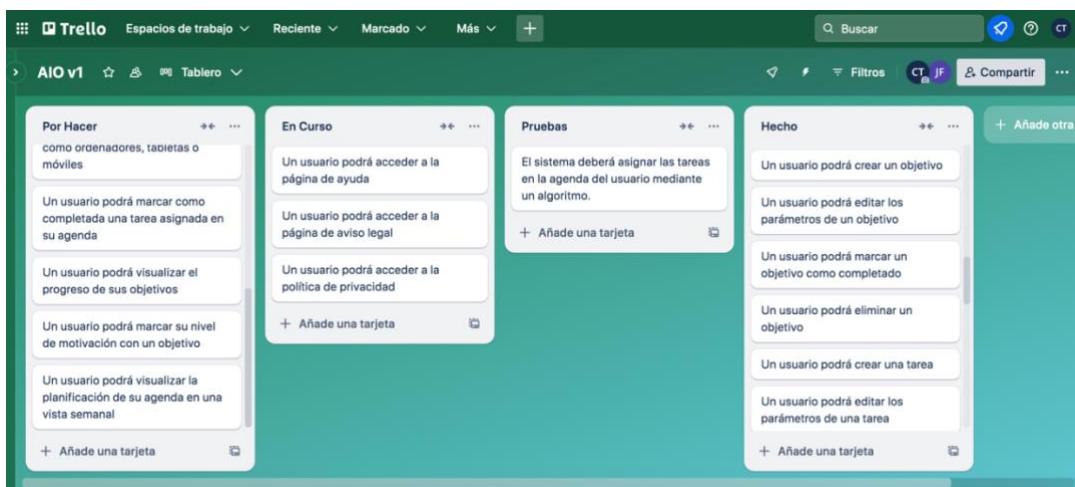
El motivo para aplicar estos cuadros modales ha sido tratar de mejorar la experiencia del usuario al proporcionar una experiencia más fluida en la que no se debe ni cargar una nueva página ni perder la página actual.

## 5. Desarrollo

### 5.1. Organización

Para aplicar la metodología Kanban, se ha utilizado la aplicación web Trello. Esta herramienta ha permitido crear un tablón en el que definir las etapas del flujo de trabajo y colocar las tarjetas (representando cada tarea) en la etapa correspondiente. Las tarjetas han representado las historias de usuario y los requisitos.

Ilustración 2. Tablero de Trello durante el desarrollo de la aplicación.



Fuente: Elaboración propia.

La metodología Kanban ha permitido una gran flexibilidad a la hora de desarrollar esta aplicación web. El desarrollo no se ha dividido en fases concretas con tiempos específicos para completar una serie de tareas. Esta decisión vino motivada por la incertidumbre en mi disponibilidad en los meses hábiles para completar el proyecto. Las tareas en el tablero Kanban se han conformado por las Historias de Usuario y los requisitos, tanto funcionales como no funcionales. Las tareas se han realizado en el orden marcado por la prioridad en caso de ser Historias de Usuario y en caso de ser requisitos, se han implementado basado en la relación con la Historia de Usuario en curso. Por ejemplo, la HU-01 ('Un usuario podrá registrarse') se ha implementado en paralelo con las tareas del requisito funcional 6 ('Registro'). Para aprovechar el beneficio de Kanban de reducir los cuellos de botella, siempre se ha respetado no tener más de cinco tareas en curso y así mantener el enfoque y evitar una sobrecarga de trabajo.

## 5.2. Estructura del Proyecto

- **instance**
  - AIO.db
- **static**
  - **images**
    - **objectives**
    - **profile**
    - objective.png
    - profile.png
  - **js**
  - **styles**
- **templates**
- algorithm.py
- app.py
- db.py

### **instance:**

El directorio ‘instance’ se utilizará para almacenar el archivo ‘AIO.db’. En él, se almacenarán todos los datos de la base de datos.

### **static:**

El directorio ‘static’ se empleará para almacenar los recursos estáticos. Es por ello por lo que en él se encuentran los directorios ‘images’ (conteniendo archivos de imagen), ‘js’ (con los scripts de JavaScript para cada página) y ‘styles’ (con los archivos CSS [Cascading Style Sheets] que diseñan la apariencia de las páginas).

Dentro de ‘images’, se encuentran los directorios ‘objectives’ y ‘profile’. En ellos, se almacenarán las imágenes cargadas a la aplicación por parte de los usuarios al crear objetivos o registrarse respectivamente.

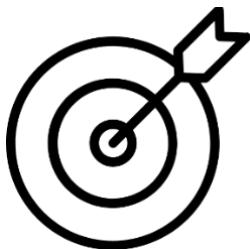
Por su parte, ‘objective.png’ y ‘profile.png’ serán las imágenes por defecto que se mostrarán en el caso de que un usuario no decida incorporar una imagen al crear un objetivo o registrarse.

*Ilustración 3. Imagen de ‘profile.png’.*



Fuente: (PNGWing, s.f.)

*Ilustración 4. Imagen de 'objective.png'.*



*Fuente: (Freepik, s.f.)*

#### **templates:**

En el directorio ‘templates’, se encuentran las plantillas HTML (HyperText Markup Language) con la estructura de todas las páginas del sitio web.

#### **Archivos .py:**

Por último, se encuentran los archivos algorithm.py, app.py y db.py. Principalmente, el archivo app.py es el núcleo de la aplicación. En él, se estructura y se organiza la lógica de la aplicación web con Flask. El archivo db.py define la estructura de la base de datos con SQLAlchemy como ORM (Object-Relational Mapping). Finalmente, en algorithm.py se encuentran las funciones con la lógica del algoritmo determinista implementando para la asignación de tareas a la agenda de un usuario.

## 5.3. Base de Datos

La base de datos utiliza el ORM SQLAlchemy (The Python SQL Toolkit and Object Relational Mapper, s.f.) que permite a los desarrolladores trabajar con bases de datos relacionales de manera más sencilla al tratarlas como objetos de Python. Permite un acceso eficiente, un alto rendimiento y una simplificación en las consultas SQL.

Esta base de datos se ha creado para gestionar un sistema en el que los usuarios puedan crear objetivos, tareas, actividades y rutinas además de un sistema que pueda generar notificaciones y la asignación de tareas a una agenda. A continuación, se describe el propósito de cada tabla y los detalles de su implementación.

#### **User (Usuarios):**

La tabla ‘users’ almacena la información personal de los usuarios. Incluye la información básica como el nombre de usuario o la contraseña además de campos no obligatorios como el correo electrónico, número de teléfono compuesto por el código del país y el número o una imagen de perfil. Además, incluye un campo de personalización como el color de la agenda del usuario.

**id:** Identificador único del usuario (clave primaria).  
**image:** Imagen del perfil del usuario (opcional).  
**username:** Nombre único del usuario (obligatorio y único).  
**email:** Correo electrónico del usuario (opcional y único).  
**phone\_country\_code:** Código telefónico del país (opcional).  
**phone\_number:** Número de teléfono (opcional).  
**password:** Contraseña del usuario (obligatorio).  
**agenda\_color:** Color de la agenda del usuario (opcional).

Relación 1:N con la tabla Objective:  
Un usuario puede tener múltiples objetivos.

Relación 1:N con la tabla Activity:  
Un usuario puede tener múltiples actividades.

Relación 1:N con la tabla Routine:  
Un usuario puede tener múltiples rutinas.

Relación 1:N con la tabla Notification:  
Un usuario puede tener múltiples notificaciones.

*Ilustración 5. Implementación de la tabla ‘users’.*

```
class User(db.Model):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    image = db.Column(db.String(255), nullable=True)
    username = db.Column(db.String(100), unique=True, nullable=False)
    email = db.Column(db.String(100), unique=True, nullable=True)
    phone_country_code = db.Column(db.String(10), nullable=True)
    phone_number = db.Column(db.String(20), nullable=True)
    password = db.Column(db.String(100), nullable=False)
    agenda_color = db.Column(db.String(7), nullable=True)
    objectives = db.relationship('Objective', backref='user', lazy=True, cascade="all, delete-orphan")
```

Fuente: Elaboración propia.

### **Objective (Objetivos):**

La tabla ‘objectives’ representa los objetivos personales estipulados por un usuario. Un objetivo tendrá los parámetros necesarios como la prioridad respecto a los otros objetivos, las horas de dedicación semanales o las fechas de inicio y fin además de un nombre o título, imagen o color asociado. Habrá un campo booleano que establecerá el estado de finalización del objetivo (si se ha completado o no).

**id:** Identificador único del objetivo (clave primaria).  
**name:** Nombre del objetivo (obligatorio).  
**image:** Imagen del objetivo (opcional).  
**color:** Color asignado al objetivo (opcional).  
**priority:** Prioridad del objetivo (obligatorio).  
**hours:** Horas semanales de dedicación al objetivo (obligatorio).  
**start\_date:** Fecha de inicio del objetivo (obligatorio).  
**end\_date:** Fecha de finalización del objetivo (obligatorio).

**completed:** Estado de finalización del objetivo (obligatorio).

**user\_id:** Identificador del usuario propietario del objetivo (clave foránea).

Relación 1:N con la tabla Task:

Un objetivo puede tener múltiples tareas.

*Ilustración 6. Implementación de la tabla 'objectives'.*

```
class Objective(db.Model):
    __tablename__ = 'objectives'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    image = db.Column(db.String(255), nullable=True)
    color = db.Column(db.String(7), nullable=True)
    priority = db.Column(db.String(100), nullable=False)
    hours = db.Column(db.Integer, nullable=False)
    start_date = db.Column(db.Date, nullable=False)
    end_date = db.Column(db.Date, nullable=False)
    completed = db.Column(db.Boolean, default=False)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    tasks = db.relationship('Task', backref='objective', lazy=True, cascade="all, delete")
```

Fuente: Elaboración propia.

### **Task (Tareas):**

La tabla 'tasks' almacena las tareas asociadas a un objetivo. Una tarea tendrá una prioridad respecto a las demás tareas del objetivo, una cantidad de horas mínimas y máximas diarias a dedicar y una localización además del nombre o título de la tarea.

**id:** Identificador único de la tarea (clave primaria).

**name:** Nombre de la tarea (obligatorio).

**priority:** Prioridad de la tarea (obligatorio).

**min\_hours:** Número de horas diarias mínimas (obligatorio).

**max\_hours:** Número de horas diarias máximas (obligatorio).

**location:** Ubicación de la tarea (opcional).

**objective\_id:** Identificador del objetivo al que pertenece la tarea (clave foránea).

Relación 1:N con la tabla TaskSchedule:

Una tarea puede tener múltiples horarios.

Relación 1:N con la tabla AllocatedTask:

Una tarea puede tener múltiples tareas asignadas.

Ilustración 7. Implementación de la tabla ‘tasks’.

```
class Task(db.Model):
    __tablename__ = 'tasks'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    priority = db.Column(db.String(100), nullable=False)
    min_hours = db.Column(db.Integer, nullable=False)
    max_hours = db.Column(db.Integer, nullable=False)
    location = db.Column(db.String(100), nullable=True)
    objective_id = db.Column(db.Integer, db.ForeignKey('objectives.id'), nullable=False)
    schedules = db.relationship('TaskSchedule', backref='task', cascade='all, delete-orphan')
```

Fuente: Elaboración propia.

### TaskSchedule (Horarios de una Tarea):

La tabla ‘task\_schedules’ define el día de la semana y horario en el que una tarea se puede realizar. Una tarea se podrá realizar en diferentes días y horarios.

**id:** Identificador único del horario de la tarea (clave primaria).

**day\_of\_week:** Día de la semana definido como un Enum con los días de la semana con los valores: ‘Lunes’, ‘Martes’, ‘Miércoles’, ‘Jueves’, ‘Viernes’, ‘Sábado’ y ‘Domingo’ (obligatorio).

**start\_time:** Hora de inicio de la tarea (obligatorio).

**end\_time:** Hora de fin de la tarea (obligatorio).

**task\_id:** Identificador de la tarea al que está asociado el horario (clave foránea).

Ilustración 8. Implementación de la tabla ‘task\_schedules’.

```
class TaskSchedule(db.Model):
    __tablename__ = 'task_schedules'
    id = db.Column(db.Integer, primary_key=True)
    task_id = db.Column(db.Integer, db.ForeignKey('tasks.id'), nullable=False)
    day_of_week = db.Column(db.Enum(DayOfWeek), nullable=False)
    start_time = db.Column(db.Time, nullable=False)
    end_time = db.Column(db.Time, nullable=False)
```

Fuente: Elaboración propia.

### Activity (Actividades):

La tabla ‘activities’ representa las actividades puntuales que realiza un usuario. Tienen una fecha y un horario definido además de un nombre o título. Opcionalmente, se le podrá añadir un texto descriptivo para detallar más información acerca de la actividad y un color.

**id:** Identificador único de la actividad (clave primaria).

**name:** Nombre o título de la actividad (obligatorio).

**date:** Fecha de realización de la actividad (obligatorio).

**start\_time:** Hora de inicio de la actividad (obligatorio).

**end\_time:** Hora de fin de la actividad (obligatorio).

**description:** Descripción o detalles de la actividad (opcional).

**color:** Color asociado a la actividad (opcional).

**user\_id:** Identificador del usuario al que pertenece la actividad (clave foránea).

Ilustración 9. Implementación de la tabla ‘activities’.

```
class Activity(db.Model):
    __tablename__ = 'activities'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    date = db.Column(db.Date, nullable=False)
    start_time = db.Column(db.Time, nullable=False)
    end_time = db.Column(db.Time, nullable=False)
    description = db.Column(db.Text, nullable=True)
    color = db.Column(db.String(7), nullable=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    user = db.relationship('User', backref=db.backref('activities', lazy=True))
```

Fuente: Elaboración propia.

### Routine (Rutinas):

La tabla ‘routines’ almacena las rutinas o hábitos de los usuarios en un día de la semana concreto en un horario concreto.

**id:** Identificador único de la rutina (clave primaria).

**name:** Nombre o título de la rutina (obligatorio).

**description:** Descripción o detalles de la rutina (opcional).

**color:** Color asociado a la rutina (opcional).

**user\_id:** Identificador del usuario al que pertenece la rutina (clave foránea).

Relación 1:N con la tabla RoutineSchedule:

Una rutina puede tener múltiples horarios.

Ilustración 10. Implementación de la tabla ‘routines’.

```
class Routine(db.Model):
    __tablename__ = 'routines'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=True)
    color = db.Column(db.String(7), nullable=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    user = db.relationship('User', backref=db.backref('routines', lazy=True))
    routine_schedules = db.relationship('RoutineSchedule', backref='routine', cascade='all, delete-orphan')
```

Fuente: Elaboración propia.

### RoutineSchedule (Horarios de una Rutina):

La tabla ‘routine\_schedules’ define los horarios específicos en los que una rutina se repite, especificando el día de la semana, la hora de inicio y la hora de fin.

**id:** Identificador único del horario de la rutina (clave primaria).

**day\_of\_week:** Día de la semana definido como un Enum con los días de la semana con los valores: ‘Lunes’, ‘Martes’, ‘Miércoles’, ‘Jueves’, ‘Viernes’, ‘Sábado’ y ‘Domingo’ (obligatorio).

**start\_time:** Hora de inicio de la rutina (obligatorio).

**end\_time:** Hora de fin de la rutina (obligatorio).

**routine\_id:** Identificador de la rutina al que está asociado el horario (clave foránea).

Ilustración 11. Implementación de la tabla ‘routine\_schedules’.

```
class RoutineSchedule(db.Model):
    __tablename__ = 'routine_schedules'
    id = db.Column(db.Integer, primary_key=True)
    routine_id = db.Column(db.Integer, db.ForeignKey('routines.id'), nullable=False)
    day_of_week = db.Column(db.Enum(DayOfWeek), nullable=False)
    start_time = db.Column(db.Time, nullable=False)
    end_time = db.Column(db.Time, nullable=False)
```

Fuente: Elaboración propia.

### Notification (Notificaciones):

La tabla ‘notifications’ almacena las notificaciones o alertas que reciben los usuarios.

**id:** Identificador único de la notificación (clave primaria).

**date:** Fecha de creación de la notificación (obligatorio).

**time:** Hora de creación de la notificación (obligatorio).

**title:** Título de la notificación (obligatorio).

**description:** Descripción de la notificación (opcional).

**user\_id:** Identificador del usuario al que se le envía la notificación (clave foránea).

Ilustración 12. Implementación de la tabla ‘notifications’.

```
class Notification(db.Model):
    __tablename__ = 'notifications'
    id = db.Column(db.Integer, primary_key=True)
    date = db.Column(db.Date, nullable=False)
    time = db.Column(db.Time, nullable=False)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    user = db.relationship('User', backref=db.backref('notifications', lazy=True, cascade="all, delete"))
```

Fuente: Elaboración propia.

### AllocatedTask (Tareas asignadas):

La tabla ‘allocated\_tasks’ almacena las tareas asignadas en la agenda de un usuario. Cada tarea asignada tendrá el nombre de la tarea y el identificador de esa tarea. Además, tendrá una fecha y horas de comienzo y finalización junto a las horas de dedicación (duración) de la tarea. Cada tarea asignada contendrá un valor booleano informando si ha sido marcada como completada por el usuario o no.

**id:** Identificador único de la tarea asignada (clave primaria).

**name:** Nombre de la tarea (obligatorio).

**date:** Fecha asignada a la tarea (obligatorio).

**start\_time:** Hora asignada para comenzar la tarea (obligatorio).

**end\_time:** Hora asignada para finalizar la tarea (obligatorio).

**dedicated\_hours:** Número de horas que durará la tarea (obligatorio).

**completed:** Booleano indicando si la tarea ha sido completada o no (obligatorio).

**task\_id:** Identificador de la tarea a la que pertenece la tarea asignada (clave foránea).

**user\_id:** Identificador del usuario al que pertenece la tarea asignada (clave foránea).

*Ilustración 13. Implementación de la tabla 'allocated\_tasks'.*

```
class AllocatedTask(db.Model):
    __tablename__ = 'allocated_tasks'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    date = db.Column(db.Date, nullable=False)
    start_time = db.Column(db.Time, nullable=False)
    end_time = db.Column(db.Time, nullable=False)
    dedicated_hours = db.Column(db.Integer, nullable=False)
    completed = db.Column(db.Boolean, default=False)
    task_id = db.Column(db.Integer, db.ForeignKey('tasks.id'), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    task = db.relationship('Task', backref='allocated_tasks', lazy=True)
```

Fuente: Elaboración propia.

## 5.4. Vista de Usuario

En esta sección, se detallará para cada página del sitio web además del header y el footer (presentes en muchas de esas páginas) su propósito. Ahí, se describirá brevemente su utilidad. Además, se listará el contenido de cada página en orden de arriba abajo y de izquierda a derecha junto a la función que desempeña dicho elemento. Se añadirá una imagen de cada página al completo. Algunas de estas imágenes contendrán casos de ejemplo. Por último, se detallarán algunos de los aspectos más relevantes de cada página con ilustraciones mostrando extractos del código.

### 5.4.1. Header

#### Propósito:

El header, ubicado en la parte superior de la página, permitirá a los usuarios que hayan iniciado sesión cargar las páginas con las diferentes secciones de la aplicación.

#### Contenido y funcionalidad:

- Botón con el icono de una diana que dirigirá al usuario a la página de objetivos.
- Botón con el icono de un calendario que dirigirá al usuario a la página de la agenda.
- Botón con el icono de un gráfico que dirigirá al usuario a la página de progreso.
- Botón con el icono de un cajón que dirigirá al usuario a la página de notificaciones.
- Botón con el icono de una persona que dirigirá al usuario a la página de perfil.

## Imagen:

Ilustración 14. Header.



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/header.html

Estilo CSS: static/styles/header.css

En el archivo ‘header.html’, se han ordenado los cinco botones que conforman las diferentes secciones principales de la aplicación (objetivos, agenda, progreso, notificaciones y perfil). Cada botón perteneciente a la clase ‘icon-button’ y está representado por un ícono de la librería Fontawesome. Por ejemplo, la diana que representa la sección de objetivos se identifica como ‘fas fa-bullseye’. Además, al hacer clic sobre uno de los botones, esté dirigirá al usuario a la pantalla pertinente. En el caso del botón de objetivos, la aplicación cargará la ruta ‘/objectives’ implementada en app.py. Por último, se comprueba si la página activa es la correspondiente a ese botón. Esto hace que el ícono adopte el color naranja y reduzca ligeramente su tamaño para mejorar la experiencia del usuario al claramente identificar qué sección está vigentemente activa.

Ilustración 15. Implementación de ‘header.html’.

```
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><button id="objectivesButton" class="icon-button {% if active_page == 'objectives' %}active{%-endif%}"><i class="fas fa-bullseye"></i></button></li>
        <li><button id="agendaButton" class="icon-button {% if active_page == 'agenda' %}active{%-endif%}"><i class="far fa-calendar-alt"></i></button></li>
        <li><button id="progressButton" class="icon-button {% if active_page == 'progress' %}active{%-endif%}"><i class="fas fa-chart-line"></i></button></li>
        <li><button id="inboxButton" class="icon-button {% if active_page == 'inbox' %}active{%-endif%}"><i class="fas fa-inbox"></i></button></li>
        <li><button id="profileButton" class="icon-button {% if active_page == 'profile' %}active{%-endif%}"><i class="fas fa-user"></i></button></li>
      </ul>
    </nav>
  </header>
</body>
</html>
```

Fuente: Elaboración propia.

Para cargar el header en las páginas a las que le corresponda, en los HTML pertinentes se incluye el header al principio del cuerpo de la página.

Ilustración 16. Inclusión del header en ‘profile.html’.

```
<title>Profile</title>
</head>
<body>
  {% include 'header.html' %}
```

Fuente: Elaboración propia.

## 5.4.2. Footer

### Propósito:

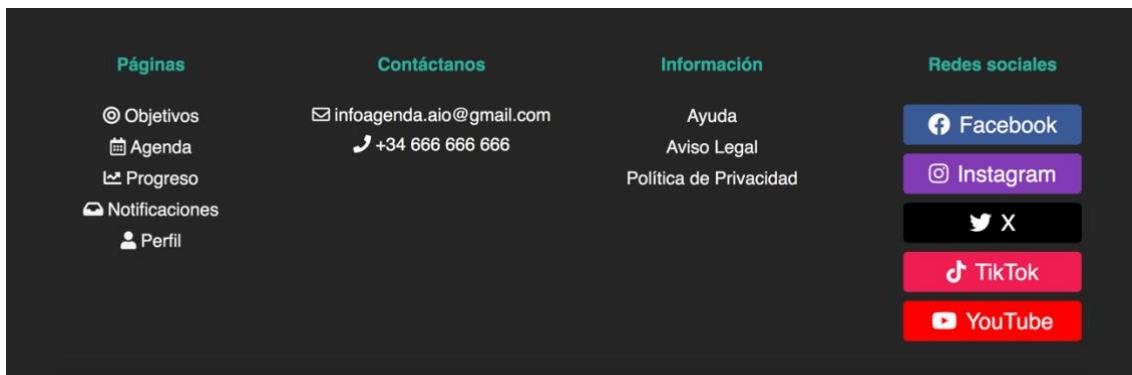
El footer, ubicado en la parte inferior de la página, permitirá a los usuarios acceder a las diferentes secciones del sitio web sin necesidad de desplazarse hasta el header. Además, ofrecerá la información de contacto como el correo electrónico y el número de teléfono para recibir soporte. En el apartado de información, encontrará los botones que dirigirán a las páginas de 'Ayuda', 'Aviso Legal' y 'Política de Privacidad' y, por último, los botones que carguen las páginas de las redes sociales Facebook, Instagram, X, TikTok y YouTube.

### Contenido y funcionalidad:

- Título 'Páginas' con los botones para acceder a las páginas de objetivos, agenda, progreso, notificaciones y perfil.
- Título 'Contáctanos' informando del correo electrónico y el número de teléfono para soporte.
- Título 'Información' con los botones que carguen las páginas con la ayuda, Aviso Legal y Política de Privacidad.
- Título 'Redes Sociales' con los botones que dirijan al usuario a las cuentas en redes sociales de Facebook, Instagram, X, TikTok y YouTube.

### Imagen:

Ilustración 17. Footer



Fuente: Elaboración propia.

### Desarrollo técnico:

Plantilla HTML: templates/footer.html

Estilo CSS: static/styles/footer.css

En el archivo 'footer.html', se han colocado las diferentes secciones que ocupan el footer (páginas, contacto, información y redes sociales). Estas secciones se han dividido por la clase 'column' al cada una representar una columna en el diseño.

Ilustración 18. Implementación de 'footer.html'.

```
<footer>
  <div class="column">
    <h5 class="title">Páginas</h5>
    <ul class="list">
      <li><button class="item" {% if active_page == 'objectives' %}active{% endif %}>
        onclick="location.href='/objectives'"><i class="fas fa-bullseye"></i>Objetivos</button></li>
      <li><button class="item" {% if active_page == 'agenda' %}active{% endif %}>
        onclick="location.href='/agenda'"><i class="far fa-calendar-alt"></i>Agenda</button></li>
      <li><button class="item" {% if active_page == 'progress' %}active{% endif %}>
        onclick="location.href='/progress'"><i class="fas fa-chart-line"></i>Progreso</button></li>
      <li><button class="item" {% if active_page == 'inbox' %}active{% endif %}>
        onclick="location.href='/inbox'"><i class="fas fa-inbox"></i>Notificaciones</button></li>
      <li><button class="item" {% if active_page == 'profile' %}active{% endif %}>
        onclick="location.href='/profile'"><i class="fas fa-user"></i>Perfil</button></li>
    </ul>
  </div>
  <div class="column">
    <h5 class="title">Contáctanos</h5>
    <ul class="list">
      <li class="contact"><i class="far fa-envelope"></i>infoagenda.aio@gmail.com</li>
      <li class="contact"><i class="fas fa-phone"></i>+34 666 666 666</li>
    </ul>
  </div>
```

Fuente: Elaboración propia.

Para cargar el footer en las páginas a las que le corresponda, se incluye el footer al final del cuerpo de la página en los HTML.

Ilustración 19. Inclusión del footer en 'profile.html'.

```
<script src="{{ url_for('static', filename='js/profile.js') }}"></script>

  {% include 'footer.html' %}
</body>
</html>
```

Fuente: Elaboración propia.

### 5.4.3. Portal de Usuario

#### Propósito:

El Portal de Usuario será la página de bienvenida a los usuarios a la aplicación web.

#### Contenido y funcionalidad:

- Nombre de la aplicación.
- Botón 'Inicia Sesión' que dirigirá al usuario al formulario para el inicio de sesión.
- Botón 'Regístrate' que dirigirá al usuario al formulario para el registro de nuevos usuarios.
- Carrusel con iconos y resúmenes de las funcionalidades de la aplicación.
- Footer.

## Imagen:

Ilustración 20. Página de Portal de Usuario



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/index.html  
Estilo CSS: static/styles/index.css

El Portal de Usuario tiene una lógica muy básica ya que tan solo muestra el título de la aplicación, los botones para el inicio de sesión y el registro que se encuentran en el contenedor 'buttons'. Al hacer clic sobre ellos se cargarán las rutas '/login' y '/register' respectivamente. Además, el contenedor 'functions-container' contendrá las subclases 'function' en las que se establecen el ícono, título y descripción de cada una de las secciones de la aplicación web. Por último, se incluye el footer.

Ilustración 21. Implementación de ‘index.html’.

```
<body>
    <div class="login-container">
        <div class="text">
            <h1>AIO</h1>
            <p id="subtitle">Agenda Inteligente para la gestión eficiente de Objetivos personales</p>
        </div>
        <div class="buttons">
            <button class="btn" onclick="window.location.href='/login'">Inicia Sesión</button>
            <button class="btn" onclick="window.location.href='/register'">Regístrate</button>
        </div>
    </div>
    <div class="functions-container">
        <div class="function">
            <i class="fas fa-bullseye"></i>
            <h3 class="function-title">Objetivos</h3>
            <p class="function-description">Marca tus objetivos</p>
        </div>
        <div class="function">
            <i class="far fa-calendar-alt"></i>
            <h3 class="function-title">Agenda</h3>
            <p class="function-description">Consulta tu agenda</p>
        </div>
        <div class="function">
            <i class="fas fa-chart-line"></i>
            <h3 class="function-title">Progreso</h3>
            <p class="function-description">Monitoriza tu progreso</p>
        </div>
        <div class="function">
            <i class="fas fa-inbox"></i>
            <h3 class="function-title">Notificaciones</h3>
            <p class="function-description">Atento a las alertas</p>
        </div>
    </div>

```

Fuente: Elaboración propia.

Empleando el framework Flask, se define que para la ruta en la URL (Uniform Resource Locators) ‘/’ se cargue el contenido de la función ‘index()’ que en este caso tan solo contiene la carga de la página ‘index.html’.

Ilustración 22. Implementación de ‘index()’.

```
@app.route('/')
def index():
    return render_template('index.html')
```

Fuente: Elaboración propia.

## 5.4.4. Registro

### Propósito:

La página de registro permitirá la creación de nuevas cuentas para usuarios a través de un formulario.

### Contenido y funcionalidad:

- Botón con el icono de una flecha apuntando hacia la izquierda que dirigirá al portal de usuario.
- Título de la página: ‘Regístrate’.

- Campo ‘Foto de perfil’ junto al botón ‘Seleccionar archivo’ para que el usuario seleccione un archivo de tipo imagen de su dispositivo.
- Campo obligatorio ‘Nombre de usuario’.
- Campo ‘Correo electrónico’.
- Campo ‘Teléfono’.
- Campo obligatorio ‘Contraseña’ junto al botón con el icono de un ojo para cambiar la visibilidad del contenido del campo.
- Campo obligatorio ‘Confirmar contraseña’ junto al botón con el icono de un ojo para cambiar la visibilidad del contenido del campo.
- Casilla de verificación obligatoria para aceptar la Política de Privacidad.
- Casilla de verificación obligatoria para aceptar el Aviso Legal.
- Botón ‘Regístrate’ para dar de alta la cuenta con los datos introducidos en los campos y dirigir al usuario a la página de objetivos en caso de introducir los datos correctamente.
- Botón ‘Inicia Sesión’ que cargará la página de inicio de sesión.
- Botón de ayuda ‘i’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

### Imagen:

Ilustración 23. Página de registro.

The screenshot shows a registration form titled "Regístrate". The form includes fields for profile photo, user name, email, phone number, password, and password confirmation. It also features checkboxes for accepting the Privacy Policy and Legal Notice, and a "Register" button. At the bottom, there are links for existing users and a "Log In" button. The footer contains navigation links for Pages, Contact Us, Information, and Social Media.

Páginas	Contactanos	Información	Redes sociales
Objetivos Agenda Progreso Notificaciones Perfil	infoagenda.aio@gmail.com +34 666 666 666	Ayuda Aviso Legal Política de Privacidad	<a href="#">Facebook</a> <a href="#">Instagram</a> <a href="#">X</a> <a href="#">TikTok</a> <a href="#">YouTube</a>

Fuente: Elaboración propia.

## **Desarrollo técnico:**

Plantilla HTML: templates/register.html

Estilo CSS: static/styles/register.css

JavaScript: static/js/register.js

Para el formulario de registro estructurado en templates/register.html y diseñado en static/styles/register.css, se han realizado las validaciones necesarias en static/js/register.js para garantizar que los datos introducidos son válidos. Para ello, al enviar el formulario se comprueba que todos los campos obligatorios han sido rellenados. En caso contrario, un aviso saltará indicando el campo que aún ha de ser completado.

*Ilustración 24. Aviso de campo obligatorio.*

The screenshot shows a registration form with two fields. The first field is labeled "Foto de perfil:" with a file input button "Seleccionar archivo" and the message "Ninguno archivo selec.". The second field is labeled "Nombre de usuario:" with a text input field containing a placeholder "Completa este campo". Below the input field is a tooltip with the text "Completa este campo" and a small exclamation mark icon. The entire form is set against a light blue background.

Fuente: Elaboración propia.

En caso de que al menos los campos necesarios han sido rellenados y el usuario hace clic en el botón ‘Regístrate’, se envía el formulario y se procede a la validación de cada uno de los campos. Primeramente, se comprueba que la longitud del nombre de usuario no sobrepasa los diez caracteres. Después, a través de la función ‘validatePhone(phone)’ se comprueba que el número de teléfono solo contiene números y tiene una longitud de entre nueve y quince caracteres. Si esa validación prospera, se comprueba que los campos de ‘Contraseña’ y ‘Confirmar contraseña’ contienen exactamente los mismos valores. Finalmente, si las contraseñas introducidas coinciden, se salta a la función ‘validatePassword(password1, user)’ que toma la contraseña introducida además del nombre de usuario para validar que la contraseña cumple con todos los parámetros obligatorios. Estos son: al menos ocho caracteres de longitud, contener al menos una letra mayúscula, contener al menos una letra minúscula, contener al menos un número, contener al menos un símbolo de los establecidos en la variable ‘hasSpecialChar’ y que la contraseña no contenga el nombre de usuario.

Ilustración 25. Validación del formulario de registro en 'register.js'.

```
document.getElementById('formRegister').addEventListener('submit', function(event) {
    var password1 = document.getElementById('password1').value;
    var password2 = document.getElementById('password2').value;
    var user = document.getElementById('user').value;
    var phone = document.getElementById('phone_number').value;

    if (user.length > 10) {
        alert('El nombre de usuario no debe tener más de 10 caracteres.');
        event.preventDefault();
        return;
    }

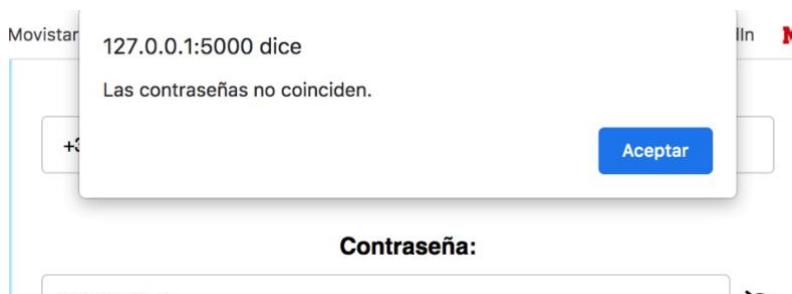
    if (!validatePhone(phone)) {
        event.preventDefault();
        return;
    }

    if (password1 !== password2) {
        alert('Las contraseñas no coinciden.');
        event.preventDefault();
        return;
    }

    if (!validatePassword(password1, user)) {
        event.preventDefault();
    }
});
```

Fuente: Elaboración propia.

Ilustración 26. Alerta por error en el registro.



Fuente: Elaboración propia.

Ilustración 27. Implementación de validatePhone(phone).

```
function validatePhone(phone) {  
    if (phone === "") {  
        return true;  
    }  
  
    var phoneRegex = /^\d+$/;  
  
    if (!phoneRegex.test(phone)) {  
        alert('El número de teléfono solo debe contener números.');//  
        return false;  
    }  
    if (phone.length < 9 || phone.length > 15) {  
        alert('El número de teléfono debe tener entre 9 y 15 dígitos.');//  
        return false;  
    }  
  
    return true;  
}
```

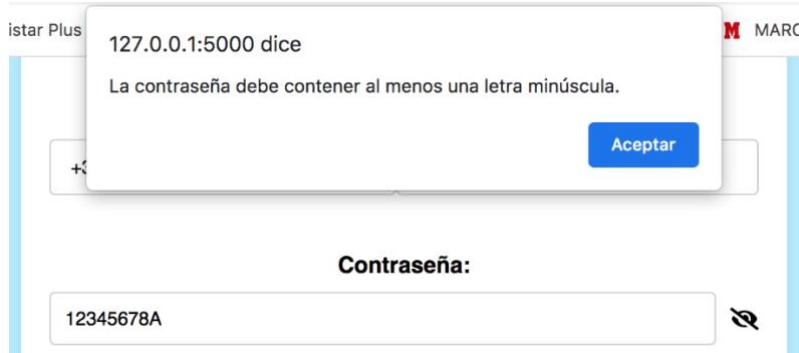
Fuente: Elaboración propia.

Ilustración 28. Implementación de validatePassword(password, username).

```
function validatePassword(password, username) {  
    var minLength = 8;  
    var hasUpperCase = /[A-Z]/.test(password);  
    var hasLowerCase = /[a-z]/.test(password);  
    var hasNumber = /\d/.test(password);  
    var hasSpecialChar = /[@#$%^&*(),.?{}|<>]/.test(password);  
    var containsUsername = password.includes(username);  
  
    if (password.length < minLength) {  
        alert('La contraseña debe tener al menos 8 caracteres.');//  
        return false;  
    }  
    if (!hasUpperCase) {  
        alert('La contraseña debe contener al menos una letra mayúscula.');//  
        return false;  
    }  
    if (!hasLowerCase) {  
        alert('La contraseña debe contener al menos una letra minúscula.');//  
        return false;  
    }  
    if (!hasNumber) {  
        alert('La contraseña debe contener al menos un número.');//  
        return false;  
    }  
    if (!hasSpecialChar) {  
        alert('La contraseña debe contener al menos un símbolo válido.');//  
        return false;  
    }  
    if (containsUsername) {  
        alert('La contraseña no puede contener el nombre de usuario.');//  
        return false;  
    }  
    return true;  
}
```

Fuente: Elaboración propia.

Ilustración 29. Alerta de error en la validación de la contraseña.



Fuente: Elaboración propia.

Obligatoriamente, el usuario deberá seleccionar las casillas de confirmación de aceptación de la Política de Privacidad y de Aviso Legal para registrarse.

Ilustración 30. Aviso de la aceptación obligatoria de la Política de Privacidad.



Fuente: Elaboración propia.

Una vez el formulario pasa los filtros de validación, los datos de los campos son recibidos por la aplicación en la función 'register()' en app.py. A partir de ahí, se le aplica un algoritmo de hash a la contraseña empleando la función 'generate\_password\_hash(password)' de la librería Werkzeug.

Ilustración 31. Recogida de datos del formulario de registro en 'register()' .

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        image = request.files['imageInput']
        username = request.form['user']
        email = request.form['email']
        phone_country_code = request.form['country_code']
        phone_number = request.form['phone_number']
        password = request.form['password1']
        hashed_password = generate_password_hash(password)
```

Fuente: Elaboración propia.

Después, se almacena la imagen de perfil en caso de haber sido seleccionada y se comprueba si el nombre de usuario, el correo electrónico o el número de teléfono ya están registrados en la aplicación. En tal caso, se mostrará un

mensaje de error con el texto de color rojo en la página de registro encima del botón ‘Regístrate’.

Ilustración 32. Comprobación de que el correo electrónico no está registrado en ‘register()’.

```
if not email:  
    email = None  
else:  
    existing_email = User.query.filter_by(email=email).first()  
    if existing_email:  
        message = "El correo electrónico ya está en uso por otro usuario."  
        return render_template('register.html', message=message)
```

Fuente: Elaboración propia.

Ilustración 33. Mensaje de error si el nombre de usuario ya está registrado.



Fuente: Elaboración propia.

Finalmente, si todos los campos son válidos y ninguno de los datos nombrados anteriormente ya están registrados en la base de datos, se procede a la creación del nuevo usuario y se añade a la base de datos. El usuario es dirigido a la página de objetivos.

Ilustración 34. Creación de un nuevo usuario en la base de datos en ‘register()’.

```
new_user = User(image=filename, username=username, email=email,  
                phone_country_code=phone_country_code, phone_number=phone_number,  
                password=hashed_password, agenda_color='#2eb19f')  
db.session.add(new_user)  
db.session.commit()
```

Fuente: Elaboración propia.

## 5.4.5. Inicio de Sesión

### Propósito:

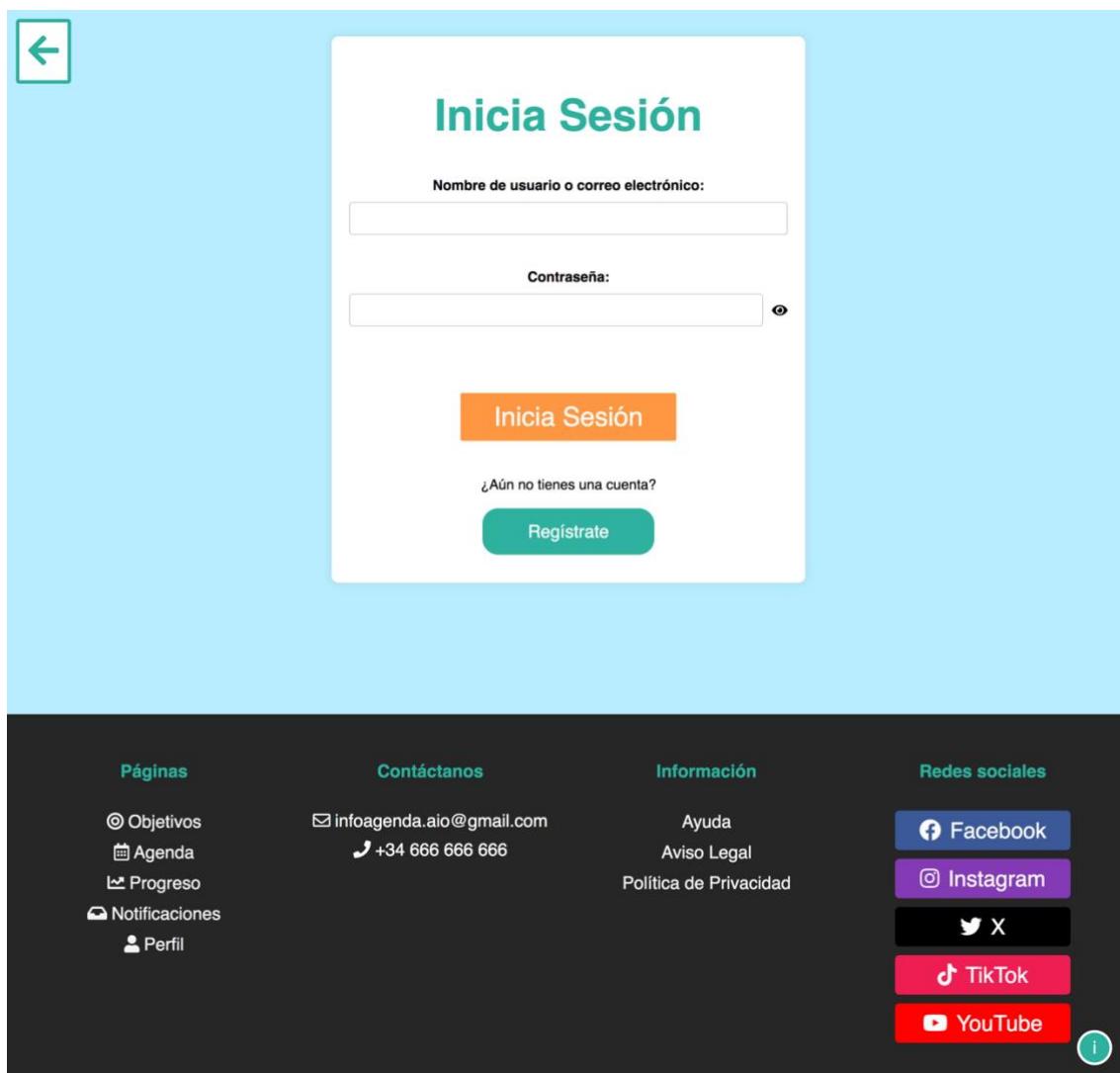
La página de inicio de sesión permitirá a un usuario que se haya registrado previamente acceder a las funcionalidades de la aplicación en su cuenta. Para iniciar sesión, un usuario deberá ingresar en el primer campo su nombre de usuario o el correo electrónico. En el segundo campo, la contraseña.

## Contenido y funcionalidad:

- Botón con el icono de una flecha apuntando hacia la izquierda que dirigirá al portal de usuario.
- Título de la página: ‘Inicia Sesión’.
- Campo obligatorio ‘Nombre de usuario o correo electrónico’.
- Campo obligatorio ‘Contraseña’ junto al botón con el icono de un ojo para cambiar la visibilidad del contenido del campo.
- Botón ‘Inicia Sesión’ que cargará la página de la agenda si los datos introducidos en los campos son correctos.
- Botón ‘Regístrate’ que cargará la página de registro bajo el texto ‘¿Aún no tienes una cuenta?’.
- Botón de ayuda ‘i’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

## Imagen:

Ilustración 35. Página de inicio de sesión.



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/login.html

Estilo CSS: static/styles/login.css

JavaScript: static/js/login.js

Este formulario tan solo cuenta con dos campos para iniciar sesión: nombre de usuario o correo electrónico y contraseña. Al igual que en el registro, el campo de la contraseña viene acompañado de un botón con el icono de un ojo. Por defecto, por motivos de seguridad, cuando el usuario introduzca la contraseña, ésta estará oculta. En caso de que el usuario desee ver el contenido del campo, haciendo clic en este botón con el icono de un ojo, el contenido será visible y el icono del ojo cambiará al icono de un ojo tachado. Al hacer clic sobre el botón con el icono del ojo tachado, se realizará la labor inversa y el contenido del campo volverá a estar en modo oculto y el icono volverá a ser el de un ojo. Para implementar esta funcionalidad, se ha desarrollado la función 'togglePassword(inputId, eyeId)'.

*Ilustración 36. Implementación de 'togglePassword(inputId, eyeId)'.*

```
function togglePassword(inputId, eyeId) {
    var passwordInput = document.getElementById(inputId);
    var eyeIcon = document.getElementById(eyeId);

    if (passwordInput.type === "password") {
        passwordInput.type = "text";
        eyeIcon.classList.remove("fa-eye");
        eyeIcon.classList.add("fa-eye-slash");
    } else {
        passwordInput.type = "password";
        eyeIcon.classList.remove("fa-eye-slash");
        eyeIcon.classList.add("fa-eye");
    }
}
```

Fuente: Elaboración propia.

En app.py, se encuentra la función 'login()' que recibe los datos del formulario de inicio de sesión y realiza las comprobaciones de que en la base de datos el nombre de usuario o el correo electrónico introducido coincide con la contraseña. La contraseña en el momento del registro fue almacenada con un algoritmo de hash aplicado. Por ello, para poder hacer la comprobación, se aplica la función check\_password\_hash(user.password, password) de la librería Werkzeug que comparará el campo introducido por el usuario con la contraseña almacenada en la base de datos y devolverá verdadero si coinciden.

Ilustración 37. Implementación de 'login()'.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['user']
        password = request.form['password']

        user = User.query.filter_by(username=username).first()
        user2 = User.query.filter_by(email=username).first()
        if user and check_password_hash(user.password, password):
            session['user_id'] = user.id
            return redirect('/agenda')
        elif user2 and check_password_hash(user2.password, password):
            session['user_id'] = user2.id
            return redirect('/agenda')
        else:
            message = "Nombre de usuario y/o contraseña incorrectos"
            return render_template('Login.html', message=message)

    return render_template('login.html')
```

Fuente: Elaboración propia.

## 5.4.6. Objetivos

### Propósito:

La página de objetivos mostrará al usuario todos los objetivos personales creados. Los objetivos se dividirán en cuatro categorías. La primera será la de objetivos actuales en la que figurarán todos los objetivos que estén vigentes en el periodo establecido y aún no hayan sido marcados como completados. Seguidamente, se mostrarán los objetivos completados. Éstos serán los objetivos que, sin importar el periodo establecido, hayan sido marcados como completados. Después, aparecerán los objetivos pasados que son aquellos objetivos no completados cuya fecha de fin ya haya pasado. Finalmente, estarán los objetivos futuros que son aquellos objetivos no completados cuya fecha de inicio es posterior a la fecha actual.

### Contenido y funcionalidad:

- Header.
- Título de la página: 'Mis Objetivos'.
- Botón 'Crear Objetivo'.
- Título: 'Objetivos Actuales' con todos los objetivos actuales.
- Título: 'Objetivos Completados' con todos los objetivos completados.
- Título: 'Objetivos Pasados' con todos los objetivos pasados.
- Título: 'Objetivos Futuros' con todos los objetivos futuros.
- Botón de ayuda 'í' que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

## Imagen:

Ilustración 38. Página de objetivos.

The screenshot shows the 'Mis Objetivos' (My Goals) section of a mobile application. At the top, there is a navigation bar with icons for search, calendar, graph, car, and profile. Below the title 'Mis Objetivos', there is a button '+ Crear Objetivo'. The 'Objetivos Actuales' (Current Goals) section contains two items: 'Hacer deporte' (Orange box) and 'Aprender a pintar' (Teal box). Each item has a thumbnail image and a description. Below this, there are sections for 'Objetivos Completados' (Completed Goals) and 'Objetivos Pasados' (Past Goals), both of which are currently empty. The footer of the app includes links to 'Páginas' (Pages), 'Contáctanos' (Contact us), 'Información' (Information), and 'Redes sociales' (Social media). The 'Páginas' section lists 'Objetivos', 'Agenda', 'Progreso', 'Notificaciones', and 'Perfil'. The 'Contáctanos' section provides an email address 'infoagenda.aio@gmail.com' and a phone number '+34 666 666 666'. The 'Información' section links to 'Ayuda', 'Aviso Legal', and 'Política de Privacidad'. The 'Redes sociales' section lists links for Facebook, Instagram, X (Twitter), TikTok, and YouTube.

Mis Objetivos

+ Crear Objetivo

Objetivos Actuales

Hacer deporte

Aprender a pintar

Objetivos Completados

No tienes objetivos completados.

Objetivos Pasados

No tienes objetivos pasados.

Objetivos Futuros

No tienes objetivos futuros.

Páginas

- Objetivos
- Agenda
- Progreso
- Notificaciones
- Perfil

Contáctanos

infoagenda.aio@gmail.com  
+34 666 666 666

Información

Ayuda  
Aviso Legal  
Política de Privacidad

Redes sociales

Facebook  
Instagram  
X  
TikTok  
YouTube

Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/my\_objectives.html

Estilo CSS: static/styles/my\_objectives.css

JavaScript: static/js/my\_objectives.js

En app.py se ha creado la función ‘show\_objectives()’ que es invocada cuando la ruta de la URL es ‘/objectives’. La lógica implementada en esta función divide los objetivos del usuario en la base de datos entre los objetivos actuales, completados, pasados y futuros. Para ello, utilizando la librería datetime, se conoce cuál es la fecha actual y basado en ella, se filtra mediante sentencias SQL la base de datos. Una vez se obtienen las listas con los objetivos diferenciados, éstos se mandan a ‘my\_objectives.html’.

Ilustración 39. Implementación de ‘show\_objectives()’.

```
@app.route('/objectives', methods=['GET', 'POST'])
def show_objectives():
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

    today = datetime.now().date()

    objectives_current = Objective.query.filter_by(user_id=user_id, completed=False).filter(
        Objective.start_date <= today, (Objective.end_date >= today) | (Objective.end_date == None)).all()

    objectives_completed = Objective.query.filter_by(user_id=user_id, completed=True).all()

    objectives_past = Objective.query.filter_by(user_id=user_id, completed=False).filter(
        Objective.end_date < today).all()

    objectives_future = Objective.query.filter_by(user_id=user_id, completed=False).filter(
        Objective.start_date > today).all()

    return render_template(
        'my_objectives.html',
        objectives_current=objectives_current,
        objectives_completed=objectives_completed,
        objectives_past=objectives_past,
        objectives_future=objectives_future,
        active_page='objectives')
```

Fuente: Elaboración propia.

En ‘my\_objectives.html’, se proyectan todos estos objetivos. Cada objetivo se proyectará como un cuadrado del color asociado al objetivo en el que se visualizará la imagen del objetivo sobre el nombre del objetivo. En caso de que un objetivo no tenga una imagen asociada, se cargará la imagen de objetivo por defecto ‘objective.png’. En caso contrario, se cargará la imagen del objetivo correspondiente almacenada en el directorio ‘static/images/objectives’. En caso de no existir ningún objetivo en alguna de las categorías, se mostrará un mensaje indicando que no existen objetivos de esa categoría. Por ejemplo, ‘No tienes objetivos actuales.’

Ilustración 40. Implementación de 'my\_objectives.html'.

```
<h2>Mis Objetivos</h2>

<button id="btnAddObjective"><i class="fas fa-plus"></i>Crear Objetivo</button>

<h3>Objetivos Actuales</h3>
{%
  if objectives_current %
    <ul class="objectives">
      {% for objective in objectives_current %}
        <button class="btnObjective" data-id="{{ objective.id }}" style="background-color: {{ objective.color }}; border: none; width: 100px; height: 100px; position: relative; overflow: hidden;">
          <div class="image-container">
            {% if objective.image %}
              
          <div class="text-container">
            <p class="objective_name">{{ objective.name }}</p>
          </div>
        </button>
      {% endfor %}
    </ul>
  {% else %}
    <p class="no_objectives">No tienes objetivos actuales.</p>
  {% endif %}
}

<h3>Objetivos Completados</h3>
```

Fuente: Elaboración propia.

## 5.4.7. Nuevo Objetivo

### Propósito:

La página de nuevo objetivo contiene el formulario con los campos para la creación de un objetivo. Está compuesto por los campos básicos obligatorios como el nombre, la prioridad respecto al resto de objetivos, las horas semanales de dedicación, la fecha de inicio y la fecha de finalización. Además, se podrá personalizar mediante una imagen opcional y un color de objetivo que, por defecto, será verde.

### Contenido y funcionalidad:

- Header.
- Título de la página: 'Nuevo Objetivo'.
- Campo obligatorio 'Nombre del objetivo'.
- Campo 'Imagen del objetivo' junto al botón 'Seleccionar archivo' para que el usuario seleccione un archivo de tipo imagen de su dispositivo.
- Campo 'Color del objetivo'.
- Campo obligatorio 'Prioridad'.
- Campo obligatorio 'Horas semanales'.
- Campo obligatorio 'Fecha de inicio'.
- Campo obligatorio 'Fecha de fin'.
- Botón 'Crear Objetivo'.
- Botón de ayuda 'í' que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

## Imagen:

Ilustración 41. Página de nuevo objetivo.

Nuevo Objetivo

Nombre del objetivo:

Imagen del objetivo:  
Seleccionar archivo Ninguno archivo selec.

Color del objetivo:

Prioridad:  50

Horas semanales: 1

Fecha de inicio: dd/mm/aaaa

Fecha de finalización: dd/mm/aaaa

Crear Objetivo

Páginas

- Objetivos
- Agenda
- Progreso
- Notificaciones
- Perfil

Contáctanos

infoagenda.aio@gmail.com  
+34 666 666 666

Información

Ayuda  
Aviso Legal  
Política de Privacidad

Redes sociales

Facebook  
Instagram  
X  
TikTok  
YouTube

Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/objective\_form.html

Estilo CSS: static/styles/objective\_form.css

JavaScript: static/js/objective\_form.js

En el momento en el que todos los datos obligatorios son introducidos en el formulario, éstos serán recibidos por la función 'objective\_form()' en app.py donde se almacena la imagen y se valida que la fecha de inicio sea anterior a la

fecha de finalización del objetivo. El nuevo objetivo es almacenado en la base de datos y se genera una notificación de creación de objetivo.

Ilustración 42. Implementación de 'objective\_form()'.

```
@app.route('/objective_form', methods=['GET', 'POST'])
def objective_form():
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

    if request.method == 'POST':
        # Recogida de datos del formulario de objetivo
        name = request.form['name']
        image = request.files['imageInput']
        color = request.form['color']
        priority = request.form['priority']
        hours = request.form['hours']
        start_date_str = request.form['start_date']
        end_date_str = request.form['end_date']
        user_id = session.get('user_id')

        if image:
            filename = secure_filename(image.filename)
            image_path = os.path.join(app.config['OBJECTIVE_IMAGES'], filename)
            image.save(image_path)
        else:
            filename = None

        if not start_date_str or not end_date_str:
            return render_template('objective_form.html',
                                   error_message="La fecha de inicio y la fecha de finalización son obligatorias.")

        start_date = datetime.strptime(start_date_str, '%Y-%m-%d').date()
        end_date = datetime.strptime(end_date_str, '%Y-%m-%d').date()

        if end_date < start_date:
            return render_template('objective_form.html',
                                   error_message="La fecha de finalización debe ser mayor o igual a la fecha de inicio.")

        new_objective = Objective(name=name, image=filename, color=color, priority=priority, hours=hours,
                                  start_date=start_date, end_date=end_date, user_id=user_id)
        db.session.add(new_objective)
        db.session.commit()

        create_notification(user_id, "Nuevo Objetivo", f"Has creado un nuevo objetivo: {name}.")

        return redirect('/objective/{0}'.format(new_objective.id))

    return render_template('objective_form.html', active_page='objectives')
```

Fuente: Elaboración propia.

Al igual que en el momento del registro, la creación de un objetivo permite al usuario seleccionar un archivo de su dispositivo de tipo imagen. Una vez la imagen es seleccionada, se muestra una vista previa para que el usuario tenga la certeza del archivo que ha seleccionado. Para esto, se implementa la función 'previewImage()'.

Ilustración 43. Implementación de 'previewImage()'.

```
function previewImage() {  
    var file = document.getElementById('imageInput').files[0];  
    if (file) {  
        var reader = new FileReader();  
        reader.onload = function(event) {  
            var imagenPreview = document.getElementById('objective_image');  
            imagenPreview.src = event.target.result;  
            imagenPreview.style.display = 'block';  
        };  
        reader.readAsDataURL(file);  
    }  
}
```

Fuente: Elaboración propia.

## 5.4.8. Objetivo

### Propósito:

La página de objetivo pretende mostrar la información establecida de un objetivo además de dar la opción al usuario de editar los parámetros o eliminar el objetivo. Editar un objetivo no solo permitirá al usuario corregir posibles errores al introducir los datos, sino que también podrá ir modificando la prioridad de sus objetivos según pase el tiempo para que se asignen en mayor o en menor medida en su planificación y la prioridad de los objetivos se vaya ajustando con el tiempo a la realidad del usuario. Los objetivos se conforman de tareas. Dentro de cada objetivo, un usuario podrá añadir todas las tareas necesarias y acceder a su información.

### Contenido y funcionalidad:

- Título de la página: nombre del objetivo.
- Imagen del objetivo junto a sus parámetros.
- Botón ‘Editar Objetivo’.
- Botón ‘Eliminar Objetivo’.
- Título: ‘Lista de Tareas’.
- Botón: ‘Añadir Tarea’.
- Lista de botones con las tareas correspondientes al objetivo.
- Botón de ayuda ‘i’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

## Imagen:

Ilustración 44. Página de objetivo.

The screenshot shows a mobile application interface. At the top, there is a green header bar with several icons: a circle with a dot, a calendar, a graph, a car, and a person. Below the header, the title 'Aprender inglés' is displayed in a teal-colored font. To the left of the title is a small thumbnail image of a desk with a laptop, a notebook labeled 'Learn English', and some stationery. To the right of the title are three pieces of information: 'Prioridad: 85', 'Horas semanales: 12', and 'Periodo: 01/01/2025 - 31/12/2025'. Below this section are two buttons: 'Editar Objetivo' (in green) and 'Eliminar Objetivo' (in black). The main content area is titled 'Lista de Tareas' and contains four blue rectangular boxes, each listing a task: 'Hacer examen de prueba escrito', 'Hacer examen de prueba oral', 'Hacer examen de prueba de comprensión lectora', and 'Hacer examen de prueba de comprensión auditiva'. Above this list is a button labeled '+ Añadir Tarea' in orange. At the bottom of the screen is a dark footer navigation bar with four sections: 'Páginas' (with links to 'Objetivos', 'Agenda', 'Progreso', 'Notificaciones', and 'Perfil'), 'Contáctanos' (with email 'infoagenda.ai@gmail.com' and phone '+34 666 666 666'), 'Información' (with links to 'Ayuda', 'Aviso Legal', and 'Política de Privacidad'), and 'Redes sociales' (with links to Facebook, Instagram, X, TikTok, and YouTube).

Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/objective.html

Estilo CSS: static/styles/objective.css

JavaScript: static/js/objective.js

En esta página se muestran todas las tareas añadidas al objetivo del color seleccionado al crear el objetivo. Para ello, se comprueba primeramente que existen tareas añadidas al objetivo. En caso de no haber ninguna tarea añadida, se muestra el mensaje 'Añade tareas para completar tu objetivo'. De lo contrario, se iteran todas las tareas asociadas al objetivo y para cada una, se crea un botón del color del objetivo ('objective.color') y se rellena con el nombre de la tarea ('task.name') alojado en el contenedor 'text-container'.

Ilustración 45. Implementación de la lista de tareas en ‘objective.html’.

```
<h3>Lista de Tareas</h3>
<button id="btnAddTask" data-objective-id="{{ objective.id }}">
    <i class="fas fa-plus" aria-label="Añadir una nueva tarea al objetivo"></i>Añadir Tarea
</button>

{% if tasks %}
    <ul>
        {% for task in tasks %}
            <button class="btnTask" data-id="{{ task.id }}" style="background-color: {{ objective.color }};>
                <li>
                    <div class="text-container">
                        <p id="task_name">{{ task.name }}</p>
                    </div>
                </li>
            </button>
        {% endfor %}
    </ul>
{% else %}
    <p id="no_tasks">Añade tareas para completar tu objetivo</p>
{% endif %}
```

Fuente: Elaboración propia.

Las tareas en la lista de tareas se mostrarán en orden de prioridad (de mayor a menor). Esto se determina en la función ‘view\_objective(objective\_id)’ donde, una vez se obtiene el objetivo de la base de datos encontrado a través de su identificador proporcionado como parámetro de la función, se ordenan todas sus tareas según la prioridad de mayor a menor (‘reverse = True’) y se almacena en la variable ‘tasks\_sorted’.

Ilustración 46. Implementación de ‘view\_objective(objective\_id)’

```
@app.route('/objective<int:objective_id>')
def view_objective(objective_id):
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

    objective = Objective.query.get(objective_id)
    if objective:
        tasks_sorted = sorted(objective.tasks, key=lambda task: int(task.priority), reverse=True)
        return render_template('objective.html', objective=objective, tasks=tasks_sorted, active_page='objective')
    else:
        return render_template("error.html", error="Objetivo no encontrado")
```

Fuente: Elaboración propia.

Para editar un objetivo, al hacer clic sobre el botón ‘Editar Objetivo’, se abre un cuadro modal con el formulario con los parámetros de un objetivo. Los valores en los campos serán los valores actuales de esos parámetros del objetivo. En este cuadro modal será donde un usuario deba seleccionar la casilla de verificación para marcar o desmarcar un objetivo como completado.

Ilustración 47. Implementación del cuadro modal para editar un objetivo en 'objective.html'.

```
<div id="editObjectiveModal" class="modal">
  <div class="modal-content">
    <h2>Editar Objetivo</h2>
    <form id="editObjectiveForm" method="post" action="/edit_objective/{{ objective.id }}" onsubmit="return validateForm(this)">
      <label for="edit_name">Nombre del Objetivo:</label>
      <input type="text" id="edit_name" name="name" value="{{ objective.name }}" required><br><br>

      <label for="edit_priority">Prioridad:</label>
      <input type="range" id="edit_priority" name="priority" min="0" max="100" value="{{ objective.priority | default(50) }}" oninput="updatePriorityValue(this.value)">
      <span id="priorityValue">{{ objective.priority | default(50) }}</span><br><br>

      <label for="edit_hours">Horas Semanales:</label>
      <input type="number" id="edit_hours" name="hours" value="{{ objective.hours }}" min="1" required><br>

      <label for="edit_start_date">Fecha de Inicio:</label>
      <input type="date" id="edit_start_date" name="start_date" value="{{ objective.start_date }}" required><br>

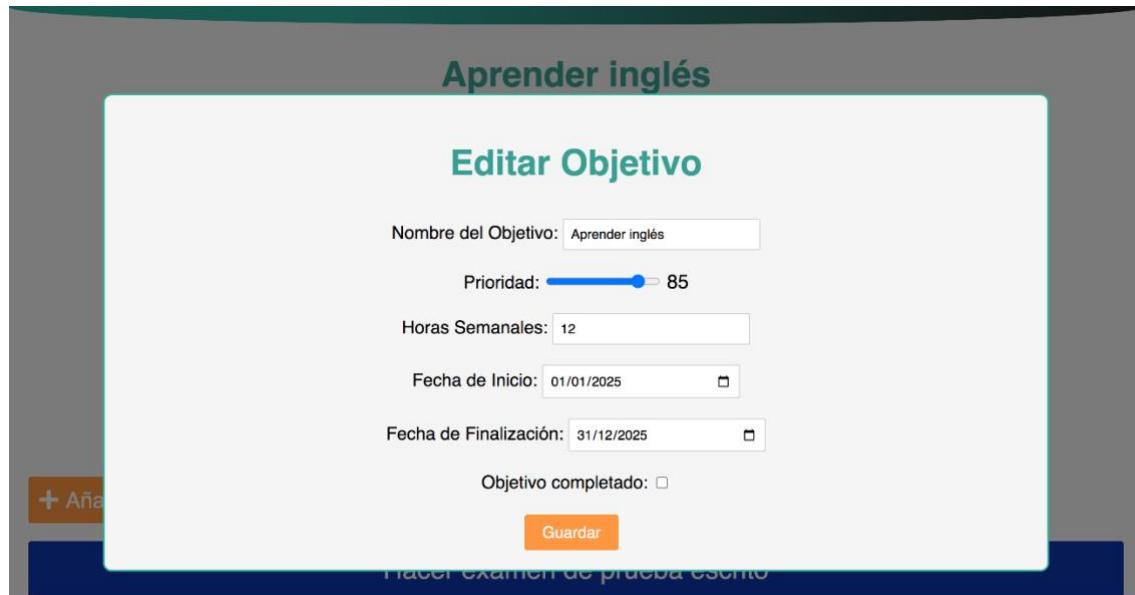
      <label for="edit_end_date">Fecha de Finalización:</label>
      <input type="date" id="edit_end_date" name="end_date" value="{{ objective.end_date }}" required><br>

      <label for="edit_completed">Objetivo completado:</label>
      <input type="checkbox" id="edit_completed" name="completed" {{ objective.completed ? checked }}>

      <button id="saveEdit" class="modal-button">Guardar</button>
    </form>
  </div>
</div>
```

Fuente: Elaboración propia.

Ilustración 48. Cuadro modal para editar un objetivo.



Fuente: Elaboración propia.

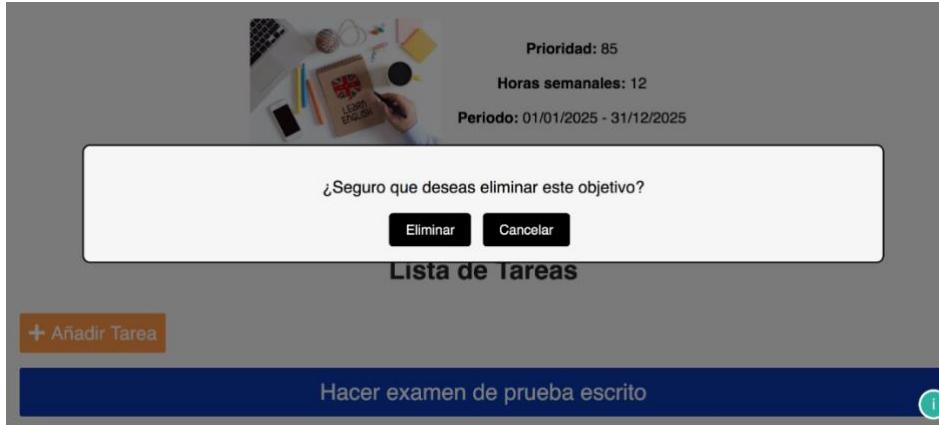
Para eliminar un objetivo, al hacer clic sobre el botón 'Eliminar Objetivo', se abre un cuadro modal para confirmar la eliminación del objetivo ya que una vez eliminado, un usuario no podrá recuperar esa información.

Ilustración 49. Implementación del cuadro modal para eliminar un objetivo en ‘objective.html’.

```
<div id="deleteObjectiveModal" class="modal">
  <div class="modal-content">
    <p>¿Seguro que deseas eliminar este objetivo?</p>
    <button id="confirmDelete" class="modal-button">Eliminar</button>
    <button id="cancelDelete" class="modal-button">Cancelar</button>
  </div>
</div>
```

Fuente: Elaboración propia.

Ilustración 50. Cuadro modal para eliminar un objetivo.



Fuente: Elaboración propia.

## 5.4.9. Nueva Tarea

### Propósito:

La página de nueva tarea contiene el formulario con los campos para la creación de una tarea. Está compuesto por los campos básicos obligatorios como el nombre, la prioridad respecto al resto de tareas del objetivo, las horas mínimas y máximas de dedicación diaria y los días de la semana en los que sería posible realizar la tarea y su horario. Además, se podrá añadir opcionalmente la ubicación en la que se desempeña la tarea.

### Contenido y funcionalidad:

- Header.
- Título de la página: ‘Nueva Tarea’.
- Campo obligatorio ‘Nombre de la tarea’.
- Campo obligatorio ‘Prioridad’.
- Campo obligatorio ‘Horas mínimas diarias’.
- Campo obligatorio ‘Horas máximas diarias’.
- Campo ‘Ubicación’.
- Campos obligatorios ‘Día de la semana’ y ‘Horario’.
- Botón ‘Añadir Día’.
- Botón ‘Quitar Día’.
- Botón ‘Añadir Tarea’.
- Botón de ayuda ‘i’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

## Imagen:

Ilustración 51. Página de nueva tarea.

The screenshot shows a web-based task management application. At the top, there is a dark green header bar with several icons: a target, a calendar, a chart, a car, and a user profile. Below the header, the title "Nueva Tarea" is displayed in a large, bold, teal font. The main form area has a light gray background. It contains the following fields:

- Nombre de la tarea:** A text input field.
- Prioridad:** A horizontal slider with a value of 50.
- Horas mínimas diarias:** A text input field containing the value 1.
- Horas máximas diarias:** A text input field containing the value 1.
- Ubicación:** A text input field with placeholder text "Introduce una ubicación".
- Día de la semana:** A dropdown menu set to "Lunes".
- Horario:** A time picker input field showing "---:---" followed by a minus sign and another time picker showing "---:---".
- Añadir Día** and **Quitar Día** buttons.
- Añadir Tarea** button, which is orange and prominent at the bottom of the form.



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/task\_form.html

Estilo CSS: static/styles/task\_form.css

JavaScript: static/js/task\_form.js

Para la validación de las horas introducidas en los campos de horas mínimas, horas máximas y horario, se ha desarrollado la función 'validateHours()'. En esta función, primeramente, se garantiza que el valor introducido como horas máximas sea superior al de horas mínimas en el momento en el que el usuario envíe el formulario para añadir la tarea al hacer clic en el botón 'Añadir Tarea'.

En caso contrario, se proyectará una alerta indicando al usuario el error y el formulario no se enviará. Además, esta función se cerciora de que los campos del horario son completados para cada día que se haya añadido.

Ilustración 52. Implementación de 'validateHours()'.

```
function validateHours() {  
    var min_hours = parseInt(document.getElementById('min_hours').value);  
    var max_hours = parseInt(document.getElementById('max_hours').value);  
  
    if (min_hours > max_hours) {  
        alert("Las horas mínimas no pueden ser mayores que las horas máximas.");  
        return false;  
    }  
  
    var start_times = document.querySelectorAll('input[name="start_time[]"]');  
    var end_times = document.querySelectorAll('input[name="end_time[]"]');  
  
    for (var i = 0; i < start_times.length; i++) {  
        if (!start_times[i].value || !end_times[i].value) {  
            alert("Por favor, introduzca las horas de inicio y finalización para cada día.");  
            return false;  
        }  
    }  
  
    return true;  
}
```

Fuente: Elaboración propia.

El botón ‘Añadir Día’ añadirá debajo del último campo de día de la semana un nuevo campo de ‘Día de la semana’ y ‘Horario’. Un usuario podrá añadir todos los días que desee. Para realizar esta acción se invocará al método addSchedule().

Ilustración 53. Implementación de 'addSchedule()'.

```
function addSchedule() {  
    var newSchedule = document.querySelector('.schedule').cloneNode(true);  
    document.getElementById('schedules').appendChild(newSchedule);  
}
```

Fuente: Elaboración propia.

Ilustración 54. Formulario de tarea tras añadir días.



The screenshot shows a user interface for managing daily tasks. It features three identical input groups for scheduling:

- Día de la semana: Lunes Horario: 08:30 - 12:00
- Día de la semana: Lunes Horario: 17:30 - 21:20
- Día de la semana: Viernes Horario: 16:00 - 20:00

Below these entries are two buttons: "Añadir Día" (Add Day) in a green box and "Quitar Día" (Remove Day) in a red box. At the bottom center is a large orange button labeled "Añadir Tarea" (Add Task).

Fuente: Elaboración propia.

Por su parte, para quitar días, el método 'removeSchedule()' eliminará el último día añadido excepto el primero ya que debe haber al menos un día de la semana en el que se pueda realizar la tarea. En caso de que un usuario trate de quitar el primer día, se mostrará la alerta con el mensaje informativo de que debe haber al menos un día de realización de la tarea.

Ilustración 55. Implementación de 'removeSchedule()'.

```
function removeSchedule() {  
    var schedules = document.getElementById('schedules');  
    if (schedules.children.length > 1) {  
        schedules.removeChild(schedules.lastChild);  
    } else {  
        alert("Debe haber al menos un día de realización.");  
    }  
}
```

Fuente: Elaboración propia.

Ilustración 56. Alerta de que debe haber al menos un día de la semana asociado a una tarea.



Fuente: Elaboración propia.

## 5.4.10. Tarea

### Propósito:

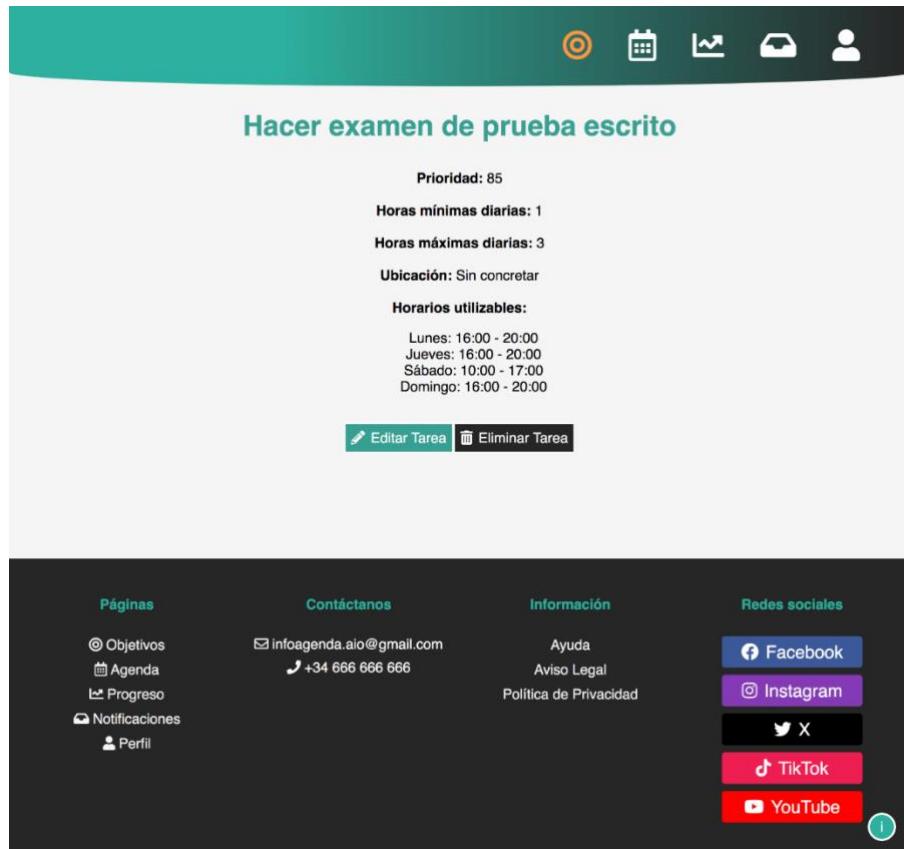
La página de tarea pretende mostrar la información establecida de una tarea además de dar la opción al usuario de editar los parámetros o eliminar la tarea. Editar una tarea no solo permitirá al usuario corregir posibles errores al introducir los datos, sino que también podrá ir modificando la prioridad de cada tarea según pase el tiempo para que se asignen en mayor o en menor medida en su planificación y la prioridad de las tareas se vaya ajustando con el tiempo a la realidad del usuario. En caso de considerar una tarea como completada y/o no querer que se asigne más en la planificación, un usuario podrá eliminarla.

### Contenido y funcionalidad:

- Título de la página: nombre de la tarea.
- Parámetros de la tarea.
- Botón 'Editar Tarea'.
- Botón 'Eliminar Tarea'.
- Botón de ayuda 'i' que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

## Imagen:

Ilustración 57. Página de tarea.



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/task.html

Estilo CSS: static/styles/task.css

JavaScript: static/js/task.js

En app.py se encuentra la función 'show\_task(task\_id)' que basado en el identificador de la tarea pasado por parámetro, carga la página de la tarea con toda su información. Para obtener toda esa información, se realiza una consulta SQL en la que se recibe toda la información de la tabla 'Task' en la base de datos de ese identificador de tarea. A la página HTML, se le pasa tarea y sus horarios.

Ilustración 58. Implementación de 'show\_task(task\_id)'.

```
@app.route('/task/<int:task_id>')
def show_task(task_id):
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

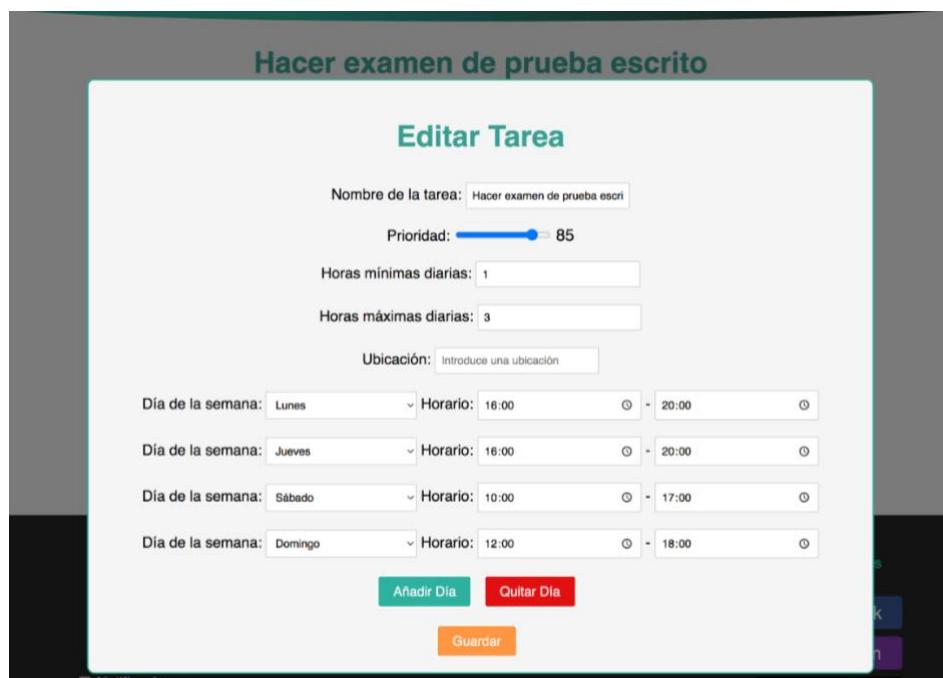
    task = Task.query.get(task_id)

    if task:
        task_schedules = task.schedules
        return render_template('task.html', task=task, task_schedules=task_schedules, active_page='objectives')
    else:
```

Fuente: Elaboración propia.

Al igual que en la página de objetivo, un usuario podrá editar a través de un cuadro modal. En este caso, aparecerán todos los parámetros de la tarea con los campos rellenos por sus valores actuales y un usuario podrá añadir y quitar días de la misma manera y con la condición de que al menos debe haber un día de la semana como se implementó en el formulario para crear una nueva tarea.

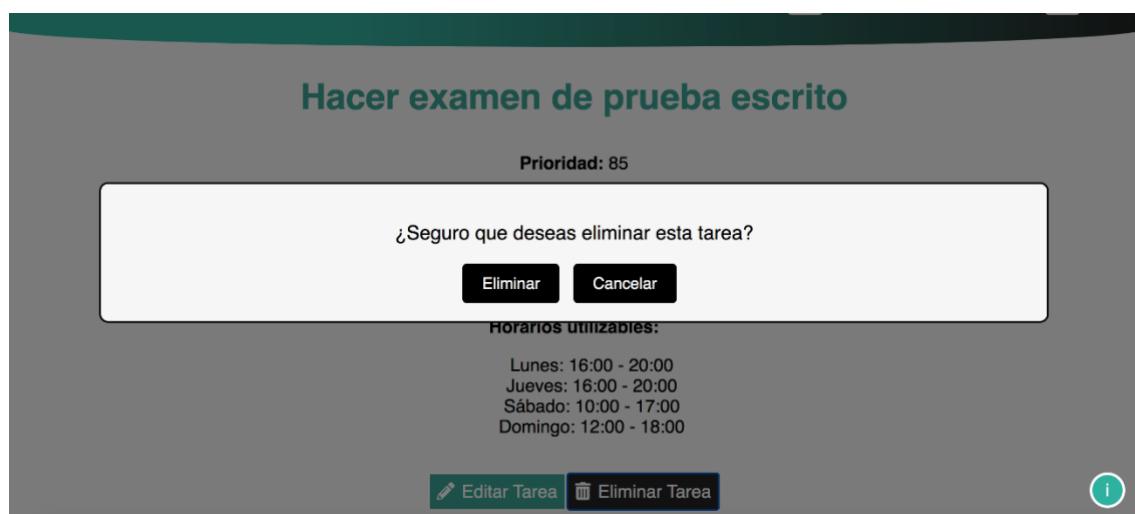
Ilustración 59. Cuadro modal para editar una tarea.



Fuente: Elaboración propia.

Eliminar una tarea también se deberá confirmar con un cuadro modal implementado de la misma manera que la eliminación de un objetivo.

Ilustración 60. Cuadro modal para eliminar una tarea.



Fuente: Elaboración propia.

## 5.4.11. Agenda

### Propósito:

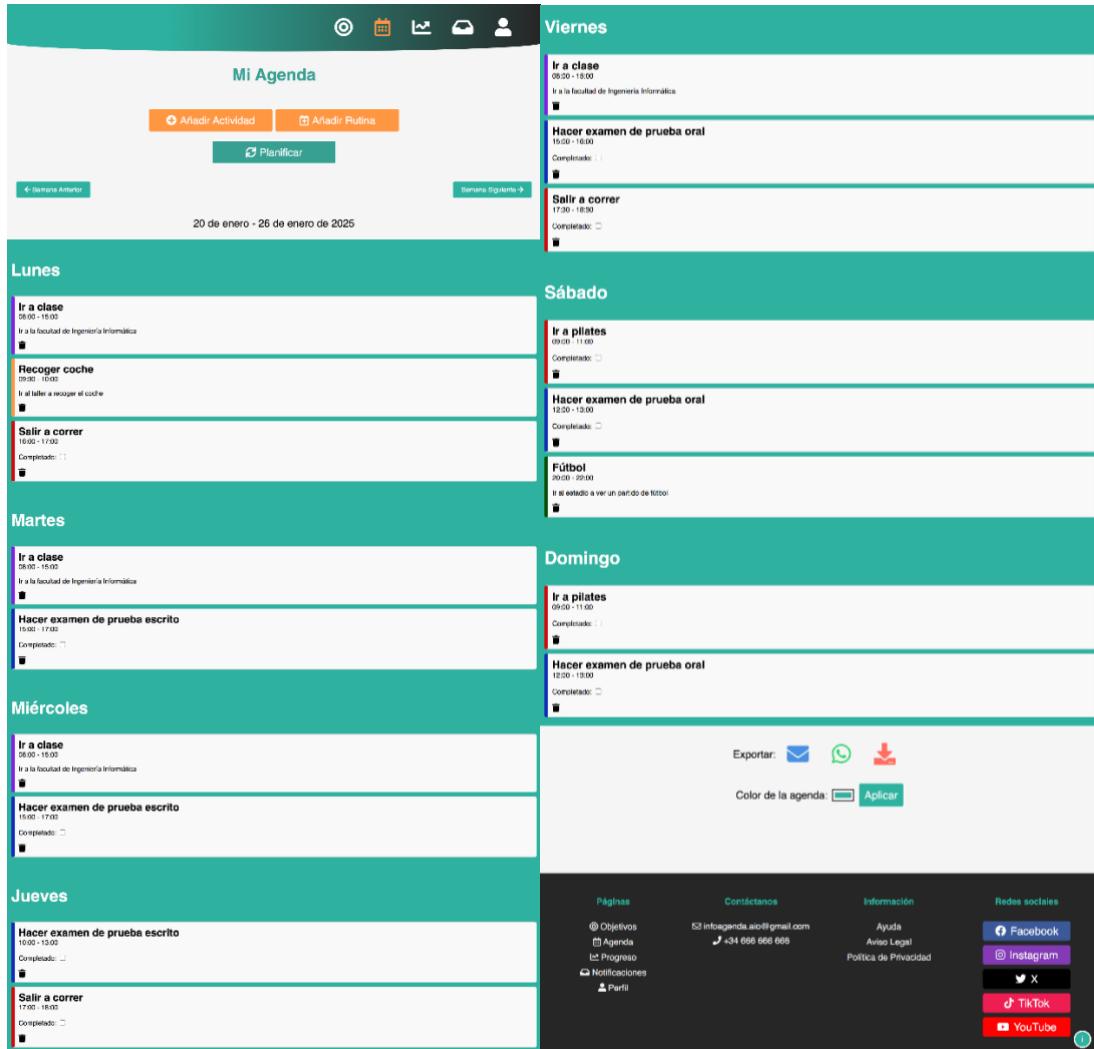
La agenda mostrará la planificación semanal de un usuario. En ella, además, se podrán añadir las actividades y rutinas. Haciendo clic en el botón para planificar, se generará la asignación de tareas y se mostrarán en la agenda junto a las actividades y rutinas que correspondan.

### Contenido y funcionalidad:

- Header.
- Título de la página: ‘Mi Agenda’.
- Botón ‘Añadir Actividad’ para abrir el cuadro modal con el formulario para crear una actividad.
- Botón ‘Añadir Rutina’ para abrir el cuadro modal con el formulario para crear una rutina.
- Botón ‘Semana anterior’ para cargar la planificación semanal de la semana previa a la mostrada en el momento.
- Botón ‘Semana siguiente’ para cargar la planificación semanal de la semana próxima a la mostrada en el momento.
- Texto informativo de la semana que se está proyectando en la agenda.
- Agenda con los eventos para cada día de la semana.
- Botones para la exportación de la planificación de esa semana (no funcional en esta fase del proyecto).
- Cuadro de color junto al botón ‘Aplicar’ para aplicar el color seleccionado al fondo de la agenda.
- Botón de ayuda ‘i’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

## Imagen:

Ilustración 61. Página de agenda.



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/agenda.html

Estilo CSS: static/styles/agenda.css

JavaScript: static/js/agenda.js

Para cargar la página de la agenda, en app.py se encuentra la función 'agenda(offset=0)'. El parámetro offset también ubicado en la ruta para cargar la correspondiente página de la agenda contiene un valor 'offset'. Este número se empleará para cargar la correspondiente semana de la agenda siendo la semana actual la 0. Basado en este valor, se encuentra el día en el que comienza la semana y el día en el que acaba.

Ilustración 62. Implementación de 'agenda(offset=0)'.

```
@app.route('/agenda/<int:offset>', methods=['GET'])
def agenda(offset=0):
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return redirect(url_for('login'))

    today = datetime.now().date()
    start_of_week = today + timedelta(days=offset * 7 - today.weekday())
    end_of_week = start_of_week + timedelta(days=6)
```

Fuente: Elaboración propia.

En el HTML de la página se establece que, en caso de hacer clic en botón para navegar a la página anterior, se disminuya el offset en 1 y en caso de navegar a la página siguiente, éste incremente en 1. Eso cargará la página de la agenda mostrando los detalles de la semana que corresponda.

Ilustración 63. Implementación de botones de navegación entre semanas.

```
<div class="navigation">
    <a href="{{ url_for('agenda', offset=offset-1) }}" class="btnWeek">
        <i class="fas fa-arrow-left" aria-label="Cargar agenda de la semana anterior"></i> Semana Anterior</a>
    <a href="{{ url_for('agenda', offset=offset+1) }}" class="btnWeek">
        <i class="fas fa-arrow-right" aria-label="Cargar agenda de la semana siguiente">Semana Siguiente </i></a>
</div>
```

Fuente: Elaboración propia.

Para proyectar los eventos en la agenda, se extraen las actividades, rutinas y tareas asignadas de la base de datos que tengan una fecha comprendida entre la fecha de inicio y fin de semana.

Ilustración 64. Extracción de tareas asignadas

```
tasks = AllocatedTask.query.filter(
    AllocatedTask.date.between(start_of_week, end_of_week)
).all()

weekday_to_enum = {
    0: DayOfWeek.LUNES,
    1: DayOfWeek.MARTES,
    2: DayOfWeek.MIERCOLES,
    3: DayOfWeek.JUEVES,
    4: DayOfWeek.VIERNES,
    5: DayOfWeek.SABADO,
    6: DayOfWeek.DOMINGO
}
locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')

week_data = {day: [] for day in DayOfWeek}
```

Fuente: Elaboración propia.

Los parámetros para cada evento se almacenan en la variable 'week\_data' donde después, se ordenan en orden de hora de inicio para ya tenerlos en orden cronológico ascendente.

Ilustración 65. Asignación de tareas a 'week\_data'.

```
for task in tasks:
    objective_color = task.task.objective.color if task.task.objective else None
    day_enum = weekday_to_enum[task.date.weekday()]
    week_data[day_enum].append({
        'id': task.id,
        'type': 'task',
        'name': task.name,
        'start_time': task.start_time.strftime('%H:%M'),
        'end_time': task.end_time.strftime('%H:%M'),
        'completed': task.completed,
        'color': objective_color
    })

    for day, events in week_data.items():
        events.sort(key=lambda x: x['start_time'])

return render_template('agenda.html', week_data=week_data, offset=offset, start_date=start_of_week.strftime('%Y-%m-%d'))
```

Fuente: Elaboración propia.

Por último, la función 'update\_agenda\_color()' se emplea para que al aplicar un cambio de color a la agenda, ésta se adapte instantáneamente. Para ello, se modifica el valor 'agenda\_color' en la tabla 'users' en la base de datos donde el identificador sea el del usuario con la sesión abierta.

Ilustración 66. Implementación de 'update\_agenda\_color()'

```
@app.route('/update_agenda_color', methods=['POST'])
def update_agenda_color():
    user_id = session.get('user_id')
    user = User.query.filter_by(id=user_id).first()
    new_color = request.json.get('color')
    if new_color:
        user.agenda_color = new_color
        db.session.commit()
        return jsonify({'message': 'Color de agenda actualizado correctamente'})
    else:
        return jsonify({'error': 'No se proporcionó un nuevo color para la agenda'})
```

Fuente: Elaboración propia.

## 5.4.12. Progreso

### Propósito:

La agenda mostrará la planificación semanal de un usuario. En ella, además, se podrán añadir las actividades y rutinas. Haciendo clic en el botón para planificar, se generará la asignación de tareas y se mostrarán en la agenda junto a las actividades y rutinas que correspondan.

### Contenido y funcionalidad:

- Header.
- Título de la página: 'Mi Progreso'.
- Texto informativo del número de total de objetivos creados.
- Texto informativo del número total de objetivos completados.
- Gráfico representando el porcentaje de objetivos completados.

- Lista de objetivos con información del objetivo y de las veces que se ha completado cada tarea del objetivo y el número de horas que se le ha dedicado a la tarea.
- Botón de ayuda ‘i’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

### Imagen:

Ilustración 67. Página de progreso.

**Mi Progreso**

Total de Objetivos: 3  
Total de Objetivos Completados: 1  
% de Objetivos Completados: 33.33%

**Hacer deporte**  
01/01/2025 - 31/12/2026  
Objetivo en curso  
**Salir a correr**  
Tareas completadas: 3  
Horas totales dedicadas: 3  
**Ir a pilates**  
Tareas completadas: 2  
Horas totales dedicadas: 4

**Aprender inglés**  
01/01/2025 - 31/03/2025  
Objetivo en curso  
**Hacer examen de prueba escrito**  
Tareas completadas: 2  
Horas totales dedicadas: 4  
**Hacer examen de prueba oral**  
Tareas completadas: 3  
Horas totales dedicadas: 3  
**Hacer examen de prueba de comprensión lectora**  
Tareas completadas: 0  
Horas totales dedicadas: 0  
**Hacer examen de prueba de comprensión auditiva**  
Tareas completadas: 0  
Horas totales dedicadas: 0

**Leer libros**  
01/08/2024 - 31/12/2024  
Objetivo cumplido  
**Leer Harry Potter**  
Tareas completadas: 0  
Horas totales dedicadas: 0  
**Leer Juego de Tronos**  
Tareas completadas: 0  
Horas totales dedicadas: 0

Páginas: [Objetivos](#), [Agenda](#), [Progreso](#), [Notificaciones](#), [Perfil](#)

Contáctanos: [infagenda.aio@gmail.com](mailto:infagenda.aio@gmail.com), [+34 666 666 666](tel:+34666666666)

Información: [Ayuda](#), [Aviso Legal](#), [Política de Privacidad](#)

Redes sociales: [Facebook](#), [Instagram](#), [X](#), [TikTok](#), [YouTube](#)

Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/progress.html

Estilo CSS: static/styles/progress.css

JavaScript: static/js/progress.js

Para la página de progreso existe la función 'progress()' en app.py. En esta función se extraen de la base de datos los objetivos, el recuento de tareas asignadas completadas y la suma total de horas dedicadas con cada tarea. Así, se establecen los datos que hay que proyectar en esta página. A cada objetivo, se le atribuye un estado basado en la fecha actual indicando si es un objetivo ya cumplido, los días que restan para comenzarlo, los días que han pasado desde que acabó el plazo para cumplirlo o si el objetivo ya ha sido marcado como completado. A partir de ahí, se calculan los valores para hacer el resumen general de objetivos. Para ello, se encuentra el número total de objetivos, el número total de objetivos completados y, usando esos dos valores, el porcentaje de objetivos completados.

Ilustración 68. Implementación de 'progress()'.

```
task_count_map = {task_id: count for task_id, count in allocated_task_counts}
task_hours_map = {task_id: hours or 0 for task_id, hours in
                  allocated_task_hours}

objectives_status = {}
today = datetime.now().date()
for objective in objectives:
    if objective.completed:
        status = "Objetivo cumplido"
    elif objective.start_date > today:
        days_remaining = (objective.start_date - today).days
        status = f"Días restantes para iniciar el objetivo: {days_remaining}"
    elif objective.end_date < today:
        days_expired = (today - objective.end_date).days
        status = f"El objetivo expiró hace {days_expired} días"
    else:
        status = "Objetivo en curso"

    objectives_status[objective.id] = status

total_objectives = len(objectives)
completed_objectives = sum(1 for obj in objectives if obj.completed)
completed_percentage = (completed_objectives / total_objectives * 100) if total_objectives > 0 else 0
```

Fuente: Elaboración propia.

Para proyectar la información previamente calculada, en el contenedor 'objectives-progress-container' se muestran los valores del total de objetivos y del total de objetivos completados. Después, para mostrar el porcentaje, se ha optado por adoptar un estilo más visual en vez de mostrar únicamente el valor. Para ello, se ha aplicado un SVG (Scalable Vector Graphics) que permite con facilidad generar gráficos vectoriales en dos dimensiones. En él, se mostrará un círculo de 120x120 píxeles (no muy grande para que no sea tan invasivo en la página) con un círculo de color gris cuya circunferencia se irá rellenando de color verde en el porcentaje marcado en el valor mostrado en el centro del círculo. Este porcentaje se redondeará a dos decimales para evitar valores demasiado largos.

Ilustración 69. Implementación del resumen de objetivos..

```
<div id="objectives-progress-container">
    <div id="totals-containern">
        <span>Total de Objetivos: <strong>{{ total_objectives }}</strong></span><br>
        <span>Total de Objetivos Completados: <strong>{{ completed_objectives }}</strong></span><br>
    </div>
    <div id="percentage-container">
        <span>% de Objetivos Completados:</span>
        <svg width="120" height="120" viewBox="0 0 120 120">
            <circle cx="60" cy="60" r="50" stroke="#ddd" stroke-width="10" fill="none" />
            <circle cx="60" cy="60" r="50" stroke="#4caf50" stroke-width="10" fill="none"
                stroke-dasharray="314" stroke-dashoffset="{{ 314 - (314 * completed_percentage / 100) }}"
                style="transition: stroke-dashoffset 0.5s ease;" />
            <text x="60" y="65" text-anchor="middle" font-size="20" fill="#000">
                {{ completed_percentage|round(2) }}%
            </text>
        </svg>
    </div>
</div>
```

Fuente: Elaboración propia.

En el HTML también, se proyectarán todos los objetivos en forma de columna. Cada objetivo mostrará su nombre, imagen de objetivo, periodo para completarlo, estado actual y la lista de tareas. Cada tarea identificada por su ‘task.id’ proyectará el número de tareas completadas almacenado en la variable ‘task\_map\_count’ y el número de horas totales dedicadas en ‘task\_hours\_map’.

Ilustración 70. Estructuración del progreso de los objetivos.

```
{% for objective in objectives %}
    <div id="objective-container" style="border: 5px solid {{ objective.color }}">
        <div id="objective-info">
            <h3>{{ objective.name }}</h3>
            <p>{{ objective.start_date.strftime('%d/%m/%Y') }} - {{ objective.end_date.strftime('%d/%m/%Y') }}
            <p id="status">{{ objectives_status[objective.id] }}</p>
            <ul>
                {% for task in objective.tasks %}
                    <li id="tasks">
                        <strong id="task_name">{{ task.name }}</strong>
                        <p>Tareas completadas: {{ task_count_map.get(task.id, 0) }}<br>
                            Horas totales dedicadas: {{ task_hours_map.get(task.id, 0) }}</p><br>
                    </li>
                {% endfor %}
            </ul>
        </div>
        <div id="objective-image">
            {% if objective.image %}
                
            {% else %}
                
            {% endif %}
        </div>
    </div>
```

Fuente: Elaboración propia.

## 5.4.13. Notificaciones

### Propósito:

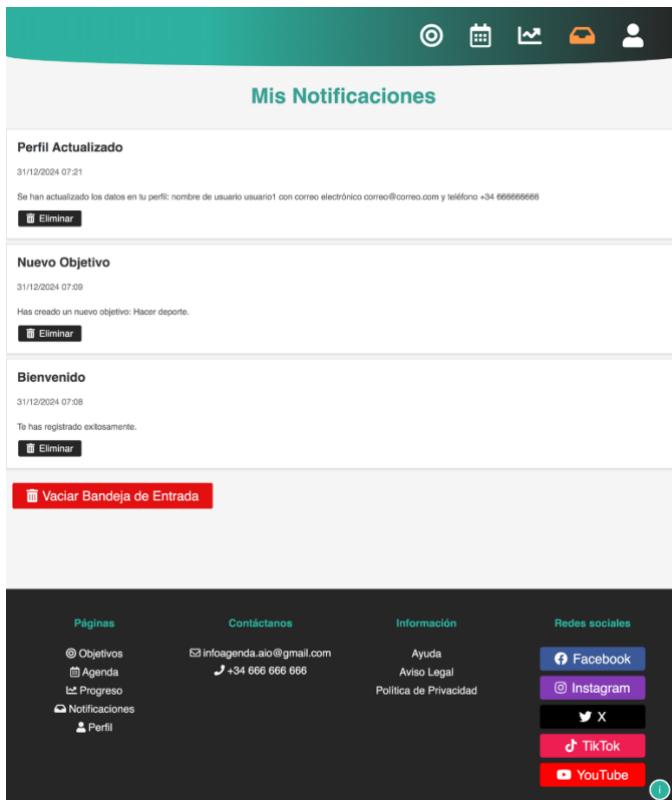
La página de notificaciones mostrará una bandeja de entrada con alertas y mensajes para el usuario como un mensaje de bienvenida al registrarse, la notificación de que se ha creado un nuevo objetivo o el aviso de que los datos del perfil han sido actualizados.

### Contenido y funcionalidad:

- Header.
- Título de la página: ‘Mis Notificaciones’.
- Carrusel de notificaciones. Cada notificación contará con:
  - Título.
  - Fecha y hora de creación.
  - Descripción.
  - Botón de eliminación que desplegará un cuadro modal con la confirmación de la eliminación.
- Botón ‘Vaciar Bandeja de Entrada’ que desplegará un cuadro modal para la confirmación de la eliminación de todas las notificaciones.
- Botón de ayuda ‘í’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

### Imagen:

Ilustración 71. Página de notificaciones.



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/inbox.html

Estilo CSS: static/styles/inbox.css

JavaScript: static/js/inbox.js

Para mostrar todas las notificaciones de un usuario, éstas se cargan desde la base de datos en la función 'inbox()' en app.py. Para cargar todas las notificaciones, se realiza una consulta SQL ordenada por fecha y hora para que aparezcan en orden cronológico en la bandeja de entrada.

*Ilustración 72. Implementación de 'inbox()'.*

```
@app.route('/inbox')
def inbox():
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('Login.html')

    notifications = Notification.query.filter_by(user_id=user_id).order_by(Notification.date.desc(),
                                                                           Notification.time.desc()).all()
    return render_template('inbox.html', notifications=notifications, active_page='inbox')
```

Fuente: Elaboración propia.

Para la creación de notificaciones, se ha desarrollado la función 'create\_notification(user\_id, title, description)' en app.py que toma como parámetros el identificador del usuario al que le corresponde la notificación, el título de la notificación y la descripción con los detalles de la misma. A partir de ahí, empleando la librería datetime, se identifica la fecha y la hora actual para almacenar la notificación en la base de datos con los parámetros de día, hora, título, descripción e identificador de usuario.

*Ilustración 73. Implementación de 'create\_notification(user\_id, title, description)'.*

```
def create_notification(user_id, title, description):
    now = datetime.now()
    new_notification = Notification(
        date=now.date(),
        time=now.time(),
        title=title,
        description=description,
        user_id=user_id
    )
    db.session.add(new_notification)
    db.session.commit()
```

Fuente: Elaboración propia.

Empleando esta función, se crearán las notificaciones de manera sencilla cuando sea oportuno.

*Ilustración 74. Creación de una notificación al crear un objetivo.*

```
new_objective = Objective(name=name, image=filename, color=color, priority=priority, hours=hours,
                           start_date=start_date, end_date=end_date, user_id=user_id)
db.session.add(new_objective)
db.session.commit()

create_notification(user_id, "Nuevo Objetivo", f"Has creado un nuevo objetivo: {name}.")
```

Fuente: Elaboración propia.

Para eliminar una notificación de la base de datos de forma permanente, el usuario deberá hacer clic sobre el botón negro ‘Eliminar’ y confirmarlo en el cuadro modal que desbloqueará a continuación. Al aceptar la eliminación de la notificación, ésta se localizará mediante una consulta SQL a través de su identificador único y se eliminará.

Ilustración 75. Implementación de ‘delete\_notification(notification\_id)’.

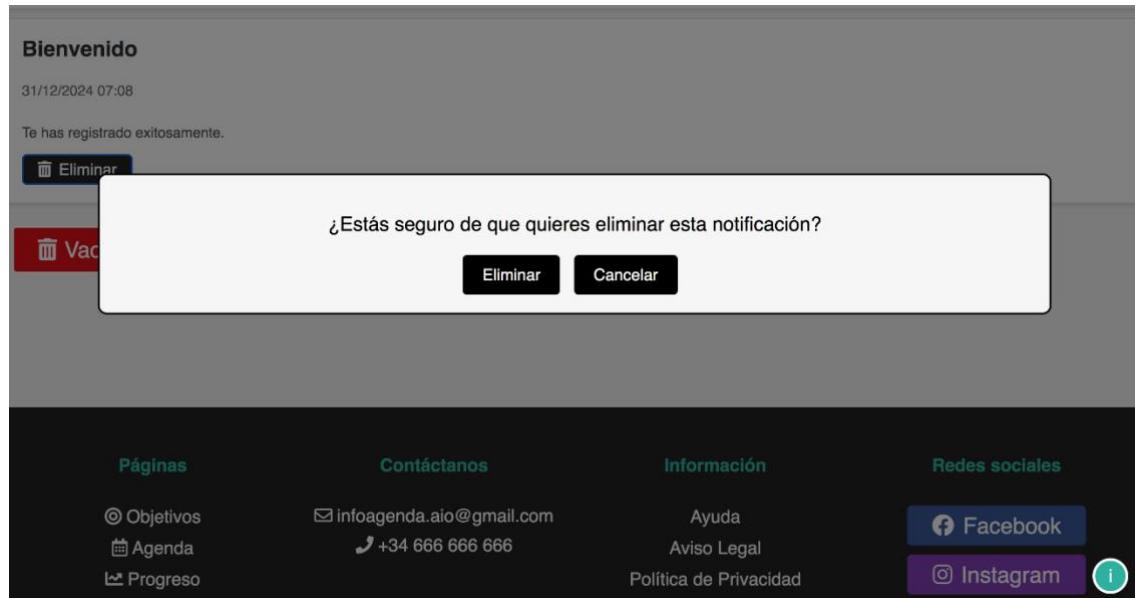
```
@app.route('/inbox/<int:notification_id>/delete_notification', methods=['GET', 'POST'])
def delete_notification(notification_id):
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

    notification = Notification.query.filter_by(id=notification_id).first()
    if notification:
        Notification.query.filter_by(id=notification.id).delete()
        db.session.delete(notification)
        db.session.commit()
    return redirect('/inbox')
```

Fuente: Elaboración propia.

Para confirmar que el usuario quiere eliminar la notificación, tras hacer clic en el botón ‘Eliminar’, aparecerá en pantalla un cuadro modal para confirmar la acción.

Ilustración 76. Cuadro modal para eliminar una notificación.



Fuente: Elaboración propia.

En el caso de vaciar la bandeja de entrada y eliminar permanentemente todas las notificaciones, esto se hará a través del identificador del usuario y se eliminarán todas las notificaciones vinculadas al usuario.

Ilustración 77. Implementación de 'delete\_all\_notifications()'.

```
@app.route('/inbox/delete_all_notifications', methods=['POST'])
def delete_all_notifications():
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

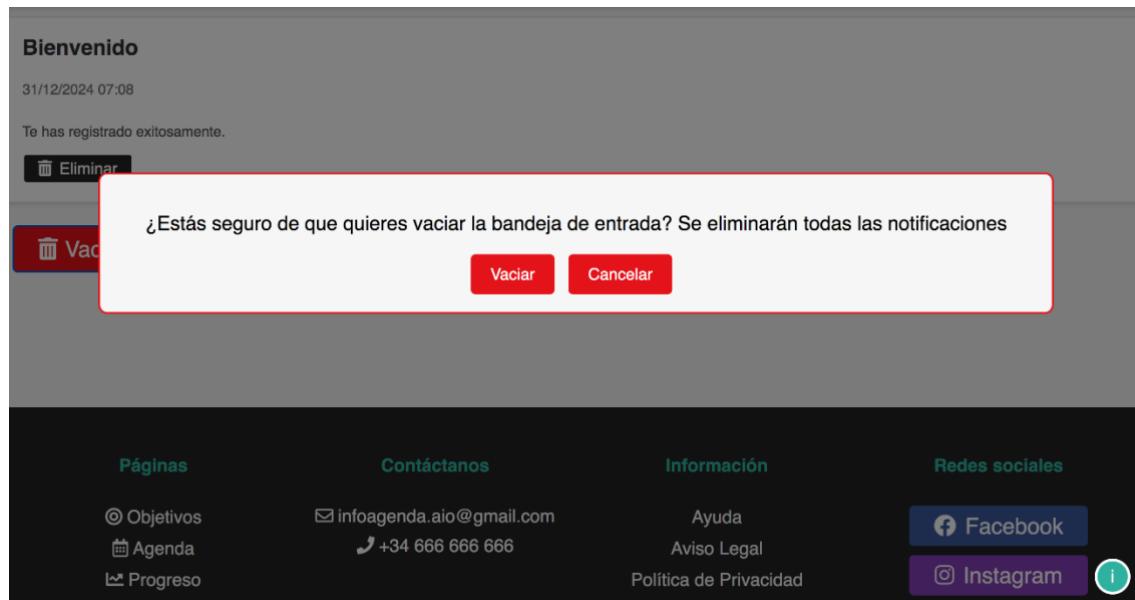
    Notification.query.filter_by(user_id=user_id).delete()
    db.session.commit()

    return redirect('/inbox')
```

Fuente: Elaboración propia.

Para confirmar que el usuario quiere vaciar la bandeja de entrada, tras hacer clic en el botón ‘Vaciar Bandeja de Entrada’, aparecerá en pantalla un cuadro modal para confirmar la acción.

Ilustración 78. Cuadro modal para vaciar la bandeja de entrada.



Fuente: Elaboración propia.

#### 5.4.14. Perfil

##### Propósito:

La página de perfil permitirá al usuario visualizar y editar los datos de su perfil, cerrar la sesión actual y eliminar la cuenta.

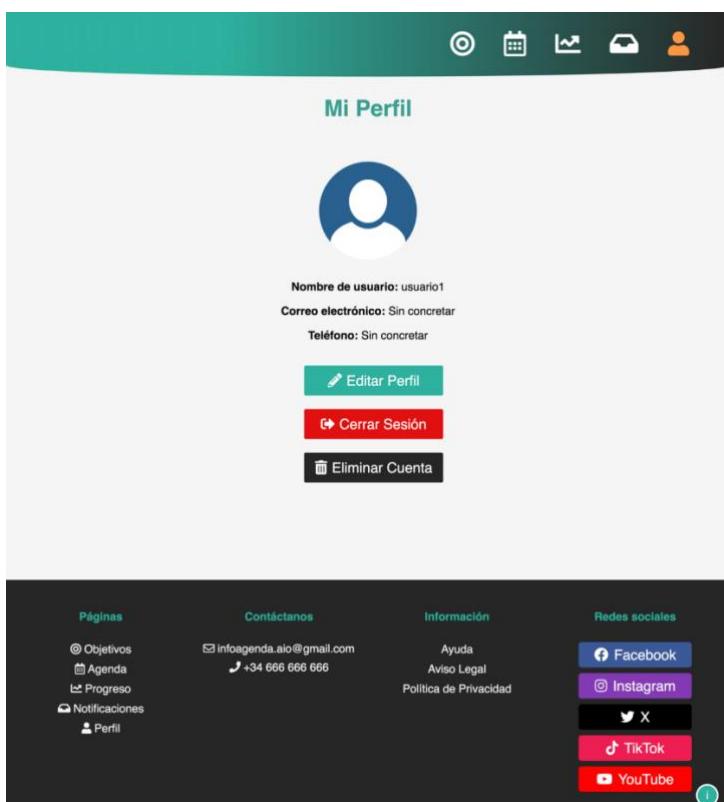
##### Contenido y funcionalidad:

- Header.
- Título de la página: ‘Mi Perfil’.
- Botón ‘Editar Perfil’ que desplegará un cuadro modal en el que el usuario podrá modificar sus datos actuales.

- Botón ‘Cerrar Sesión’ que desglosará un cuadro modal para la confirmación del usuario para cerrar la sesión actual. Se dirigirá al portal de usuario.
- Botón ‘Eliminar Cuenta’ que eliminará de la base de datos todos los datos del usuario en la aplicación. Se dirigirá al portal de usuario.
- Botón de ayuda ‘í’ que desplegará un cuadro modal con información de ayuda al usuario en esa página.
- Footer.

### Imagen:

Ilustración 79. Página de perfil.



Fuente: Elaboración propia.

### Desarrollo técnico:

Plantilla HTML: templates/profile.html

Estilo CSS: static/styles/profile.css

JavaScript: static/js/profile.js

Para validar que el número de teléfono introducido cumple con los criterios establecidos a la hora de editar el perfil, la función ‘validatePhone(phone)’ comentada en la sección de registro se aplica de igual manera en ‘profile.js’. También, se comprueba al igual que en el formulario de registro en la función ‘edit\_profile()’ en app.py si el nuevo nombre de usuario, correo electrónico y teléfono ya están en uso por otro usuario ya que éstos deben ser únicos. En caso de haber un error por este motivo y el perfil no pueda ser editado, se mostrará un mensaje en rojo indicando el motivo. En cambio, si la edición del perfil

fructifica, se mostrará un mensaje en color verde confirmando la correcta actualización del perfil y se generará una notificación indicando los nuevos datos.

Ilustración 80. Implementación del mensaje de error o éxito al editar el perfil.

```
existing_phone = User.query.filter(User.phone_country_code == phone_country_code,
                                    User.phone_number == phone_number,
                                    User.id != user_id).first()

if existing_phone:
    error_message = "No se ha podido actualizar el perfil. El número de teléfono ya está en uso."
    return render_template('profile.html', user=user, active_page='profile', error_message=error_message)

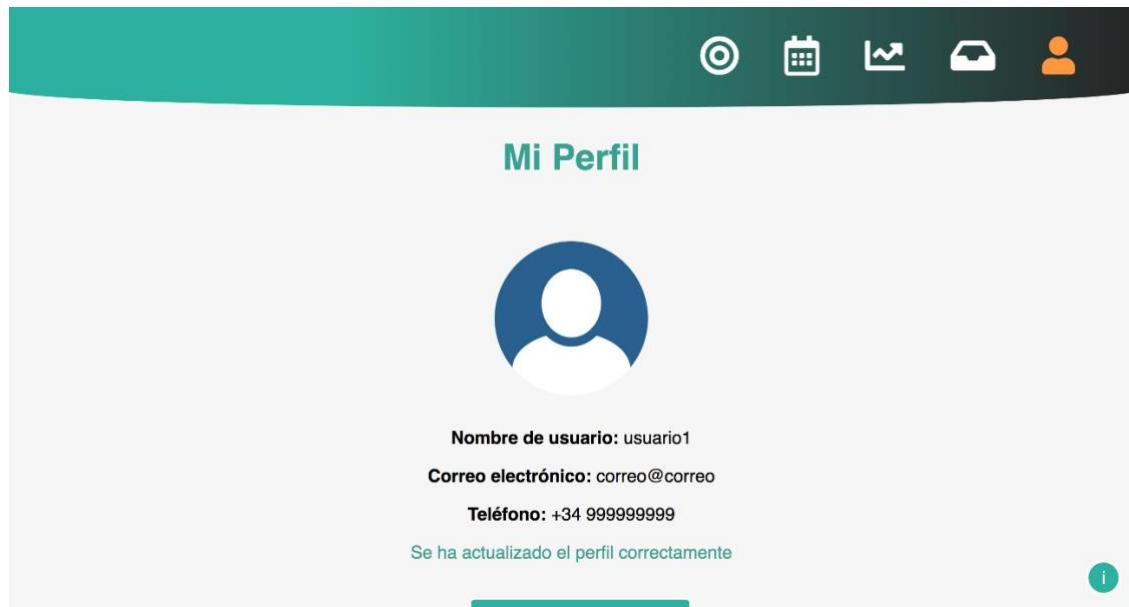
user.username = username
user.email = email
user.phone_country_code = phone_country_code
user.phone_number = phone_number

success_message = "Se ha actualizado el perfil correctamente"
db.session.commit()

create_notification(user.id, "Perfil Actualizado",
                    '''Se han actualizado los datos en tu perfil:
                    nombre de usuario {0} con
                    correo electrónico {1} y
                    teléfono {2} {3}'''.format(username, email, phone_country_code, phone_number))
return render_template('profile.html', user=user, active_page='profile', success_message=success_message)
```

Fuente: Elaboración propia.

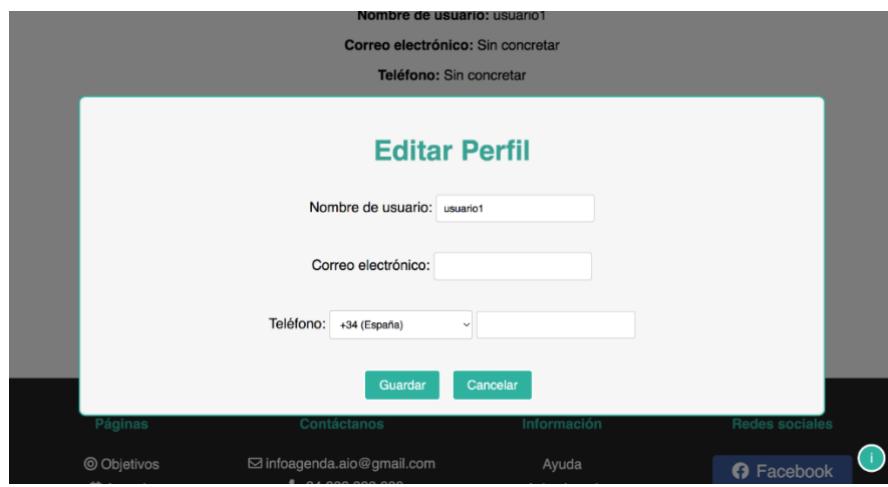
Ilustración 81. Mensaje de éxito al editar el perfil.



Fuente: Elaboración propia.

Para editar el perfil, tras hacer clic en el botón ‘Editar Perfil’, aparecerá en pantalla un cuadro modal con el formulario pertinente en el cual se encuentran los campos que el usuario debe modificar.

Ilustración 82. Cuadro modal para editar el perfil.



Fuente: Elaboración propia.

En la página de perfil, un usuario también será capaz de cerrar su sesión actualmente activa. Para ello, la función 'logout()' utilizará el método de Flask 'session.pop()' que eliminará la clave de la sesión iniciada (identificador del usuario) y redirigirá al usuario al Portal de Usuario.

Ilustración 83. Implementación de 'logout()'.

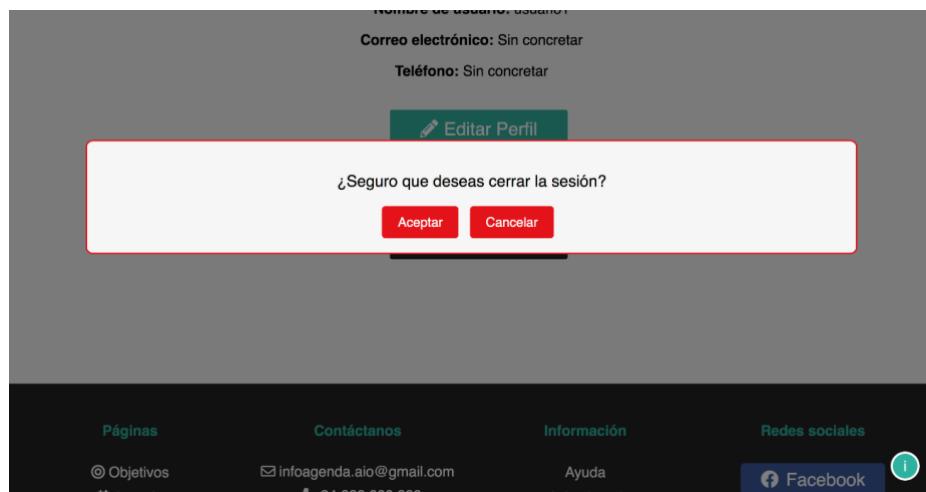
```
@app.route('/logout')
def logout():
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

    session.pop('user_id', None)
    return redirect('/')
```

Fuente: Elaboración propia.

Para confirmar que el usuario quiere cerrar la sesión, tras hacer clic en el botón 'Cerrar Sesión', aparecerá en pantalla un cuadro modal para confirmar la acción.

Ilustración 84. Cuadro modal para cerrar sesión.



Fuente: Elaboración propia.

Además, un usuario podrá eliminar su cuenta de forma permanente. Esto borrará todo el rastro del usuario de la base de datos. Para ello, se ha implementado la función ‘delete\_account()’ que basado en el identificador del usuario busca y elimina todas las actividades, rutinas, horarios, objetivos y tareas asociados a ese usuario.

Ilustración 85. Implementación de ‘delete\_account()’.

```
@app.route('/delete_account', methods=['GET', 'POST'])
def delete_account():
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return render_template('login.html')

    if user_id:
        user = User.query.get(user_id)
        if user:
            Activity.query.filter_by(user_id=user_id).delete()
            for routine in user.routines:
                for schedule in routine.routine_schedules:
                    db.session.delete(schedule)
                db.session.delete(routine)
            for objective in user.objectives:
                for task in objective.tasks:
                    db.session.delete(task)

            Objective.query.filter_by(user_id=user_id).delete()
            db.session.delete(user)
            db.session.commit()

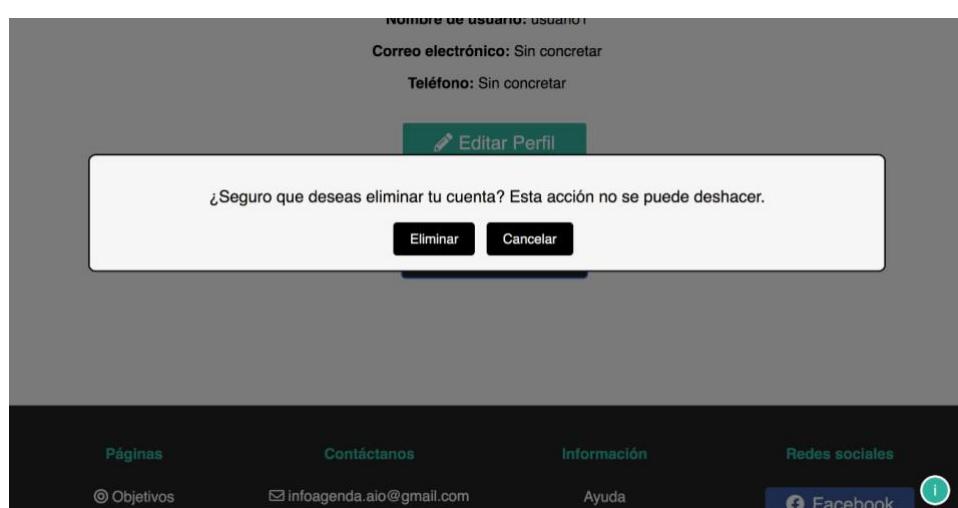
    session.pop('user_id', None)

    return redirect('/')
```

Fuente: Elaboración propia.

Para confirmar que el usuario quiere eliminar su cuenta, tras hacer clic en el botón ‘Eliminar Cuenta’, aparecerá en pantalla un cuadro modal para confirmar la acción.

Ilustración 86. Cuadro modal para eliminar una cuenta.



Fuente: Elaboración propia.

## 5.4.15. Ayuda

## **Propósito:**

La página de ayuda tiene como objetivo informar al usuario de forma clara y detallada todo lo necesario para comprender e interactuar con la aplicación correctamente.

## **Contenido y funcionalidad:**

- Título de la página: 'Ayuda'.
  - Información para el usuario de las diferentes secciones del sitio web.
  - Footer.

## Imagen:

*Ilustración 87. Página de ayuda.*

Ayuda

## Registrarse

Requiere de forma totalmente gratuita en la aplicación web. Para ello tan solo tendrás que rellenar los siguientes campos obligatorios del formulario:

### Nombre de usuario

Tú mismo o el nombre que te lo identificara en la aplicación.

### Contraseña

Contraseña para acceder a tu cuenta.

### Confirmar contraseña

Vuelve a introducir la contraseña con el fin de asegurar que no hay ningún error.

### La contraseña debe cumplir las siguientes condiciones:

La contraseña debe contener al menos una letra mayúscula.

La contraseña debe contener al menos un número.

La contraseña debe contener al menos un carácter especial.

La contraseña debe tener entre 8 y 16 caracteres (8&gt;16</8>).  
7</div>

La contraseña no puede contener el nombre de usuario.

Además, puedes añadir los siguientes campos opcionales:

### Foto de perfil

Imagen que personalizarás tu perfil.

### Correos electrónicos

Diseñar de correo electrónico para recuperar la contraseña o exportar tu agenda.

### Teléfono

Número de teléfono para exportar tu agenda a través de WhatsApp.

El correo electrónico y el teléfono los podrás añadir editando el perfil más adelante. También podrás activarla la notificación de mensajes.

Se deberá marcar los cuadros de verificación aceptando la Política de Privacidad.

## Iniciar Sesión

Si ya te has registrado, accede a tu cuenta introduciendo tu nombre de usuario o correo electrónico además de la contraseña.

## Objetivos

Establece tus objetivos. A cada objetivo, añadele tareas para que se apliquen de manera automática e inteligente a tu agenda. Puedes crear un número ilimitado de objetivos.

### OBJETIVOS

Los objetivos son las metas que tienes en tu vida. Crea un número infinito de objetivos y a cada uno, añadele tareas para que se plasmiten en tu agenda.

Los objetivos se dividirán en 4 categorías:

#### Objetivos Actuales

Objetivos que has completado y están actualmente en curso.

#### Objetivos Comprometidos

Objetivos que han sido marcados como completados. No importa si están en curso o no. Las tareas de estos objetivos aparecerán en tu agenda.

#### Objetivos Pasados

Objetivos que estén marcados como completados y su fecha de finalización ya ha pasado. No se mantendrá en tu calendario.

#### Objetivos Futuros

Objetivos que han sido marcados como completados y su fecha de inicio aún no ha llegado. Se emplearán a mostrar en la planificación una vez llegue su fecha de inicio.

Los campos obligatorios a rellenar para CREAR UN OBJETIVO serán:

### Nombre del objetivo

Título o descripción del objetivo.

### Color

Por defecto, será de color verde. Estilo el color a cada objetivo para diferenciarlo en la pantalla de objetivos y en tu agenda.

### Prioridad

Selecciona un valor de prioridad entre 0 y 10. Basado en los valores de prioridad de tus objetivos, el algoritmo asignará tu agenda de la forma más óptima.

### Horas semanales

El número de horas semanales que se estima o se desea dedicar al objetivo ayudará a determinar la distribución de las tareas que se crean para este objetivo.

### Las fechas

Las fechas obligatorias son:

#### Imagen del objetivo

Añage a un objetivo una imagen que se pague de objetivos sea lo más personalizada posible.

#### Fecha de inicio

Establece la fecha en la que quieres empezar a realizar tareas para completar un objetivo.

#### Fecha de finalización

Indica la fecha límite que tienes para completar el objetivo propuesto.

### TAREAS

Una vez creado el objetivo, podrás añadir tareas para cumplir el objetivo.

Para ello, pulsa sobre el objetivo, selecciona las tareas que se crean una tras otra.

El orden en que tú objectives las tareas es importante para que se cumplan en la mejor forma y en el mejor momento.

Para que puedas tener más información, crea la tarea dentro del objetivo para que se pueda presentar en la agenda.

Los campos obligatorios a rellenar para CREAR UNA TAREA serán:

### Nombre de la tarea

Título o descripción de la tarea.

### Prioridad

Selecciona un valor de prioridad entre 0 y 10. Basado en los valores de prioridad de las demás tareas de tu objetivo, el algoritmo asignará tu agenda a tu agenda de la forma más óptima.

### Horas mínimas diarias

Indicar el número mínimo de horas que le podrás dedicar a una tarea en un día.

### Día de la semana

Selecciona los días de la semana y la rango de horas del día en los que podrás realizar la tarea.

El único campo que no es obligatorio es:

### Ubicación

Lugar en el que deseas la tarea.

## EDITAR Y ELIMINAR

Se podrán editar los parámetros y eliminar de forma permanente, excepto si debes hacerlo en el cuadro de edición del objetivo.

Para poder editar el objetivo y eliminar de forma permanente, es necesario que se encuentre en el cuadro de edición del objetivo.

## Agenda

### Consultar la agenda

Revisa de forma totalmente gratuita en la aplicación web. Para ello tan solo tendrás que rellenar los siguientes campos obligatorios del formulario:

### Nombre de usuario

Tu nombre o el nombre que te lo identificara en la aplicación.

### Contraseña

Contraseña para acceder a tu cuenta.

### Confirmar contraseña

Vuelve a introducir la contraseña con el fin de asegurar que no hay ningún error.

### La contraseña debe cumplir las siguientes condiciones:

La contraseña debe contener al menos una letra mayúscula.

La contraseña debe contener al menos un número.

La contraseña debe contener al menos un carácter especial.

La contraseña no puede contener el nombre de usuario.

Además, puedes añadir los siguientes campos opcionales:

### Foto de perfil

Imagen que personalizarás tu perfil.

### Correos electrónicos

Diseñar de correo electrónico para recuperar la contraseña o exportar tu agenda.

### Teléfono

Número de teléfono para exportar tu agenda a través de WhatsApp.

El correo electrónico y el teléfono los podrás añadir editando el perfil más adelante. También podrás activarla la notificación de mensajes.

Se deberá marcar los cuadros de verificación aceptando la Política de Privacidad.

### ACTIVIDAD

## Una ACTIVIDAD será aquél evento que se realizará una única vez.

Por ejemplo, 'Recoger el coche del taller'.

Todos los campos para ANADIR ACTIVIDAD serán obligatorios:

### Nombre de la actividad

Título de la actividad.

### Fecha

Día en el que se va a realizar la actividad.

### Hora de inicio

Hora en la que comienza la actividad.

### Hora de finalización

Hora de conclusión de la actividad.

### Color

Color con el que se representará la actividad en la agenda. Por defecto, será de color naranja.

El campo opcional que no es obligatorio completa:

### Descripción

Descripción o información adicional de la actividad.

## RUTINA

## Una RUTINA será aquél evento que se repetirá periódicamente ciertos días de la semana en una franja horaria.

Por ejemplo, 'Ir a trabajar'.

Todos los campos para ANADIR RUTINA serán obligatorios:

### Nombre de la rutina

Título de la rutina.

### Día de la semana y Horario

Días en los que se va a realizar el evento y su franja horaria. Se podrán añadir y quitar todos los días necesarios.

### Color

Color con el que se representará la rutina en la agenda. Por defecto, será de color azul marino.

El campo opcional que no es obligatorio completa:

### Descripción

Descripción o información adicional de la rutina.

## REPLANIFICAR

El botón replanificar hará la asignación de tareas en los días restantes de la semana (hasta el domingo) y la semana siguiente. En caso de ya existir una asignación de tareas, éstas se sobreescrivirán.

## ELIMINAR EVENTOS

Una vez añadido a la agenda un evento o una rutina, éste se podrán editar para eliminar.

Para eliminar una actividad o rutina, se deberá hacer clic en el icono del calendario del evento en la agenda. No se podrá eliminar un único evento de una rutina sino la rutina completa.

## COLOR DE LA AGENDA

Se podrá actualizar el color de la agenda en cualquier momento. Por defecto, será de color verde.

## PROGRESO

Monitorea el progreso de cada uno de tus objetivos. Puedes consultar cuántas tareas has completado, los días que te quedan para cumplir tu objetivo y el grado de satisfacción de tu objetivo.

## Notificaciones

Recibe todas las alertas y notificaciones en tu bandeja de entrada para que estés al tanto en todo momento.

Puedes eliminar una o todas las notificaciones y vaciar la bandeja de entrada.

## Perfil

Visualiza y edita tus datos personales dentro de la aplicación. Además, podrás cerrar sesión o eliminar tu cuenta de forma permanente.

Página	Contactame	Información	Redes sociales
Objetivo Agenda Tarea Progreso Notificaciones Perfil	<ul> <li>Objetivo</li> <li>Agenda</li> <li>Tarea</li> <li>Progreso</li> <li>Notificaciones</li> <li>Perfil</li>	Ayuda Aviso Aviso Legal Política de Privacidad	Facebook Instagram Twitter LinkedIn YouTube

*Fuente: Elaboración propia.*

## Desarrollo técnico:

Plantilla HTML: templates/help.html  
Estilo CSS: static/styles/help.css

Esta página se carga mediante la función ‘help()’ cuando en la URL se selecciona la página ‘/help’. Esta función en app.py simplemente carga el archivo ‘help.html’.

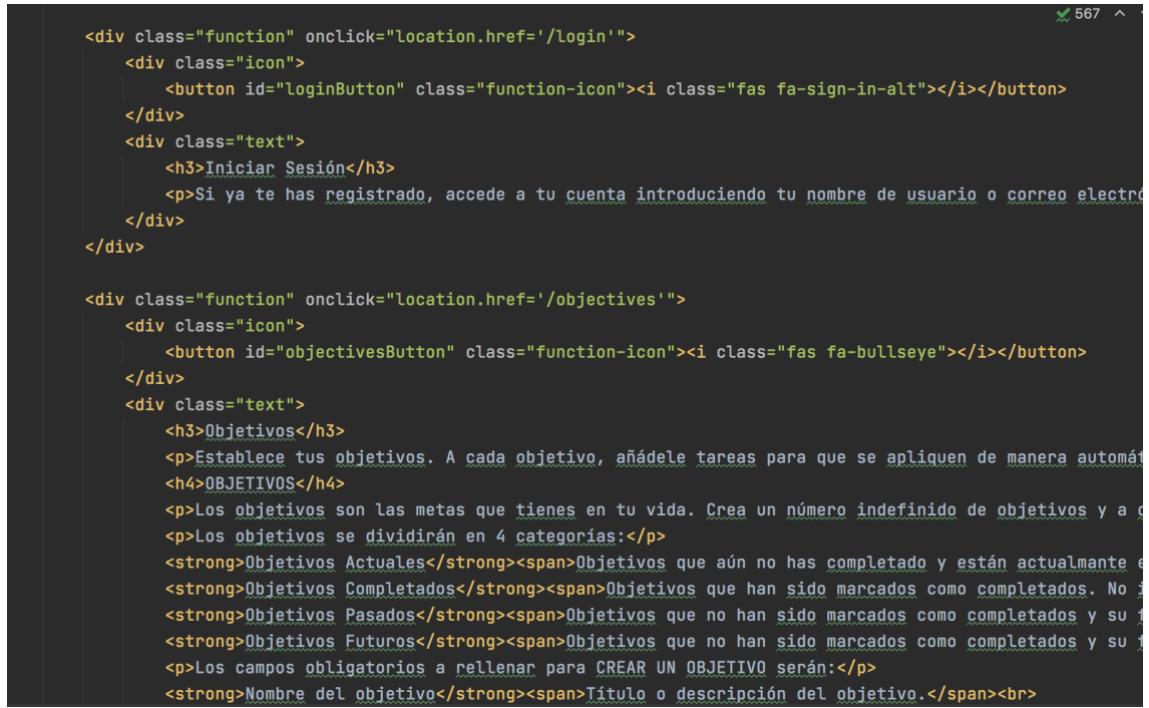
*Ilustración 88. Implementación de help().*

```
@app.route('/help')
def help():
    return render_template('help.html', active_page='help')
```

Fuente: Elaboración propia.

Las diferentes secciones de ayuda están divididas en la clase ‘function’. Cada uno de estos contenedores contienen el ícono, el título y la descripción de la ayuda para cada caso. El contenedor también ejercerá como botón que dirigirá al usuario a la sección correspondiente.

*Ilustración 89. Implementación de ‘help.html’.*



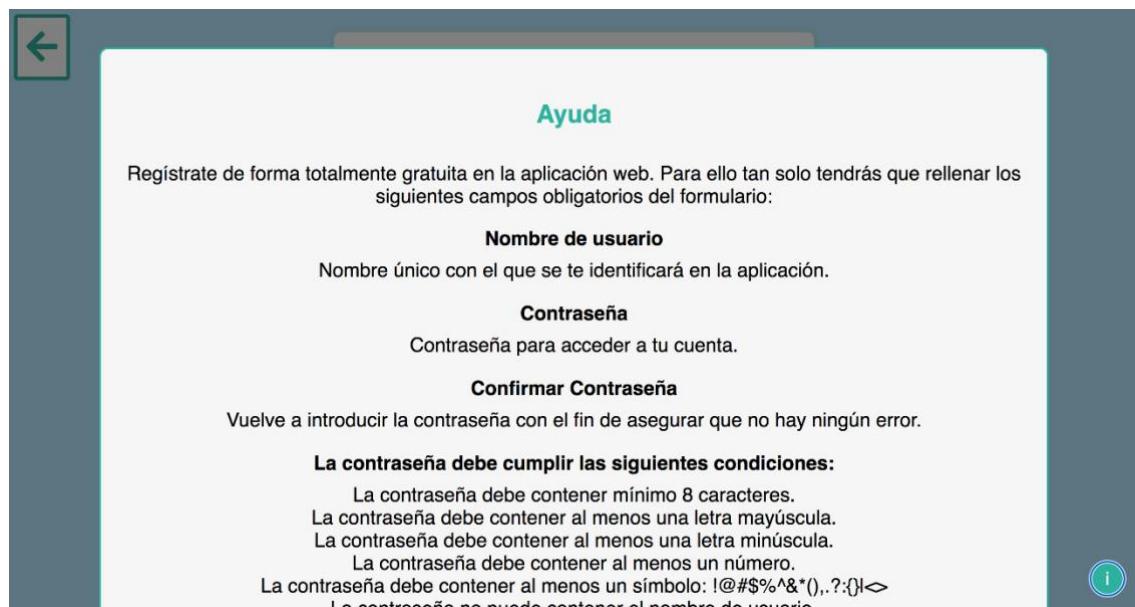
```
<div class="function" onclick="location.href='/Login'>
    <div class="icon">
        <button id="loginButton" class="function-icon"><i class="fas fa-sign-in-alt"></i></button>
    </div>
    <div class="text">
        <h3>Iniciar Sesión</h3>
        <p>Si ya te has registrado, accede a tu cuenta introduciendo tu nombre de usuario o correo electrónico</p>
    </div>
</div>

<div class="function" onclick="location.href='/objectives'">
    <div class="icon">
        <button id="objectivesButton" class="function-icon"><i class="fas fa-bullseye"></i></button>
    </div>
    <div class="text">
        <h3>Objetivos</h3>
        <p>Establece tus objetivos. A cada objetivo, añádele tareas para que se apliquen de manera automática.</p>
        <h4>OBJETIVOS</h4>
        <p>Los objetivos son las metas que tienes en tu vida. Crea un número indefinido de objetivos y a cada uno de ellos, establece tareas para que se apliquen de manera automática.</p>
        <p>Los objetivos se dividirán en 4 categorías:</p>
        <strong>Objetivos Actuales</strong><span>Objetivos que aún no has completado y están actualmente en desarrollo.</span>
        <strong>Objetivos Completados</strong><span>Objetivos que han sido marcados como completados. No necesitas hacer nada más para cumplirlos.</span>
        <strong>Objetivos Pasados</strong><span>Objetivos que no han sido marcados como completados y su cumplimiento ya es responsabilidad del sistema.</span>
        <strong>Objetivos Futuros</strong><span>Objetivos que no han sido marcados como completados y su cumplimiento aún no ha comenzado.</span>
        <p>Los campos obligatorios a llenar para CREAR UN OBJETIVO serán:</p>
        <strong>Nombre del objetivo</strong><span>Titular o descripción del objetivo.</span><br>
    </div>
</div>
```

Fuente: Elaboración propia.

Además, la sección de Ayuda se encontrará en las diferentes secciones del sitio web y los usuarios podrán acceder a través de un botón flotante ubicado en la esquina inferior derecha de la pantalla desde el punto de vista del usuario con forma de círculo y el texto ‘i’ de información. Al hacer clic sobre el botón, se desplegará un cuadro modal con la información de ayuda específica de la página actual.

Ilustración 90. Cuadro modal con información de ayuda.



Fuente: Elaboración propia.

#### 5.4.16. Aviso Legal

##### Propósito:

La página de Aviso Legal pretende informar de forma clara al usuario tanto de la titularidad como del contacto del sitio web. Además, informa de la motivación detrás de la creación de la aplicación como su finalidad. Por último, advierte de las responsabilidades eludidas por los responsables.

##### Contenido y funcionalidad:

- Título de la página: 'Aviso Legal'
- Contenido del Aviso Legal.
- Botón 'Volver' que dirigirá al usuario a la página de inicio de sesión en caso de no haber iniciado sesión o a la página de la agenda en caso de tener la sesión ya iniciada.
- Footer.

## Imagen:

Ilustración 91. Página de Aviso Legal.

**Aviso Legal**

### Titular y contacto

Siguiendo el Artículo 10 de la Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico, el titular de este sitio web es Carlos Octavio Rodríguez del Toro, con N.I.F. 45344594W y correo electrónico carlos.rodriguez153@alu.ulpgc.es.

### Motivación del sitio web

Este sitio web ha sido desarrollado como Trabajo de Fin de Título del Grado de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria de Carlos Octavio Rodríguez del Toro con José Fortes Gálvez como tutor para el curso académico 2024/2025.

### Finalidad del sitio web

Este sitio web trata de ayudar a sus usuarios registrados a marcar sus objetivos y cuadricularlos de la forma más óptima posible en su agenda. Para ello, los usuarios registran las actividades y rutinas que realizan día a día.

### Responsabilidades

El responsable de este sitio, excluye, cualquier responsabilidad por los daños y perjuicios de toda naturaleza causados por la falta de disponibilidad o de continuidad del acceso al sitio web.

El acceso a la información y servicios que presta este sitio web tiene, en principio, una duración indefinida. No obstante, se podrá dar por terminada o suspender el acceso a este sitio web en cualquier momento.

Además, se excluye, cualquier responsabilidad por los daños y perjuicios de toda naturaleza que puedan deberse a la presencia de virus u otros códigos maliciosos en los contenidos que puedan producir cualquier tipo de daños en el sistema informático, documentos electrónicos o ficheros de los usuarios.

No obstante, se declara que se han adoptado las medidas de seguridad dentro de las posibilidades para garantizar el correcto funcionamiento del sitio web y evitar ataques a la ciberseguridad.

Este Aviso Legal se actualizó por última vez el 2 de enero de 2025.

Páginas	Contáctanos	Información	Redes sociales
Objetivos Agenda Progreso Notificaciones Perfil	infoagenda.aio@gmail.com +34 666 666 666	Ayuda Aviso Legal Política de Privacidad	<a href="#">Facebook</a> <a href="#">Instagram</a> <a href="#">X</a> <a href="#">TikTok</a> <a href="#">YouTube</a>

Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/legal.html

Estilo CSS: static/styles/legal.css

Esta página se carga mediante la función 'legal()' cuando en la URL se selecciona la página '/legal'. Esta función en app.py simplemente carga el archivo 'legal.html'.

Ilustración 92. Implementación de 'legal()'.

```
@app.route('/legal')
def legal():
    return render_template('legal.html', active_page='legal')
```

Fuente: Elaboración propia.

Esta página simplemente proyectará el texto describiendo el Aviso Legal del sitio web.

*Ilustración 93. Implementación de 'legal.html'.*

```
<body>
    <h2>Aviso Legal</h2>

    <button class="return-button" onclick="window.location.href='/agenda'">Volver</button>

    <h3>Titular y contacto</h3>
    <p>Siguiendo el Artículo 10 de la Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico, el titular de este sitio web es Carlos Octavio Rodríguez del Toro, con N.I.F. 45344594W y correo electrónico carlos.rodriguez153@alu.ulpgc.es.</p>

    <h3>Motivación del sitio web</h3>
    <p>Este sitio web ha sido desarrollado como Trabajo de Fin de Título del Grado de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria de Carlos Octavio Rodríguez del Toro con José Fortes Gálvez como tutor para el curso académico 2024/2025.</p>

    <h3>Finalidad del sitio web</h3>
    <p>Este sitio web trata de ayudar a sus usuarios registrados a marcar sus objetivos y cuadricularlos de la forma
```

Fuente: Elaboración propia.

#### 5.4.17. Política de Privacidad

##### **Propósito:**

La página de Política de Privacidad trata de informar a los usuarios claramente quién es el titular del sitio web y cómo contactarle, qué datos se recopilan y por qué además de la duración de la conservación de los datos almacenados, los derechos que tiene un usuario por la legislación, las medidas de seguridad adoptadas e informar de la posibilidad de cambios en la política.

##### **Contenido y funcionalidad:**

- Título de la página: ‘Política de Privacidad’.
- Contenido de la Política de Privacidad.
- Botón ‘Volver’ que dirigirá al usuario a la página de inicio de sesión en caso de no haber iniciado sesión o a la página de la agenda en caso de tener la sesión ya iniciada.
- Footer.

## Imagen:

Ilustración 94. Página de Política de Privacidad.

The screenshot shows a privacy policy page with the following sections and content:

- Titular y contacto**: States that following the Organic Law 3/2018, on December 5th, of Protection of Personal Data and guarantee of digital rights, the owner of this website is Carlos Octavio Rodríguez del Toro, with N.I.F. 45344594W and email carlos.rodriguez153@alu.ulpgc.es.
- ¿Qué datos recopilamos?**: Lists non-optional personal data:
  - Tu correo electrónico.
  - Tu número de teléfono.
  - Las imágenes que cargas.
- ¿Para qué recopilamos estos datos?**: Explains data usage:
  - Correo electrónico: used for session initiation if the user has not saved their name, and for recovering their password if they have saved it or want to change it. It can also be used to export their agenda via email or WhatsApp.
  - Images uploaded by users will be used as profile pictures or objectives when creating a goal.
- Conservación de los datos**: States that data provided by users is stored indefinitely in the application's database until the account is deleted. At that point, all user data is permanently erased.
- Derechos**: All users have the following rights:
  - Derecho de acceso.
  - Derecho a la rectificación.
  - Derecho de supresión.
  - Derecho a la limitación del tratamiento.
  - Derecho a la portabilidad.
  - Derecho de oposición.Users can exercise these rights by sending a request to infoagenda.alo@gmail.com.
- Medidas de seguridad**: Notes that security measures are in place to ensure the safety of user data stored in our database.
- Cambios en la Política de Privacidad**: States that the privacy policy may be modified to adapt to changes in legislation or to improve the service. The last update was on January 2, 2025.

Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/privacy\_policy.html

Estilo CSS: static/styles/privacy\_policy.css

Esta página se carga mediante la función 'privacy\_policy()' cuando en la URL se selecciona la página '/privacy\_policy'. Esta función en app.py simplemente carga el archivo 'privacy\_policy.html'.

Ilustración 95. Implementación de 'privacy\_policy()'.

```
@app.route('/privacy_policy')
def privacy_policy():
    return render_template('privacy_policy.html', active_page='privacy-policy')
```

Fuente: Elaboración propia.

Esta página simplemente proyectará el texto describiendo la Política de Privacidad del sitio web.

Ilustración 96. Implementación de 'privacy\_policy.html'.

```
<body>
    <h2>Política de Privacidad</h2>

    <button class="return-button" onclick="window.location.href='/agenda'">Volver</button>

    <h3>Titular y contacto</h3>
    <p>Siguiendo la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, el titular de este sitio web es Carlos Octavio Rodríguez del Toro, con N.I.F. 45344594W y correo electrónico carlos.rodriguez153@alu.ulpgc.es.</p>

    <h3>¿Qué datos recopilamos?</h3>
    <p><br>-Los datos personales que recopilamos <strong>no son obligatorios</strong>. Éstos son:<br>-Tu correo electrónico.<br>-Tu número de teléfono.<br>-Las imágenes que cargas.</p>

    <h3>¿Para qué recopilamos estos datos?</h3>
    <p>Tu correo electrónico lo podrás utilizar para iniciar sesión en caso de haber olvidado</p>
```

Fuente: Elaboración propia.

## 5.4.18. Error

### Propósito:

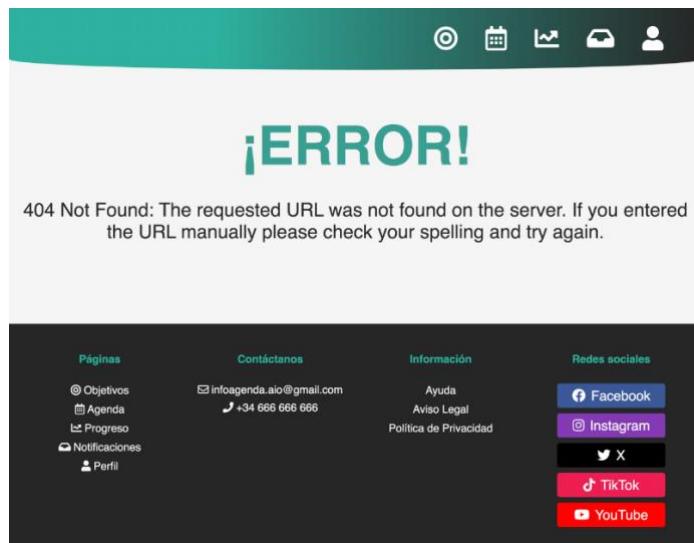
La página de error informará al usuario de forma clara en caso de haber algún tipo de error en el sistema.

### Contenido y funcionalidad:

- Header.
- Título de la página: '¡ERROR!'
- Descripción del error.
- Footer.

### Imagen:

Ilustración 97. Página de error.



Fuente: Elaboración propia.

## Desarrollo técnico:

Plantilla HTML: templates/error.html  
Estilo CSS: static/styles/error.css

La página de error cargará en caso de existir alguna excepción o fallo. Este fallo podrá saltar por cualquier excepción detectada por Flask como páginas no encontradas o fallos en el servidor. Para ello, se ha desarrollado la función ‘handle\_exception(e)’ que, al recibir una excepción, cargará la página ‘error.html’ y le mandará el texto con la especificación del error.

*Ilustración 98. Implementación de ‘handle\_exception(e)’.*

```
@app.errorhandler(Exception)
def handle_exception(e):
    return render_template("error.html", error=e)
```

Fuente: Elaboración propia.

La función ‘error()’ será la encargada de abrir ‘error.html’ en la ruta ‘/error’.

*Ilustración 99. Implementación de ‘error()’.*

```
@app.route('/error')
def error():
    return render_template('error.html', active_page='error')
```

Fuente: Elaboración propia.

El código HTML de la página de error tan solo mostrará un título indicando el error con los detalles del error debajo.

*Ilustración 100. Implementación de ‘error.html’.*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="{{ url_for('static', filename='styles/error.css') }}>
    <title>Error</title>
</head>
<body>
    {% include 'header.html' %}

    <h2>¡ERROR!</h2>
    <p>{{ error }}</p>

    {% include 'footer.html' %}
</body>
</html>
```

Fuente: Elaboración propia.

## 5.5. Planificación

El algoritmo realizará la asignación de tareas para lo que resta de la semana vigente y la semana siguiente al completo. Los motivos principales detrás de esta decisión son dos. Primeramente, no tendría demasiada lógica asignar tareas para días lejanos en el tiempo ya que las agendas son muy dinámicas y la disponibilidad de un usuario variará diaria o semanalmente. En segundo lugar, uno de los requisitos del sistema es tener un tiempo de respuesta rápido. En caso de realizar la asignación para todos los días en un periodo largo de tiempo (por ejemplo, un año), el algoritmo tardaría demasiado tiempo en ejecutarse para asignar todas las tareas y este tiempo podría resultar tiempo perdido ya que desde que un usuario se proponga nuevos objetivos, modifique algún parámetro de un objetivo o una tarea, añada alguna actividad o rutina y deba volver a planificar, se repetirá todo el proceso. Por ello, el periodo de tiempo para asignar tareas es entre el día en el que el usuario manualmente comienza el proceso de planificación hasta el domingo de la semana siguiente. Para encontrar la fecha del domingo de la semana siguiente, se ha elaborado la función ‘calculate\_next\_sunday(date)’ a la cual se le pasa por parámetro la fecha del día actual.

*Ilustración 101. Implementación de ‘calculate\_next\_sunday(date)’.*

```
def calculate_next_sunday(date):
    this_week_sunday = date + timedelta(days=(6 - date.weekday()))
    return this_week_sunday + timedelta(days=7)

start_date = datetime.now().date()
end_date = calculate_next_sunday(start_date)
```

Fuente: Elaboración propia.

El siguiente paso para realizar por el algoritmo es la extracción de las actividades y rutinas ya programadas en esos días del periodo de asignación de tareas. Los espacios de tiempo ocupados por estos eventos no podrán ser utilizados por las tareas a la hora de asignarles un horario. También, se consideran todas las tareas asignadas en la planificación anterior que han sido marcadas como completadas. Esto se debe a que si en el día en el que se vuelve a planificar ya se ha realizado alguna tarea, ésta no se pueda replanificar o planificar alguna otra en ese espacio de tiempo. Esta acción se realiza en la función ‘extract\_activities\_routines\_tasks()’.

Ilustración 102. Implementación de 'extract\_activities\_routines\_tasks()'.

```
def extract_activities_routines_tasks():
    user_id = session.get('user_id')
    activities_array = []
    routines_array = []
    tasks_array = []
    activities = (Activity.query.filter_by(user_id=user_id).filter(Activity.date >= start_date, Activity.date < end_date).all())
    for activity in activities:
        activities_array.append(
            {
                "name": activity.name,
                "date": activity.date,
                "start_time": activity.start_time,
                "end_time": activity.end_time,
                "description": activity.description,
                "color": activity.color,
            }
        )

    routines = Routine.query.filter_by(user_id=user_id).all()
    for routine in routines:
        routines_array.append(
            {
                "name": routine.name,
                "description": routine.description,
                "color": routine.color,
            }
        )
```

Fuente: Elaboración propia.

A partir de ahí, la función ‘order\_busy\_time(activities, routines, tasks, start\_date, end\_date)’ incorpora las fechas y los horarios de las actividades, tareas y rutinas previamente extraídas de la base de datos y las almacena en una única lista en la que se ordenan en orden cronológico.

Ilustración 103. Implementación de 'order\_busy\_time(activities, routines, tasks, start\_date, end\_date)'.

```
def order_busy_time(activities, routines, tasks, start_date, end_date):
    busy_time = []

    for activity in activities:
        busy_time.append([activity['date'], activity['start_time'], activity['end_time']])

    for routine in routines:
        for schedule in routine['schedules']:
            current_date = start_date
            while current_date <= end_date:
                if current_date.weekday() == get_weekday_number(schedule['day_of_week']):
                    busy_time.append([current_date, schedule['start_time'], schedule['end_time']])
                current_date += timedelta(days=1)

    for task in tasks:
        busy_time.append([task['date'], task['start_time'], task['end_time']])

    busy_time = sorted(busy_time, key=lambda x: (x[0], x[1]))

    return busy_time
```

Fuente: Elaboración propia.

Por último, antes de asignar las tareas, se calculan a partir de los horarios ocupados, cuáles son los intervalos de tiempo disponibles para cada día en los que se puede realizar una tarea. Para ello, el método ‘calculate\_free\_time(busy\_time, start\_date, end\_date)’ itera para que en cada

día se encuentren todos los períodos de tiempo en los que no hay ningún evento en curso.

Ilustración 104. Implementación de 'calculate\_free\_time(busy\_time, start\_date, end\_date)'.

```
def calculate_free_time(busy_time, start_date, end_date):
    free_time = []
    current_date = start_date
    current_time = datetime.min.time()
    end_time = datetime.max.time()

    while current_date <= end_date:
        daily_busy = [time for time in busy_time if time[0] == current_date]
        daily_busy = sorted(daily_busy, key=lambda x: x[1])

        last_end_time = current_time
        for busy in daily_busy:
            busy_start_time, busy_end_time = busy[1], busy[2]
            if last_end_time < busy_start_time:
                free_time.append([current_date, last_end_time, busy_start_time])
            last_end_time = max(last_end_time, busy_end_time)

        if last_end_time < end_time:
            free_time.append([current_date, last_end_time, end_time])

        current_date += timedelta(days=1)

    return free_time
```

Fuente: Elaboración propia.

La función 'allocate\_tasks(objectives, free\_time)' lo primero que hará será encontrar las tareas que ya se han completado en la semana actual. Esto se debe a que, si se decide planificar un miércoles, la planificación se realizará entre el miércoles y el domingo de esa semana y el lunes y el domingo de la siguiente. Es importante conocer cuáles son las tareas ya completadas para que el algoritmo sepa cuántas horas por objetivo restan por asignar esa primera semana. Para ello, en el diccionario 'task\_hours\_by\_week' se almacenan las horas totales ya dedicadas en cada semana a cada objetivo. Una vez obtenida esa información, se diferencian los horarios disponibles totales entre los que restan de la primera semana y la segunda y se almacenan en el diccionario 'free\_time\_by\_week' donde la clave será la semana del año (por ejemplo, la quinta semana del año será la 5) y el valor, una lista con todos los horarios disponibles esa semana. A partir de ahí, se procede a iterar por objetivos de mayor a menor prioridad. Primero, se asignarán las tareas en orden de prioridad de cada objetivo. Los usuarios deberán ir constantemente modificando las prioridades de sus objetivos y tareas antes de planificar según crean conveniente para que aparezcan en mayor o menor medida en su planificación. Para cada tarea, se iteran sus horarios de disponibilidad. Primeramente, se comprueba si ya se han acumulado las horas semanales máximas por objetivo. En tal caso, se salta a la siguiente semana. Por el contrario, si aún quedan horas por asignar, se iteran los intervalos de horarios disponibles y se ajustan los tiempos de comienzo y finalización de la tarea en ese intervalo. Si hay tiempo disponible en el intervalo para realizar la tarea, se asigna una cantidad de horas sin exceder el número de horas mínimo y máximo de dedicación a la tarea. Al asignar la tarea, ésta se almacena en la lista 'allocated\_tasks' y se actualizan los intervalos de tiempo libre para la siguiente iteración. Finalmente, se devuelve la lista

'allocated\_tasks' en forma de diccionario con los detalles de todas las tareas que se han asignado.

Ilustración 105. Implementación de 'allocate\_tasks(objectives, free\_time)'.

```
def allocate_tasks(objectives, free_time):
    user_id = session.get('user_id')
    allocated_tasks = []
    completed_tasks = AllocatedTask.query.filter_by(user_id=user_id).filter(AllocatedTask.date >= start_date,
                                                                           AllocatedTask.date <= end_date,
                                                                           AllocatedTask.completed == True).all()

    task_hours_by_week = {}

    for completed_task in completed_tasks:
        task_week_number = completed_task.date.isocalendar()[1]
        objective_id = completed_task.task.objective_id

        if task_week_number not in task_hours_by_week:
            task_hours_by_week[task_week_number] = {}

        if objective_id not in task_hours_by_week[task_week_number]:
            task_hours_by_week[task_week_number][objective_id] = 0

        task_hours_by_week[task_week_number][objective_id] += completed_task.dedicated_hours

    free_time_by_week = {}
    for interval in free_time:
        free_date, free_start_time, free_end_time = interval
        week_number = free_date.isocalendar()[1]

        if week_number not in free_time_by_week:
            free_time_by_week[week_number] = []
        free_time_by_week[week_number].append(interval)

    objectives = sorted(objectives, key=lambda objective: int(objective['priority']), reverse=True)
    for objective in objectives:
        weekly_hours = {week: 0 for week in free_time_by_week.keys()}
        for week, objective_hours in task_hours_by_week.items():
            if objective['id'] in objective_hours:
                weekly_hours[week] += objective_hours[objective['id']]

    tasks = sorted(objective['tasks'], key=lambda task: int(task['priority']), reverse=True)

    for task in tasks:
        for task_schedule in task['schedules']:
            day_of_week = get_weekday_number(task_schedule['day_of_week'].value)
            task_start_time = task_schedule['start_time']
            task_end_time = task_schedule['end_time']

            for week, free_intervals in free_time_by_week.items():
                if weekly_hours[week] >= objective['hours']:
                    continue

                max_available_hours = objective['hours'] - weekly_hours[week]
                if max_available_hours <= 0:
                    continue

                for free_interval in list(free_intervals):
                    free_date, free_start_time, free_end_time = free_interval
                    if free_date.weekday() != day_of_week:
                        continue

                    possible_start_time = max(task_start_time, free_start_time)
                    possible_end_time = min(task_end_time, free_end_time)

                    if possible_start_time >= possible_end_time:
                        continue

                    time_for_task = (datetime.combine(start_date, possible_end_time) - datetime.combine(start_date, possible_start_time)).to timedelta()

                    dedicated_hours = min(task['max_hours'], max(task['min_hours'], time_for_task), max_available_hours)

                    if dedicated_hours >= task['min_hours']:
                        allocated_task = {
                            'task_id': task['id'],
                            'task_name': task['name'],
                            'day': free_date,
                            'start_time': possible_start_time,
                            'end_time': (datetime.combine(start_date, possible_start_time) + timedelta(hours=dedicated_hours)).time(),
                            'dedicated_hours': dedicated_hours
                        }
                        allocated_tasks.append(allocated_task)
                        weekly_hours[week] += dedicated_hours
```

```

        free_intervals.remove(free_interval)

        if possible_start_time > free_start_time:
            free_intervals.append((free_date, free_start_time, possible_start_time))
        if possible_end_time < free_end_time:
            free_intervals.append((free_date, possible_end_time, free_end_time))

    free_intervals.sort(key=lambda x: x[1])
    break

return allocated_tasks

```

Fuente: Elaboración propia.

## 5.6. Otras implementaciones

### 5.6.1. Almacenamiento de imágenes

Un usuario podrá cargar archivos de tipo imagen a la hora de registrarse para añadir una foto de perfil y al crear un objetivo para añadir una imagen de objetivo. Para evitar conflictos y optimizar la gestión de este tipo de archivos, las imágenes cargadas desde los dispositivos de los usuarios son almacenadas con nombres de archivo únicos. Para ello, en caso de haber seleccionado una imagen en los formularios, se le aplica un UUID (Universally Unique Identifier) haciendo uso de la librería ‘uuid’ de Python. Esto generará una clave aleatoria única que garantizará que ningún nuevo archivo coincida en nombre con uno existente. En el caso de las imágenes de perfil, éstas se almacenarán bajo el nombre ‘[nombre de usuario]\_[UUID generado].[extensión del archivo cargado]’ en el directorio ‘static/images/profile’. El nombre de un usuario es único y no se debe poder repetir, pero se le aplica el UUID como segunda capa para garantizar que no se puede repetir el nombre del archivo. Por su parte, los objetivos se almacenarán bajo el nombre ‘[nombre del objetivo]\_[identificador del usuario]\_[UUID generado].[extensión del archivo cargado]’ en el directorio ‘static/images/objectives’.

*Ilustración 106. Creación del nombre del archivo de foto de perfil.*

```

if image:
    extension = os.path.splitext(image.filename)[1]
    filename = f"{username}_{uuid.uuid4().hex}{extension}"
    image_path = os.path.join(app.config['PROFILE_IMAGES'], filename)
    image.save(image_path)
else:
    filename = None

```

Fuente: Elaboración propia.

### 5.6.2. Inicio de sesión obligatorio para acceder a las funcionalidades

Para evitar que un usuario no registrado pueda acceder a las funcionalidades de la aplicación a las que solo se puede acceder tras iniciar sesión, se implementa una capa de seguridad que al abrir cada una de estas funcionalidades como la

agenda, la página de objetivos o el perfil, la primera acción que se realiza es comprobar el identificador del usuario de la sesión actual. En caso de no haber una sesión iniciada, se dirigirá al usuario a la página de inicio de sesión. Esto trata de evitar que un usuario pueda acceder a cierto contenido a través de la URL o accediendo a las páginas desde el footer en el Portal de Usuario, página de registro o página de inicio de sesión. Páginas accesibles por todos los usuarios además de las recientemente mencionadas son ayuda, Aviso Legal o Política de Privacidad.

Ilustración 107. Comprobación de que hay una sesión iniciada.

```
@app.route('/agenda/<int:offset>', methods=['GET'])
def agenda(offset=0):
    user_id = session.get('user_id')
    user = User.query.get(user_id)
    if not user:
        return redirect(url_for('login'))
```

Fuente: Elaboración propia.

### 5.6.3. Lectores de pantalla

Con el objetivo de cumplir con las pautas marcadas por WCAG 2.2, se han implementado textos alternativos y etiquetas ARIA (Accesible Rich Internet Applications) para hacer las páginas lo más perceptibles posible para un usuario con discapacidad visual. Los textos alternativos se han aplicado a todas las imágenes (objetivos y perfil) que se muestran en las diferentes páginas de la aplicación. Por su parte, las etiquetas ARIA se han aplicado a todos los botones e iconos. Gracias a estas etiquetas, los lectores de pantalla podrán a través de la salida de audio del dispositivo del usuario comunicar la descripción del contenido.

Ilustración 108. Textos alternativos en las imágenes de objetivos.

```
<div class="objective">
    <div class="image-info">
        {% if objective.image %}
            
        {% else %}
            
        {% endif %}</div>
```

Fuente: Elaboración propia.

Ilustración 109. Etiquetas ARIA en los botones para añadir una actividad o rutina

```
<div class="buttons">
    <button id="activity" class="btnAdd">
        <i class="fas fa-plus-circle" aria-label="Abrir panel con el formulario para añadir una actividad "></i>Añadir Actividad</button>
    <button id="routine" class="btnAdd">
        <i class="far fa-calendar-plus" aria-label="Abrir panel con el formulario para añadir una rutina"></i>Añadir Rutina</button>
</div>
```

Fuente: Elaboración propia

Ilustración 110. Lectura de pantalla con VoiceOver.



Fuente: Elaboración propia.

## 6. Conclusiones

### 6.1. Conclusiones

Al principio de este trabajo, se plantearon tres objetivos principales:

1. Desarrollar un algoritmo eficiente que planifique la agenda de un usuario en base a sus objetivos.
2. Desarrollar una aplicación web accesible.
3. Desarrollar una aplicación web segura.

Se ha desarrollado un algoritmo con un tiempo de respuesta corto y eficaz para la asignación de las tareas. Al asignar tareas para un tiempo máximo de dos semanas, se ha reducido mucho el tiempo de ejecución. Además, la asignación cumple con las pautas marcadas de tener una duración semanal concreta para cada objetivo y el número de horas mínimo y máximo y los horarios disponibles de cada tarea. El margen de mejora radica en aquellas tareas que se puedan realizar entre dos días. Por ejemplo, una tarea que se pueda comenzar a realizar a las 22:00 de un día y acabar a las 02:00 del día siguiente. Esto se debe a que, al calcular los intervalos de tiempo disponible, esto se realiza diariamente por lo que si hay disponibilidad entre las 20:00 de un día y las 08:00 del día siguiente, se generarán dos intervalos: 20:00-23:59 y de 00:00-08:00. Pese a ello, considero que se ha desarrollado un buen proceso para la asignación de estas tareas. Además, considero que el código del algoritmo está muy bien organizado y estructurado para realizar mejoras en el futuro por diferentes desarrolladores.

A nivel de accesibilidad, se optó por seguir las pautas WCAG 2.2. en las que se establecían diferentes requisitos para cumplir sus estándares de accesibilidad. Para que sea perceptible, se han aplicado textos alternativos y etiquetas ARIA lo cual hace la aplicación utilizable para aquellos usuarios con capacidad visual que requieran un lector de pantalla. En esta aplicación no existen audios o vídeos que deban ir acompañados de subtítulos o transcripciones por lo que no se aplica

ninguna implementación para los medios basados en el tiempo. Para que sea distingible, se ha adoptado un diseño con colores muy diferenciados entre ellos para que el contenido sea legible y haya un contraste óptimo. También, los botones son fácilmente identificables por su forma, color y reacción al movimiento del cursor sobre ellos. A nivel de percepción, se debería haber optimizado la adaptabilidad a diferentes dispositivos con un diseño responsive. En lo que refiere a la operatividad, todo el contenido es accesible por teclado. Además, al tener páginas sencillas sin una gran sobrecarga, es rápido acceder a todas las funcionalidades a través del teclado. Ningún contenido desaparece automáticamente en un periodo corto de tiempo ni hay parpadeos que puedan generar convulsiones. Se ha tratado de crear una navegabilidad muy cómoda para que los usuarios encuentren lo que buscan de forma intuitiva al ponerle textos descriptivos a los botones además de iconos. El margen de mejora a nivel de accesibilidad radica en aplicar diferentes modalidades de entrada como controladores de voz o dispositivos de seguimiento ocular. Para que la aplicación sea comprensible, se han tratado de seguir siempre las pautas marcadas. Se ha empleado un léxico sencillo y bien estructurado para que toda la información se entienda de la manera más clara posible, se han aplicado hovers, iconos y textos en los botones para que antes de interactuar el usuario pueda prever la acción que va a desencadenar y al llenar los formularios, se han creado los cuadros modales de ayuda en cada página explicando toda la información relativa a esa página o formulario y se emiten alertas y avisos cuando falta información por cumplimentar o la información introducida no es válida. Haber optado por desarrollar la aplicación con Flask, HTML, CSS y JavaScript ha permitido cumplir con el principio de robustez. Estas tecnologías siguen los estándares web y garantizan la compatibilidad y por ello, correcto funcionamiento con diferentes tipos de dispositivos y tecnologías. Por todo esto, mi grado de satisfacción con la accesibilidad de la aplicación es muy alto, aunque aún tiene margen de mejora en la optimización de los diseños en dispositivos de diferentes tamaños y en la interacción a través de otros dispositivos de entrada que no sean el teclado y el ratón. Considero que la no discriminación y desarrollar aplicaciones que todas las personas sin importar su condición tengan la oportunidad de disfrutar es un principio básico a la hora de desarrollar software.

A nivel de seguridad, se optó por basarse en la información proporcionada por OWASP. Gracias a su identificación de los diez mayores riesgos para una aplicación web y sus guías con prácticas recomendadas, se ha podido aplicar un amplio número de mecanismos de defensa para las vulnerabilidades potenciales. Para evitar fallas criptográficas, se han almacenado las contraseñas de los usuarios usando un algoritmo de hash que proteja esta información sensible en caso de que alguien sin autorización acceda a esta información. Además, ninguna de las tecnologías que se han aplicado en este proyecto es vulnerable o está obsoleta. Se han descargado todas las librerías de las fuentes oficiales que las distribuyen. Para evitar fallas en la identificación de un usuario se ha implementado un exhaustivo control en las contraseñas. Las condiciones establecidas para crear la contraseña evitan que ninguna de las contraseñas de un usuario sea débil. Al requerir diferentes tipos de caracteres como letras, números o símbolos además de una longitud considerable de mínimo ocho caracteres, se protegerá al usuario de sufrir ataques por fuerza bruta. Finalmente, para proteger la base de datos ante los ataques de inyección SQL,

se ha optado por emplear SQLAlchemy. Este ORM permite que las consultas a la base de datos sean directamente parametrizadas en lugar de acceder a la información con consultas SQL. Esto no lo hace solo más sencillo a nivel de desarrollo, sino que también previene este tipo de ataques tan popular y del cual esta aplicación podría ser muy vulnerable por la gran cantidad de formularios que posee. En cómputo global, estoy satisfecho con las medidas tomadas hasta el momento. Igualmente, pienso que aún hay muchas medidas indicadas en las ‘cheat sheets’ proporcionadas por OWASP como la monitorización a través de logs que se pueden aplicar antes de la puesta en producción de la aplicación. Considero que la seguridad de un sitio web es fundamental y que debe estar en constante revisión y mejora. Constantemente se crean nuevos tipos de ataque y es importante estar informado de cuáles son y qué medidas tomar para mitigarlos.

Como conclusión general, creo que se han cumplido mayoritariamente todos sus objetivos principales. Por ello, considero la elaboración de este proyecto como un éxito. También, considero que las tecnologías empleadas y la metodología aplicada han sido una decisión acertada teniendo en cuenta la experiencia con las que contaba con ellas. Ha habido cambios en los requisitos durante el transcurso del proyecto al identificar nuevas funcionalidades u otras necesidades mientras el desarrollo estaba en curso. Por ello, considero como principal aspecto a mejorar el tener todos los fundamentos de la aplicación claros y bien definidos desde el comienzo para evitar tener que prescindir de ciertos elementos ya desarrollados lo cual supuso una pérdida de tiempo o el tener que retomar tareas que ya se consideraban como finalizadas para añadir algún campo o funcionalidad.

## 6.2. Trabajos Futuros

La aplicación tiene aún un gran margen de crecimiento por el potencial que posee. A continuación, listaré algunas mejoras que se le podrían aplicar a la aplicación web a corto, medio o largo plazo:

### **Mejora del algoritmo:**

El algoritmo para asignar las tareas a horarios en fechas concretas considero que siempre tendrá margen de mejora. A los parámetros actualmente establecidos, se le podrían añadir nuevos. Por ejemplo, tener en cuenta la motivación actual que tiene un usuario con un objetivo. Basado en el estado emocional establecido por el usuario, las decisiones para asignar objetivos podrían variar. Un parámetro ya establecido que actualmente es simplemente informativo es la ubicación de la tarea. Considero que sería muy interesante continuar el desarrollo del algoritmo basado en la ubicación de los eventos y que, dependiendo de ésta, se asignen las tareas considerando la distancia, el tiempo y el método de transporte.

### **Mejoras en la accesibilidad:**

Optimizar las vistas de usuario para todos los tamaños de dispositivo sería clave para lograr cumplir las pautas de accesibilidad marcadas por WCAG 2.2. Esto

permitirá al usuario tener una experiencia completamente satisfactoria sin importar si accede a la web a través de un dispositivo móvil o una tableta. También, implementar la interacción a través del máximo número de dispositivos de entrada posible.

**Exportar la planificación:**

Que un usuario pueda recibir su planificación a través de mensajes de WhatsApp, correo electrónico o que pueda descargar su planificación en formato PDF haría que la agenda con la planificación semanal fuese siempre accesible cuando el usuario no pueda acceder al ordenador o en momentos en los que no pueda tener acceso a internet.

**Administrador:**

Antes de la puesta en producción, sería ideal desarrollar una vista para los administradores de la aplicación en la que éstos puedan con facilidad crear, editar y eliminar datos de los usuarios y monitorizar los registros de actividad en la web a través de logs para detectar y tener constancia de comportamientos anómalos.

**Notificaciones:**

Alertar a través de las notificaciones información relevante sobre la planificación. Por ejemplo, si una tarea no se puede asignar.

**Integraciones:**

Al igual que han aplicado muchas de las aplicaciones webs estudiadas en el mercado, sería interesante poder integrar los calendarios de otras plataformas en la agenda del usuario como Google Calendar o Outlook Calendar. Esto facilitaría mucho la migración de los eventos de un usuario que actualmente emplee una de estas herramientas e incentivaría a usar esta aplicación web.

**Idioma:**

Actualmente, la aplicación tan solo está disponible en español. El uso de esta aplicación es global ya que sería útil para personas de cualquier lugar del mundo puesto que fijarse objetivos personales no entiende de países o fronteras. Por ello, al traducir los textos a diferentes lenguas, el público potencial de la aplicación crecerá enormemente.

**Recuperación de la contraseña:**

Actualmente, no existe ningún método de recuperación de contraseña. Empleando el correo electrónico y el número de teléfono posiblemente proporcionado, un usuario podría iniciar un proceso en caso de querer cambiar la contraseña o en caso de no recordarla.

**Cookies:**

Con el objetivo de mejorar la experiencia del usuario, se podrían almacenar cookies en el dispositivo del usuario para recordar detalles como las preferencias del usuario o los datos de inicio de sesión. Esto conllevaría desarrollar una Política de Cookies.

## Bibliografía

- Campbell, A., Adams, C., Bradley Montgomery, R., Cooper, M., Kirkpatrick, A., (Eds.). (12 de diciembre de 2024). *Web Content Accessibility Guidelines (WCAG) 2.2.* Obtenido de W3C: <https://www.w3.org/TR/WCAG22/>
- Escuela de Ingeniería Informática. Universidad de Las Palmas de Gran Canaria. (5 de marzo de 2019). *Grado en Ingeniería Informática.* Obtenido de chrome-extension://efaidnbmnnibpcajpcglclefindmkaj/<https://www.eii.ulpgc.es/sites/default/files/2022-05/Grado%20en%20Ingenier%C3%ADA%20inform%C3%A1tica%20-%20memoria%20del%20plan%20de%20estudios%20-%202019.pdf>
- Fontawesome. (s.f.). Obtenido de Fontawesome: <https://fontawesome.com/>
- Freepik. (s.f.). Obtenido de Flaticon: [https://www.flaticon.es/icono-gratis/diana\\_1230178](https://www.flaticon.es/icono-gratis/diana_1230178)
- Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico. (12 de julio de 2002). Obtenido de Boletín Oficial del Estado: <https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758>
- Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. (6 de diciembre de 2018). Obtenido de Boletín Oficial del Estado: <https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673>
- OWASP. (s.f.a). *About the OWASP Foundation.* Obtenido de OWASP: <https://owasp.org/about/>
- OWASP. (s.f.b). *OWASP Top Ten 2021 : Related Cheat Sheets.* Obtenido de OWASP: <https://cheatsheetsseries.owasp.org/IndexTopTen.html>
- PNGWing. (s.f.). Obtenido de PNGWing: <https://www.pngwing.com/es/search?q=perfil+del+usuario>
- Quickstart. (s.f.). Obtenido de Flask: <https://flask.palletsprojects.com/en/stable/quickstart/>
- Radigan, D. (s.f.). *How the kanban methodology applies to software development.* Obtenido de Atlassian: <https://www.atlassian.com/agile/kanban>
- Rebelo, M. (23 de julio de 2024). *The 8 best AI scheduling assistants in 2024.* Obtenido de Zapier: <https://zapier.com/blog/best-ai-scheduling/>
- Tan, S. W. (30 de diciembre de 2020). *New Year's resolution tips to create SMART goals, successful results.* Obtenido de The Washington Times: <https://www.washingtontimes.com/news/2020/dec/30/new-years-resolution-tips-create-smart-goals-succe/>
- The Python SQL Toolkit and Object Relational Mapper. (s.f.). Obtenido de SQLAlchemy: <https://www.sqlalchemy.org/>
- Utilities. (s.f.). Obtenido de Werkzeug: <https://werkzeug.palletsprojects.com/en/stable/utils/>
- Vacanti, D., & Coleman, J. (diciembre de 2020). *Kanban Guide in English.* Obtenido de Kanban Guides: <https://kanbanguides.org/english/>